Verifying Probabilistic Procedural Programs

Javier Esparza[1] and Kousha Etessami[2]

[1] Institute for Formal Methods in Computer Science,
University of Stuttgart
[2] School of Informatics,
University of Edinburgh

Abstract. Monolithic finite-state probabilistic programs have been abstractly modeled by finite Markov chains, and the algorithmic verification problems for them have been investigated very extensively. In this paper we survey recent work conducted by the authors together with colleagues on the algorithmic verification of probabilistic procedural programs ([BKS, EKM04, EY04]). Probabilistic procedural programs can more naturally be modeled by *recursive Markov chains* ([EY04]), or equivalently, *probabilistic pushdown automata* ([EKM04]). A very rich theory emerges for these models. While our recent work solves a number of verification problems for these models, many intriguing questions remain open.

1 Introduction

The topic of this paper is the decidability and computational complexity of verification problems for models of probabilistic programs. Loosely speaking, a program is probabilistic if it can flip a coin in order to decide the next execution step. Probabilistic models of programs are of interest for at least two reasons. First, we may wish to model and analyze randomized algorithms, which are intrinsically probabilistic. Second, sometimes when we model a program's behavior we may wish to replace a deterministic branching choice by a probabilistic one in order to obtain information about the induced probability of certain behaviors, e.g., that the program terminates in a certain state. The probabilities chosen for the branches may either be subjective choices or be based, e.g., on statistics accumulated from profiling data.

As usual, in the area of automated software verification we assume that the variables of the program have a finite domain, either because the program was so designed, or because it is an abstraction of another program. The complexity of probabilistic verification has been extensively studied for finite-state *flat-programs*, where the program consists of one procedure containing no procedure calls, and the control mechanisms are only the basic **if-then-else** instructions and **while** loops.

In this case, the program has a finite number of states, and can be modeled abstractly by a finite Markov Chain. There is already an extensive literature on analysis of such models (see, e.g., [Var85, CY95, Kwi03]). Since last year, both

K. Lodaya and M. Mahajan (Eds.): FSTTCS 2004, LNCS 3328, pp. 16–31, 2004.

authors, together with colleagues, have initiated a study of verification problems for programs with multiple (possibly recursive) procedures, called *procedural programs* in this paper [EKM04, EY04]. Since the state of such a program must contain information about the stack of calls that have not yet been completed, the state space is potentially infinite, and so these programs are more naturally modeled by countably infinite Markov chains of a certain kind. As we will see, verification questions related to these models lead to very interesting algorithmic and mathematical problems.

In this paper we survey our published results and report on our work in progress [EKM04, EY04, BKS]. While a number of interesting algorithmic questions have been answered, many questions remain. We use this opportunity to emphasize the intuition behind the results and avoid the technicalities.

2 Models of Probabilistic Programs

The *state* of a running program, as usual, consists of the contents of memory together with the location of the program's control. This defines a *state transition system*, whose transitions are from a state s to a state t whenever the program can move in one step from s to t.

In the case of probabilistic programs we assume that transitions are labelled with a positive probability, i.e., a number in the interval $(0, 1]$, and that the sum of the probabilities attached to the transitions leaving a state is 1, or 0 if the state is a halting state. This transforms the state space into a *Markov chain*.

A state of a flat-program contains information about the current control point and the current values of the variables. If we assume that variables have a finite domain, as we always do in this paper, the state space of the program is finite, and so probabilistic flat-programs can be modelled as finite Markov chains.

Let us now discuss formal models for probabilistic programs with procedures. When a procedure or function Q is called from another procedure or function P, with parameter values v passed from P to Q,

(1) the return address (i.e. the point of P to control has to return after completion of the call) and the current values of the local variables of P are stored as an *activation record* on the *call stack*;
(2) control is transferred to an initial control point of Q, and the passed parameter values v can be treated as a value of a local variable of Q.
(3) upon completion of the call, control is transferred to the return address, and the values of the local variables of P are restored according to the top activation record on the call stack, and if the procedure Q returned a value r, the value r is passed back to P in a local variable.

Thus, the state of a program with procedures contains information about the current control point, the current values of the variables, and the current contents of the call stack. We may represent a state by a triple (g, l, r), where g represents the current values of the global variables, l the control point and values of the local variables of the current procedure (which may include parameters passed to it, or values returned to it), and r is a sequence of activation records, with the top of the stack as first element of the sequence. Since the stack size is not bounded a priori, the program may have an infinite state space, and the Markov chain associated with a program with procedures may be infinite.

As in the case of finite flat-programs, we assume that transitions are labelled with a positive probability. We also assume that the probability of a transition $(q_1, l_1, r_1) \rightarrow (g_2, l_2, r_2)$ depends only on g_1, l_1 and g_2, l_2. Intuitively, this means that the probability of executing a particular instruction of the program code only depends on the current program control point and the current values of the program variables, and not on the contexts of the call stack. There are some special situations in which one might like to weaken this condition (for instance, some methods of the Java Development Kit inspect the stack of activation records [BJMT01]), but even in this case, using coding tricks, one can construct equivalent Markov chains satisfying it.

When the number of control points and the domains of program variables are finite, such probabilistic procedural programs induce a particular family of infinite Markov chains. We can not work directly with infinite Markov chains, but need to work with finite representations of them. We consider two equivalent finitely presented models of these Markov chains: *Probabilistic Pushdown Automata* (PPDAs) (studied in [EKM04]) and *Recursive Markov Chains* (RMCs) (studied in [EY04]). These models have non-probabilistic counterparts which have been studied extensively in recent research on verification and program analysis: for *Pushdown Systems* (PDAs) see, e.g., [BEM97, EHRS00], and for *Recursive State Machines* (RSMs) see [AEY01, BGR01].

2.1 Recursive Markov Chains

A *recursive Markov chain* is a tuple $(A_1, \ldots, A_k)$, where each A_i is a *component*. Each component models a procedure of the program and consists of:

- A set of *nodes*, with two distinguished subsets of *entry* and *exit* nodes.
- A set of *boxes*. A box b is labelled with an integer $Y(b) \in \{1, .., k\}$, and has a *call port*, or just a *call* (en, b) for each entry node en of $A_{Y(b)}$, and a *return port*, or just a *return* (ex, b) for each exit node ex of $A_{Y(b)}$.
- A set of transitions $u \xrightarrow{x} v$ where
    - u is either a non-exit node or a call port,
    - v is either non-entry node, or a return port, and
    - x is a positive probability, with the condition that the sum of the probabilities of all the transitions having source u is 1 or 0, if the vertex u is an exit node or call port, which has no outgoing edges in A_i.

Recursive Markov chains reflect the structure of a program with procedures. Each procedure is modelled by a component. A node corresponds to a local state of the procedure, i.e., to one of its control points and a valuation of its local variables, plus the values of the global variables, if any. Entry nodes correspond to the possible initial states of the procedures, which also reflect the parameter values passed to it, and exit nodes correspond to local states from which control is returned to the caller and the returned value. A transition to a call port (en, b)

of a box b labeled by $Y(b) = i$ models a call with particular parameter values reflected by the node en, to the procedure modelled by A_i. Similarly, a transition from a return port corresponds to a return, with particular return values.

A RMC $A = (A_1, \ldots, A_k)$ defines a (possibly infinite) Markov chain M_A as follows. Let a *vertex* be either a node, a call, or a return. The states of M_A, which we call *global states*, are pairs $\langle u, B \rangle$, where u is a vertex and $B = b_1 \ldots b_n$ is a sequence of boxes. M has the following transitions:

- a transition $\langle u, B \rangle \xrightarrow{x} \langle u', B \rangle$ for every transition $u \xrightarrow{x} u'$ and every sequence of boxes B;
- a transition $\langle (en, b), B \rangle \xrightarrow{1} \langle en, bB \rangle$ for every call port (b, en), and every sequence of boxes B; and
- a transition $\langle ex, bB \rangle \xrightarrow{1} \langle (ex, b), B \rangle$ for every return port (ex, b), and every sequence of boxes B.

RMCs can be depicted visually in a natural way. An example RMC is in Figure 1. This RMC has only one component, $A1$. It contains two nodes: entry en and exit ex, and two boxes, $b1$, and $b2$, both of which are labeled by the same component, $A1$, i.e., $Y(b1) = Y(b2) = 1$. Each box bi has a call port (en, bi), and a return port (ex, bi). (In this example, it so happens that the probability of reaching the state $\langle ex, \epsilon \rangle$ from $\langle en, \epsilon \rangle$ is $1/2$. We will see why this is the case later.)

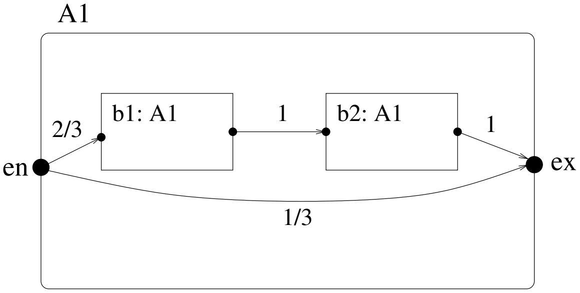

Fig. 1. An example RMC, A

2.2 Probabilistic Pushdown Automata

A probabilistic pushdown automaton (pPDA) consists of

- a finite set of *control states*,
- a finite *stack alphabet*, and
- a finite set of *rules* of the form $pX \xrightarrow{x} q\alpha$, where p and q are control states, X is a stack symbol, α is a word of stack symbols, and x is a positive probability. The left hand side pX of a rule is the rule's *head*, or just a head, for short.

A pPDA defines a Markov chain M: the states of M are pairs $\langle p, S \rangle$, where p is a control state and S is a sequence of stack symbols. M has a transition $\langle p, X\beta \rangle \xrightarrow{x} \langle q, \alpha\beta \rangle$ for every rule $pX \xrightarrow{x} q\alpha$ and sequence of stack symbols β.

We can easily transform a RMC into a pPDA. It suffices to take the set of vertices as control states, boxes as stack symbols, and the following set of rules:

– a rule $ub \xrightarrow{x} u'$ for every transition $u \xrightarrow{x} u'$ and every box b,
– a rule $(en, b)b' \xrightarrow{1} en\, bb'$ for every call port (b, en) and every box b', and
– a rule $ex\, b \xrightarrow{1} (ex, b)$ for every exit node ex and every box b.

In this paper we assume that pPDA are in the following *normal form*:

– for every rule $pX \xrightarrow{x} q\alpha$, α has length at most 2, and
– for every p and every X at least one rule has pX as left hand side.

Note that the pPDAs obtained from RMCs by the translation above are in normal form. Normal form pPDAs can also be transformed to RMCs of the same size, by mimicking a translation of PDAs to RSMs given in [AEY01]. Thus RMCs have a tight correspondence to normal form pPDAs.[1] Every pPDA can also be put in normal form in linear time while preserving all properties of interest.

3 Reachability

Given two states s_0, s_f of a probabilistic sequential program, let $[s_0, s_f]$ denote the probability of eventually reaching s_f starting from s_0. We wish to answer the following questions:

(1) The *qualitative* reachability problem: Is $[s_0, s_f] = 1$?
(2) The *quantitative* reachability problem: Given $\rho \in (0, 1]$, is $[s_0, s_f] \geq \rho$?
We may also wish to compute or approximate the probability $[s_0, s_f]$.

3.1 Flat Programs

In the case of flat-programs, s_0 and s_f are states of a finite Markov chain M. The answers to (1) and (2) are well-known, but we quickly recall them in order to compare them with the answers in the procedural case.

In a finite Markov chain $[s_0, s_f] = 1$ holds if and only if either (a) there is a single Bottom Strongly Connected Component (BSCC) of M that is reachable from s_0 and s_f belongs to that component, or (b) s_f belongs to every path from s_0 to any BSCC, i.e., removing s_f makes all BSCC's unreachable from s_0. These properties can be checked in linear time using standard graph algorithms. Observe that whether $[s_0, s_f] = 1$ or not depends only on the topology of the Markov chain, and not on the probabilities labelling the transitions.

Let us consider now the quantitative problem. Assume that the transitions leaving s_0 are $s_0 \xrightarrow{p_1} s_1, \ldots, s_0 \xrightarrow{p_k} s_k$. We have the following equation:

$$[s_0, s_f] = p_1 \cdot [s_1, s_f] + \ldots + p_k[s_k, s_f]$$

If we write down the same equation for every pair s, s', and look at the terms $[s, s']$ as unknowns, we obtain a linear system of equations $\mathbf{x} = L(\mathbf{x})$,

[1] There is a minor loss of information of the structure of the RMC when going from an RMC to a pPDA (see [AEY01]).

in m unknowns $\mathbf{x} = (x_1, \ldots, x_m)$, where each variable x_i corresponds to some unknown probability $[s, s']$. It can be show that the probabilities we wish to compute are given by the least non-negative solution for this system, by which we mean a vector $q = (q_1, \ldots, q_m) \in \mathbb{R}^m_{\geq 0}$, such that $q = L(q)$, and such that if $v \in \mathbb{R}^m_{\geq 0}$ is another solution then $q_i \leq v_i$ for all i, $1 \leq i \leq m$. We will see a generalization of this when we study RMCs and pPDAs.

There are a number of ways to compute this least solution. Since the system is linear, the least solution is rational, and one could use, e.g., linear programming methods to compute it. More efficiently, it turns out the system can be transformed into another one such that the least solution of the old system is the *unique* solution of the new system. The new system can then be solved using, e.g., Gaussian elimination, or its solution can be approximated efficiently using iterative numerical procedures like Jacobi or Gauss-Seidel, etc.

3.2 Procedural Programs

When we try to generalize the answers for probabilistic flat programs to the procedural case, we quickly encounter a number of obstacles. To begin with, the answer to the qualitative problem is no longer independent of the values of the probabilities, as shown by the following example. Consider the pPDA given by the rules.

$$pX \xrightarrow{x} pXX$$
$$pX \xrightarrow{1-x} p\epsilon$$

(We could also take the RMC A depicted in Figure 1, with the probabilities 2/3 and 1/3 replaced by x and $1-x$, respectively.) The infinite Markov chain defined by this pPDA corresponds to a 'truncated' Bernoulli walk, depicted in Figure 2, and a standard result states that $[pX, p\epsilon] = 1$ if and only if $x \leq 1/2$. So the answer to the qualitative problem "is $[pX, p\epsilon] = 1$?" depends not only on the topology of the RMC, but also on the value of x.

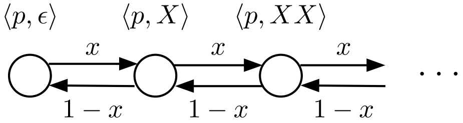

Fig. 2. The Markov chain of a pPDA

For computing these probabilities we can not simply proceed as in the finite case to write down one linear equation for each probability $[s, s']$, for every pair s, s', because the Markov chain is in general infinite and this would lead us to an infinite system of linear equations in infinitely many variables.

For the moment, let us consider a simpler problem: given a vertex u and an exit node ex of the same component A_i, what is the probability of starting at the global state $\langle u, \epsilon \rangle$, eventually reaching the state $\langle ex, \epsilon \rangle$? Let us denote this probability by $[u, ex]$. Although we do not expand on it in this survey, computing(or approximating) these probabilities is sufficient to allow us to compute (approximate) reachability probabilities between other pairs of states s, s'.

Consider three cases of what $[u, ex]$ might be, based on the vertex u:

- $u = ex$. Then $[u, ex] = 1$.
- If u is a node or a return port, and the transitions leaving u are $u \xrightarrow{p_1} v_1, \ldots, v \xrightarrow{p_n} v_n$. Then, as in the case of a finite Markov chain,

$$[u, ex] = p_1 \cdot [v_1, ex] + \ldots + p_n \cdot [v_n, ex]$$

- $u = (b, en)$ is a call port of a box b corresponding to a component A_j. Then, in order to reach $\langle ex, \epsilon\rangle$ from $\langle (b, en), \epsilon\rangle$, we must follow a path of the form

$$\langle (en, b), \epsilon\rangle \xrightarrow{1} \langle en, b\rangle \cdots \langle ex', b\rangle \xrightarrow{1} \langle (ex', b), \epsilon\rangle$$

If the exit nodes of the component A_j are $ex'_1, \ldots, ex'_n$, then, since the probability of eventually reaching $\langle ex', b\rangle$ from $\langle en, b\rangle$ is equal to the probability of eventually reaching $\langle ex', \epsilon\rangle$ from $\langle en, \epsilon\rangle$, we get

$$[(en, b), ex] = [en, ex'_1] \cdot [(ex'_1, b), ex] + \ldots + [en, ex'_n] \cdot [(ex'_n, b), ex]$$

We thus have a finite system of non-linear multi-variate polynomial equations for the unknowns $[u, ex]$, ranging over every pair u, ex, where u is a vertex of the RMC and ex is an exit node of the same component. Lets associate each unknown probability $[u, ex]$ with a corresponding variable $x_{[u,ex]}$. For convenience we index these variables $x_1, \ldots, x_m$, obtaining a vector $\mathbf{x}$, and we have m multi-variate polynomial equations, $x_j = P_j(\mathbf{x})$, which we write together as

$$\mathbf{x} = P(\mathbf{x}) \tag{1}$$

Consider the partial order on m-vectors given by $\mathbf{x} \preceq \mathbf{y}$ if and only if $x_i \leq y_i$ for all i, $1 \leq i \leq m$. The mapping $P : \mathbb{R}^m \mapsto \mathbb{R}^m$ defines a monotone operator on a compact and downward-closed (with respect to $\preceq$) subspace D of $[0, 1]^m$. Let $P^r(x)$ denote $P(\mathbf{x})$ if $r = 1$, and $P(P^{r-1}(\mathbf{x}))$, for $r > 1$. It is clear, by the non-negativity of coefficients of $P(\cdot)$, that $P^r(0) \preceq P^{r+1}(0)$, for $r \geq 1$.

Theorem 1. *(see [EY04] and see [EKM04] for an equivalent result for pPDAs)* $\mathbf{x} = P(\mathbf{x})$ *has a (unique) Least Fixed Point (LFP) solution* $q \in [0, 1]^m$*, given by* $q = \lim_{r\to\infty} P^r(\mathbf{0})$. *I.e.,* $q = P(q)$*, and* $q \preceq v$ *for any solution* v*. Moreover the vector* q *gives precisely the probabilities* $[u, ex]$*, i.e.:* $[u, ex] = q_{[u,ex]}$.

Now, how do we compute this LFP? Well, we can't compute it exactly, and there are several other nasty features to the systems $\mathbf{x} = P(\mathbf{x})$ that distinguish them from the linear systems for finite Markov chains:

Theorem 2. *([EY04])*

1. *Irrational probabilities: There is an RMC for which the probability [en,ex] is irrational, and in fact not "solvable by radicals".*

2. *Slow convergence: There is an RMC for which $|[en,ex] - P^{2^i}_{[en,ex]}(0)| \geq \frac{1}{2^i}$. In other words, we need 2^i applications of the operator P to get within i bits of precision of the LFP.*
3. *Very small & large probabilities: There is a family of hierarchical (i.e., no recursion) RMCs, $A(n)$, parameterized by their size cn, for which $[en,ex] = \frac{1}{2^{2^n}}$ in $A(n)$. And a family, $A'(n)$, of size cn, for which $[en,ex] = 1 - \frac{1}{2^{2^n}}$.*

We can still ask whether a probability is exactly 1, or at least ρ for some rational number ρ, and we can still try to efficiently approximate the probabilities to within a desired number of bits of precision.

RMCs and the Existential Theory of Reals. Given a system $x = P(x)$ associated with an RMC, and a vector $q \in [0,1]^n$, consider the following existential first-order sentence in the theory of reals:

$$\varphi \equiv \exists x_1, \ldots, x_m \bigwedge_{i=1}^{m} P_i(x_1, \ldots, x_m) = x_i \wedge \bigwedge_{i=1}^{m} 0 \leq x_i \wedge \bigwedge_{i=1}^{m} x_i \leq q_i$$

φ holds true precisely when there is some solution $0 \preceq z \preceq q$, with $z = P(z)$. Thus, if we had a way to decide the truth of this sentence, we would be able to tell whether $[u,ex] \leq q_i$, for some rational q_i, by using the vector $q = (1,1,\ldots,q_i,1,\ldots,1)$. Now consider the sentence ψ, obtained from φ by replacing $\bigwedge_{i=1}^{m} x_i \leq q_i$ with $\bigvee_{i=1}^{m} x_i < q_i$. ψ is false precisely when there is no solution $z \succeq 0$, such that $q \npreceq z$. Thus, to decide whether q is the LFP, we need to check the truth of φ and the falsehood of ψ. Furthermore, by a straightforward "binary search", we could use j "queries" to the existential theory of reals to obtain a probability $[u,ex]$ to within j bits of precision (see [EY04]).

Happily, beginning with Tarski, the decidability and complexity of the first-order theory of real and its fragments has been deeply investigated. The current state of the art (see e.g. [Can88, Ren92, BPR96]) provides a PSPACE algorithm that decides whether an existential sentence with rational coefficients is true for the real numbers. The algorithm's running time is exponential only in the number of variables of the sentence. Using these results one can obtain the following:

Theorem 3. *([EY04]) Given RMC A and rational value ρ, there is a PSPACE algorithm to decide whether $[u,ex] \leq \rho$, with running time $O(|A|^{O(1)} \cdot 2^{O(m)})$ where m is the number of variables in the system $x = P(x)$ for A. Moreover $[u,ex]$ can be approximated to within j bits of precision within PSPACE and with running time at most j times the above.*

Single-Exit RMCs and Stochastic Context-Free Grammars. Stochastic Context-Free Grammars (SCFGs) have rules $N \xrightarrow{x} \alpha$, labeled with a probability x, where N is a non-terminal, and α a string of terminals and nonterminals. The probabilities of the rules associated with each non-terminal N must sum to 1.

It can be shown that SCFGs are "equivalent" in a precise sense to *single-exit RMCs* where each component can have only a single exit (see [EY04]). In particular, the probability $[u,ex]$ of the RMC is the same as the probability of termination starting at the corresponding non-terminal $N_{[u,ex]}$ in the corresponding SCFG. They are also equivalent to pPDAs with a single control state, also known as pBPAs: just write $pN \overset{x}{\to} p\alpha$ instead of $N \overset{x}{\to} \alpha$.

SCFGs have been studied extensively since the 1970s in connection with Natural Language Processing (see, e.g., [MS99]), and their theory is intimately connected with that of *multi-type Branching Processes.* Based on results on branching processes (see, e.g., [Har63]), one can "characterize" questions of almost sure termination for SCFGs based on eigenvalues of certain matrices associated with the SCFG (see, e.g., [BT73]). These characterizations unfortunately often omit special uncovered cases, or, worse, contain errors (e.g., the often cited [BT73] contains errors). In [EY04], a detailed treatment of these characterizations is given together with their algorithmic implications, establishing the following:

Theorem 4. *([EY04]) There is polynomial-time algorithm that for a 1-exit RMC A, and every vertex u and exit ex, determines which of the following three cases hold: (1)* $[u, ex] = 0$*, (2)* $[u, ex] = 1$*, or (3)* $0 < [u, ex] < 1$.

RMCs and Newton's Method. Although we can not compute the probabilities associated with an RMC exactly, because as we saw they can be irrational, we can nevertheless aim to efficiently approximate the probabilities numerically within a desired number of bits of precision. Given that the LFP for equation system $\mathbf{x} = P(\mathbf{x})$ is given by $\lim_{r\to\infty} P^r(\mathbf{0})$, and $P^r(0)$ grows monotonically with r, one way to try to do this would be to calculate $P^r(\mathbf{0})$ for a "large enough" r. Unfortunately, as we saw in Theorem 2, there are RMCs for which this approach fails terribly, requiring 2^i iterations to obtain i bits of precision.

A powerful numerical method for obtaining roots of equations is *Newton's method.* In its n-dimensional version (see, e.g., [SB93]), given a suitably differentiable map $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ we wish to find a solution to the system $F(\mathbf{x}) = \mathbf{0}$. Starting at some $\mathbf{x_0} \in \mathbb{R}^n$, the method works by iterating

$$\mathbf{x}_{k+1} := \mathbf{x}_k - (F'(\mathbf{x}_k))^{-1} F(\mathbf{x}_k)$$

where $F'(\mathbf{x})$ is the *Jacobian matrix* of partial derivatives given by

$$F'(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} \cdots \frac{\partial f_1}{\partial x_n} \\ \vdots \vdots \vdots \\ \frac{\partial f_n}{\partial x_1} \cdots \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

The method is not even defined if for some iterate $\mathbf{x_k}$ the matrix $F'(\mathbf{x}_k)$ is not invertible, and when defined it may not converge. In practice, however, if it converges then it typically converges very fast. Remarkably, in [EY04] it is shown that for a decomposed version of the monotone non-linear systems $\mathbf{x} = P(\mathbf{x})$ arising from an RMC, Newton's method started at $\mathbf{x_0} = \mathbf{0}$ not only converges to the LFP, but does so monotonically:

Theorem 5. *([EY04]) Starting at* $\mathbf{x_0} = 0$*, Newton's method converges monotonically to the LFP,* q*, of the system* $\mathbf{x} = P(\mathbf{x})$ *(appropriately decomposed) of an RMC. In other words,* $\lim_{k\to\infty} \mathbf{x}_k = q$*, and* $\mathbf{x}_k \preceq \mathbf{x}_{k+1}$*, for all* $k \geq 0$*.*

Moreover, from the proof it follows that, for all $k \geq 0$, $\mathbf{x}_k \geq P^k(0)$, and that Newton's method corresponds to a clever "acceleration" of the standard iteration $P^k(\cdot)$ which will typically be much faster than iterating $P^k(\cdot)$. In particular, on the examples known to require exponentially many iterations of $P(\cdot)$ to achieve a given number of bits of precision, Newton's method converging in only a linear number of iterations (see [EY04] for an expanded explanation of these remarks).

Lower Bounds for Reachability. We have seen that basic questions about reachability probabilities can be answered in PSPACE by using the existential theory of reals, and that for the special case of single-exit RMCs (SCFGs), the qualitative reachability problem, whether $[u, ex] = 1$, can be answered in polynomial time. Can we provide any lower bounds for the remaining questions? Hardness for standard complexity classes, such as NP or PSPACE, remains open. However, we have the following strong evidence of "difficulty". The *square-root sum problem* is the following decision problem: given $(d_1, \ldots, d_n) \in \mathbb{N}^n$ and $k \in \mathbb{N}$, decide whether $\sum_{i=1}^{n} \sqrt{d_i} \leq k$. It is known to be solvable in PSPACE, but it has been a major open problem in the complexity of numerical computation since the 1970's (see, e.g., [GGJ76, Tiw92]) whether it is solvable even in NP, with important consequences in subjects like computational geometry.

Theorem 6. *([EY04]) The square-root sum problem is polynomial-time reducible to the problem of determining, given a single-exit RMC, a vertex* u *and exit* ex*, and a rational value* r*, whether* $[u, ex] \geq r$*.*

A simple modification of this reduction shows that the square-root sum problem is polynomial-time reducible to problem of determining, given a 2-exit RMC, a vertex u and exit ex, whether $[u, ex] = 1$.

4 Repeated Reachability

Let a *run* of a Markov chain be either an infinite path or a finite path ending at a halting state without successors. Given an initial state s_0 and a set of states S, we are interested in the probability that the runs starting at s_0 *repeatedly visit* states of S, i.e., that they visit S infinitely often. (For a formal definition of this probability and a proof that it exists, see for instance [Var85, EKM04].) If the case of flat-programs, both the qualitative and the quantitative repeated reachability problems can be solved by slight modifications of the algorithms for the reachability problems, with the same complexity. Let us now consider procedural programs. For convenience, we model the program as a pushdown automaton with an initial configuration $c_0 = \langle p_0, X_0 \rangle$. To simplify the presentation we assume that the set of configurations that should be repeatedly visited, denoted by C_r, is the set of configurations with head $p_r X_r$ for some control state p_r and some stack symbol X_r (see [EKM04] for a more general case).

We define a new *finite* Markov chain MH such that the repeated reachability problem for c_0 and C_r can be reduced to a repeated reachability problem for MH, which we already know how to solve. The key notion we need are the *minima* of an infinite run, defined inductively as follows. The first minimum of an infinite run $c_0 \xrightarrow{x_1} c_1 \xrightarrow{x2} \ldots$, where $c_i = \langle p_i, \alpha_i \rangle$, is the smallest index j such that $|\alpha_k| \geq |\alpha_j|$ for every $k \geq j$. For every $i > 1$, if j is the i-th minimum of the run, then the $(i+1)$-th minimum is the first minimum of the suffix $c_{j+1} \xrightarrow{x_{j+2}} c_{j+2} \ldots$. In words, the first minimum is the index of the first configuration having minimal stack length and the $(i+1)$-th minimum is obtained by chopping off the prefix of the run up to the i-minimum, and taking the first minimum of the rest. Now, what is the probability that the $(i+1)$-th minimum has head qY, if the i-th minimum has head pX? It is proved in [EKM04] that this probability depends only on pX and qY. Intuitively, if $\langle p, X\alpha \rangle$ is the configuration at the i-th minimum, all its successor configurations in the run have α at the bottom of the stack. So α plays no rôle in determining the head of the next minimum, because from $\langle p, X\alpha \rangle$ onward all stack operations "happen above α".

This result allows us to define a Markov chain whose states are the heads of the pPDA plus two special states *Init* and *Ter*, and whose transitions are as follows, where $\mathit{PMin}(pX, qY)$ denotes the probability that a minimum has head pY assuming that the previous minimum has head pX:

- $\mathit{Init} \xrightarrow{x} \mathit{Ter}$, where x is the probability that a run starting at c_0 terminates, i.e., reaches a configuration of the form $\langle p, \epsilon \rangle$;
- $\mathit{Ter} \xrightarrow{1} \mathit{Ter}$;
- $\mathit{Init} \xrightarrow{x} pX$ for every head pX such that $x = \mathit{PMin}(p_0X_0, pX) > 0$; and
- $pX \xrightarrow{x} qY$ for every two heads pX and qY such that $x = \mathit{PMin}(pX, qY) > 0$.

How can we decide if $\mathit{PMin}(pX, qY) > 0$? Using the results of the previous section, we can compute for every p, q, X the probability $[pXq]$ of reaching $\langle q, \epsilon \rangle$ from $\langle p, X \rangle$ (these are essentially the probabilities $[u, ex]$ of the previous section), and the probability $[pX]{\uparrow}$ of never emptying the stack from $\langle p, X \rangle$ (i.e., of never reaching a configuration of the form $\langle q, \epsilon \rangle$ for any control state q). Consider now a run starting at $\langle p, X \rangle$. In order to reach the next minimum at $\langle q, Y\beta \rangle$ for some β, the pPDA has the following possibilities:

- Apply the rule $pX \xrightarrow{x} qY$, if it exists, and then, from $\langle q, Y \rangle$, never empty the stack.
- Apply a rule $pX \xrightarrow{x} qYZ$ for some Z, and then keep Z forever at the bottom of the stack.
- Apply a rule $pX \xrightarrow{x} rZY$ for some r, Z, from $\langle r, ZY \rangle$ reach the configuration $\langle q, Y \rangle$, and then never empty the stack.

It is easy to compute the probability of each case. Adding them we obtain:

$$\mathit{PMin}(pX, qY) = \sum_{pX \xrightarrow{x} qY} x \cdot [qY]{\uparrow} + \sum_{pX \xrightarrow{x} qYZ} x \cdot [qY]{\uparrow} + \sum_{pX \xrightarrow{x} rZY} x \cdot [rZq] \cdot [qY]{\uparrow}$$

Since $[pX]{\uparrow} + \sum_{q \in Q}[pXq] = 1$, where Q is the set of control states of the pPDA, deciding if $PMin(pX, qY) > 0$ reduces to deciding if $[qY]{\uparrow} > 0$ for each head qY. By the results of the previous section, this can be done in PSPACE, and in PTIME for pBPAs or 1-exit RMCs.

Using this finite chain we can decide if a run repeatedly visits configurations of C_r *at minima* with probability 1 (at least ρ). But, what happens if the configurations of C_r occur *between* minima? To solve this problem, we split each state pX into $(pX, 0)$ and $(pX, 1)$, and assign transition probabilities as follows. For a transition $(pX, f) \xrightarrow{x_1} (qY, 1)$, where $f \in \{0, 1\}$, we set x_1 to the probability that a run starting at $\langle p, X\rangle$ hits the second minimum ($\langle p, X\rangle$ itself is the first) at a configuration with head qY *and visits some configuration of C_r in-between.* For a transition $(pX, f) \xrightarrow{x_0} (qY, 0)$ we set $x_0 = PMin(pX, qY) - x_1$. The Markov chain MH mentioned at the beginning of the section is the result of performing this modification. A run of the pPDA repeatedly visits configurations of C_r if and only if it corresponds to a run of MH that repeatedly visits states of the form $(pX, 1)$. In order to solve the qualitative repeated reachability problem, we construct MH and then apply the algorithm for the finite state case. Notice, however, that we do not need the exact values of the transition probabilities of MH, we only have to decide if they are positive. This yields a PSPACE-algorithm for the qualitative repeated reachability problem, and a PTIME-algorithm for pBPA or 1-exit RMC, the same status as for reachability. A lower bound for the general case is open, but the remarks in section 3 on lower bounds for reachability apply also to repeated reachability.

For the quantitative repeated reachability problem we need to solve a linear system of equations whose coefficients are the probabilities of the transitions of MH. Complexity questions have not been studied in detail yet.

5 Model Checking PCTL

The syntax of PCTL, the probabilistic extension of CTL proposed in [HJ94] is given by:

$$\varphi ::= \texttt{tt} \mid A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathcal{X}^{\geq\rho}\varphi \mid \varphi_1 \mathcal{U}^{\geq\rho}\varphi_2$$

where A is an atomic proposition, ρ is a probability, and $\mathcal{X}$ and $\mathcal{U}$ are the next and until operators of LTL. Formulas with operators $\leq, =, <, >$ can be 'simulated' by boolean combinations. A state of a Markov chain satisfies $\mathcal{X}^{\geq\rho}\varphi$ or $\varphi_1 \mathcal{U}^{\geq\rho}\varphi_2$ if the probability that a run starting at it satisfies $\mathcal{X}\varphi$ or $\varphi_1 \mathcal{U} \varphi_2$, respectively, is at least ρ. The *qualitative fragment* of PCTL is obtained by requiring $\rho \in \{0, 1\}$.

Given a Markov chain M and a PCTL formula φ, let $[\![\varphi]\!]$ denote the set of states of M satisfying φ. As in the case of CTL, the key to a model-checking algorithm for PCTL consists of, given $[\![\varphi_1]\!], [\![\varphi_2]\!]$, computing $[\![\varphi_1 \mathcal{U}^{\geq\rho}\varphi_2]\!]$. In the case of flat-programs, $[\![\phi]\!]$ is computed bottom-up, i.e., computing first $[\![\phi']\!]$ for all subformulas ϕ' of ϕ. This can be done using well-known graph algorithms if $\rho \in \{0, 1\}$, and solving linear systems of equations otherwise [HJ94].

In the procedural case, we face an obstacle: Since the Markov chain is infinite, the set $[\![\, \varphi \,]\!]$ may be infinite, and cannot be computed by explicit enumeration of its elements. Let us see the implications of this.

A valuation is *regular* if $[\![\, A \,]\!]$ is a regular set for every atomic proposition A, where 'regular' is used in the language-theoretic sense: A configuration $\langle p, \alpha \rangle$ is seen as the word $p\alpha$. It is shown in [EKM04] that if a valuation is effectively regular, then $[\![\, \varphi \,]\!]$ is effectively regular for every PCTL formula φ. This provides a solution to the infinity problem: Compute a finite automaton recognizing $[\![\, \varphi \,]\!]$.

We sketch the proof of this regularity result for a particular case. We show that $[\![\, \varphi_1 \mathcal{U}^{\geq 1} \varphi_2 \,]\!]$ is regular if $[\![\, \varphi_1 \,]\!]$ is the set of all configurations, and $[\![\, \varphi_2 \,]\!] = \{\langle q, \epsilon \rangle\}$ for some given control state q. Let a head pX be *almost surely terminating* (a.s.t.) if a run starting at $\langle p, X \rangle$ empties the stack with probability 1. Given an a.s.t. pX, let $\mathit{Emp}(pX)$ be the set of states r such that the probability of reaching $\langle r, \epsilon \rangle$ from $\langle p, X \rangle$ is non-zero. Then $[\![\, \varphi_1 \mathcal{U}^{\geq 1} \varphi_2 \,]\!]$ is the least set C containing $\langle q, \epsilon \rangle$ and satisfying: If pX is a.s.t. and $\langle r, \alpha \rangle \in C$ for every $r \in \mathit{Emp}(pX)$, then $\langle p, X\alpha \rangle \in C$. Consider now the automaton having the set of stack symbols as alphabet, all subsets of control states as states, all singletons $\{p\}$ as initial states, the set $\{q\}$ as final state, and a transition $P_1 \xrightarrow{X} P_2$ if and only if the head pX is a.s.t. for every $p \in P_1$, and $P_2 = \bigcup_{p \in P_1} \mathit{Emp}(pX)$. This automaton accepts $\alpha \in \Gamma^*$ from the state p if and only if $\langle p, \alpha \rangle \in C$, and so $[\![\, \varphi_1 \mathcal{U}^{\geq 1} \varphi_2 \,]\!]$ is regular.

The exact complexity of the model checking problem for pPDAs and the qualitative fragment of PCTL with regular valuations is still open. Using results of [Wal00] it is easy to show that the problem is EXPTIME-hard, even for pPBAs or 1-exit RMCs [May04]. We also know that the problem can be solved in triple exponential time [Kuč04].

If φ does not belong to the qualitative fragment, the set $[\![\, \varphi \,]\!]$ may not be regular, even for a regular valuation. Consider the pPDA

$$\begin{array}{llll}
pX \xrightarrow{1/2} qX & qX \xrightarrow{1/2} q\epsilon & rX \xrightarrow{1} r\epsilon & sX \xrightarrow{1} sX \\
pX \xrightarrow{1/2} rX & qX \xrightarrow{1/2} sX & rY \xrightarrow{1/2} r\epsilon & sY \xrightarrow{1} sY \\
 & qY \xrightarrow{1} q\epsilon & rY \xrightarrow{1/2} qY &
\end{array}$$

and atomic propositions A_1, A_2 together with the regular valuation in which $[\![\, A_1 \,]\!]$ is the set of all configurations, and $[\![\, A_2 \,]\!] = \{\langle q, \epsilon \rangle\}$. It is easy to see that

$$\{\langle p, X^n Y^m \rangle \mid n, m > 0\} \cap [\![\, A_1 \mathcal{U}^{=1/2} A_2 \,]\!] = \{\langle p, X^n Y^n \rangle \mid n > 0\}$$

which, since $\{\langle p, X^n Y^m \rangle \mid n, m \geq 0\}$ is regular and $\{\langle p, X^n Y^n \rangle \mid n \geq 0\}$ is not, implies that $[\![\, A_1 \mathcal{U}^{=1/2} A_2 \,]\!]$ is not regular. In [BKS], the pPDA above is used as a building block in a reduction from the halting problem for 2-counter machines to the model checking problem for pPDA's and PCTL, which shows that the latter is undecidable.

6 Model Checking Büchi Automata Specifications

Let M be a Markov chain modelling a program. We formalize the specification as a Büchi automaton $\mathcal{B}$. A word accepted by $\mathcal{B}$ is seen as a 'good behaviour' of the program. (Recall that $\mathcal{B}$ accepts a word $a_1a_2\ldots$ if it has a run $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots$ and an accepting state q that the run visits infinitely often.) The verification problem is to decide if a run of M is accepted by $\mathcal{B}$ (i.e., is 'a good behaviour') with probability 1, or with probability at least ρ for a given $\rho \in [0,1]$.

For flat-programs, the alphabet of $\mathcal{B}$ is the set of states of M, which is finite. For procedural programs, we take as alphabet the set of heads of $\mathcal{P}$. This means that specifications can refer to the control points and variables of the program, but not to the stack of activation records (see [BKS] for a generalization).

The verification problem for flat-programs is solved (in two ways) in [Var85, CY95]. For the procedural case, assume first that $\mathcal{B}$ is deterministic, as done in [EKM04]. We construct the pPDA $\mathcal{P} \times \mathcal{B}$ having pairs (p,b) as states, where p is a control state of $\mathcal{P}$ and b is a state of $\mathcal{B}$, and rules $(p,b)X \xrightarrow{x} (p',b')\alpha$, where $pX \xrightarrow{x} p'\alpha$ is a rule of $\mathcal{P}$ and $q \xrightarrow{pX} q'$ is a transition of $\mathcal{B}$. We construct the Markov chain MH having states of the form $((p,b)X, f)$, where $f = 1$ denotes that some configuration $\langle (q,b'), \alpha \rangle$ with b' accepting has been visited since the last minimum. A run of $\mathcal{P}$ is accepted by $\mathcal{B}$ with probability 1 (at least ρ) if and only if a run of MH repeatedly visits states satisfying $f = 1$ with probability 1 (at least ρ). So the verification problem reduces to the repeated reachability problem.

The nondeterministic case was left open in [EKM04]. The following solution is from [BKS]. In a first step, $\mathcal{B}$ is transformed into a *deterministic Muller automaton* $\mathcal{B}'$ with acceptance sets $Q_1, \ldots, Q_n$. (Recall that $\mathcal{B}'$ accepts a word $a_1a_2\ldots$ if it has a run $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots$ and an acceptance set Q_i such that the set of states visited by the run infinitely often is exactly Q_i.) The product $\mathcal{P} \times \mathcal{B}'$ is defined as above. However, we redefine the states of the Markov chain MH so that they not only reflect whether some accepting state was visited since the last minimum, but also *which states* of $\mathcal{B}'$ were visited. More formally, we replace the boolean f by a set of states of $\mathcal{B}'$, and in a transition $((p_1,b_1)X_1, S_1) \xrightarrow{x} ((p_2,b_2)X_2, S_2)$ we set x to the probability of, starting at $\langle (p_1,b_1), X_1 \rangle$, hitting the next minimum at a configuration with head $(p_2,b_2)X_2$, and visiting exactly the states of S_2 in-between. With this definition of MH, the runs of $\mathcal{P}$ are accepted by $\mathcal{B}'$ with probability 1 if and only if every bottom strongly connected component of MH satisfies the following property: if the states of the component are $((p_1,b_1)X_1, S_1), \ldots ((p_n,b_n)X_n, S_n)$, then $S_1 \cup \ldots \cup S_n$ is an acceptance set of $\mathcal{B}'$. While this shows that the problem of checking Büchi automata specifications is decidable, the exact complexity of the problem is open.

Acknowledgments. This survey is based on joint work by the first author together with Tomáš Brázdil, Antonín Kučera, Richard Mayr and Oldřich Stražovský [BKS, EKM04], and on joint work by the second author together with Mihalis Yannakakis [EY04]. We would both like to acknowledge and thank our collaborators.

References

[AEY01] R. Alur, K. Etessami, and M. Yannakakis. Analysis of recursive state machines. In *Proceedings of CAV'01*, volume 2102 of *LNCS*, pages 304–313, 2001.

[BEM97] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Applications to model checking. In *Proceedings of CONCUR'97*, volume 1243 of *LNCS*, pages 135–150, 1997.

[BGR01] M. Benedikt, P. Godefroid, and T. Reps. Model checking of unrestricted hierarchical state machines. In *Proceedings of ICALP'01*, volume 2076 of *LNCS*, pages 652–666, 2001.

[BJMT01] F. Besson, T. Jensen, D.L. Métayer, and T. Thorn. Model checking security properties of control flow graphs. *Journal of Computer Security*, 9:217–250, 2001.

[BKS] T. Brázdil, A. Kučera, and O. Stražovský. Decidability of temporal properties of probabilistic pushdown automata. Technical report. In preparation.

[BPR96] S. Basu, R. Pollack, and M. F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM*, 43(6):1002–1045, 1996.

[BT73] T. L. Booth and R. A. Thompson. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, 22(5):442–450, 1973.

[Can88] J. Canny. Some algebraic and geometric computations in pspace. In *Proceedings of 20th ACM STOC*, pages 460–467, 1988.

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

[EHRS00] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of CAV'00*, volume 1855 of *LNCS*, pages 232–247, 2000.

[EKM04] J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. In *Proceedings of LICS'04*, pages 12–21. IEEE Computer Society, 2004. Full version: Tech. report FIMU-RS-2004-03, Masaryk University, Brno, available online at http://www.fmi.uni-stuttgart.de/szs/publications/info/esparza.EKM04rep.shtml.

[EY04] K. Etessami and M. Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of non-linear equations. Technical report, 2004. School of Informatics, University of Edinburgh.

[GGJ76] M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *Proceedings of 8th ACM STOC*, pages 10–22, 1976.

[Har63] T. E. Harris. *The Theory of Branching Processes.* Springer-Verlag, 1963.

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

[Kuč04] A. Kučera. Private communication, 2004.

[Kwi03] M. Kwiatkowska. Model checking for probability and time: From theory to practice. In *Proceedings of LICS'03*, pages 351–360. IEEE Computer Society Press, 2003.

[May04] R. Mayr. Private communication, 2004.

[MS99] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

[Ren92] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Parts I,II, III. *Journal of Symbolic Computation*, pages 255–352, 1992.

[SB93] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, 1993.

[Tiw92] P. Tiwari. A problem that is easier to solve on the unit-cost algebraic RAM. *Journal of Complexity*, pages 393–397, 1992.

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

[Wal00] I. Walukiewicz. Model checking CTL properties of pushdown systems. In *Proceedings of FST&TCS'00*, volume 1974 of *Lecture Notes in Computer Science*, pages 127–138. Springer, 2000.