Recursive Markov Chains, Stochastic Grammars, and Monotone Systems of Nonlinear Equations

Kousha Etessami[1] and Mihalis Yannakakis[2]

[1] School of Informatics, University of Edinburgh
[2] Department of Computer Science, Columbia University

Abstract. We introduce and study Recursive Markov Chains (RMCs), which extend ordinary finite state Markov chains with the ability to invoke other Markov chains in a potentially recursive manner. They offer a natural abstract model for probabilistic programs with procedures, and are a probabilistic version of Recursive State Machines. RMCs generalize Stochastic Context-Free Grammars (SCFG) and multi-type Branching Processes, and are intimately related to Probabilistic Pushdown Systems. We focus here on termination and reachability analysis for RMCs. We present both positive and negative results for the general class of RMCs, as well as for important subclasses including SCFGs.

1 Introduction

We introduce and study *Recursive Markov Chains* (RMCs), a natural model for systems involving probability and recursion. Informally, a Recursive Markov Chain consists of a collection of finite state component Markov chains (MC) which can call each other in a potentially recursive manner. RMCs are a natural probabilistic version of Recursive State Machines (RSMs) ([1, 4]), with transitions labeled by probabilities. RSMs and closely related models like Pushdown Systems (PDSs) have been studied extensively in recent research on model checking and program analysis, because of their applications to verification of sequential programs with procedures. RMCs offer a natural abstract model for probabilistic procedural programs. Probabilistic models of programs are of interest for at least two reasons. First, the program may use randomization, in which case the transition probabilities reflect the random choices of the algorithm. Second, we may want to model and analyse a program under statistical conditions on its behaviour (e.g., based on profiling statistics or on statistical assumptions), and to determine the probability of properties of interest, e.g., that the program terminates, and/or that it terminates in a certain state.

Recursive Markov chains are an interesting and important model in their own right. RMCs provide a succinct finite representation for a natural class of denumerable Markov Chains that generalize other well-studied models, such as Stochastic Context-Free Grammars (SCFGs) and Multitype Branching Processes, and they are intimately related to Probabilistic Pushdown Systems (pPDSs).

Single-exit RMCs, the special case where each component MC has exactly 1 exit (terminating state), are in a precise sense "equivalent" to SCFGs. SCFGs

have been studied extensively, especially in the Natural Language Processing (NLP) community, since the 1970s (see, e.g., [23]). Their theory is directly connected with the theory of *multi-type Branching Processes* (MT-BPs) initiated by Kolmogorov and others (see, e.g., [18, 20]). BPs are an important class of stochastic processes. Their theory dates back to the 19th century and the work of Galton and Watson on population dynamics. Multi-type BPs and SCFGs have been applied in a wide variety of stochastic contexts besides NLP, including population genetics ([19]), models in molecular biology including for RNA ([27]), and the study of nuclear chain reactions ([13]). Many variants and extensions of MT-BPs have also been studied. Despite this extensive study, some basic algorithmic questions about SCFGs and MT-BPs have not been satisfactorily answered. For example, is the probability of termination of a given SCFG (i.e., the probability of its language) or MT-BP (i.e. the so-called probability of extinction) $\geq p$? Is it $= 1$? Can these questions be decided in polynomial time in general? What if there are only a constant number of types in the branching process (non-terminals in the grammar)? RMCs form a natural generalization of SCFGs and MT-BPs, however their underlying stochastic processes appear not to have been studied in their own right in the rich branching process literature.

Our goal in this paper is to provide efficient algorithms and to determine the computational complexity of reachability analysis for RMCs. Namely, we are interested in finding the probability of eventually reaching a given terminating vertex of the RMC starting from a given initial vertex. As with ordinary Markov chains, such algorithms are a core building block for model checking and other analyses of these probabilistic systems. For SCFGs (MT-BPs), this amounts to an algorithm for determining the probability of termination (extinction).

It turns out reachability probabilities for RMCs are captured by the Least Fixed Point (LFP) solution of certain monotone systems of nonlinear polynomial equations. We observe that these solutions can be irrational even for SCFGs, and not solvable by radicals. Thus we can't hope to compute them exactly.

By appealing to the deep results on the complexity of decision procedures for the Existential Theory of Reals (see, e.g., [6]), we show that for general RMCs we can decide in PSPACE whether the probability is $\leq p$, or $= p$, for some rational $p \in [0, 1]$, and we can approximate the probabilities to within any given number of bits of precision. For an SCFG where the number of non-terminals is bounded by a constant (or a MT-BP with a bounded number of types), we can answer these questions in polynomial time. We show that this holds more generally, for RMCs where the total number of entries and exits of all components is bounded by a constant. Furthermore, we show that for single-exit RMCs with an arbitrary number of components (i.e., for general SCFGs), we can decide if the probability is *exactly* 1 in P-time.

The monotone nonlinear systems for RMCs give rise to a natural iterative numerical algorithm with which to approximate the LFP. Namely, the system of equations has the form $\mathbf{x} = P(\mathbf{x})$, where $\mathbf{x}$ is a vector, such that the vector of probabilities we are seeking is given by $\lim_{k\to\infty} P^k(\mathbf{0})$, where $P^1(\mathbf{0}) = P(\mathbf{0})$ and $P^{k+1}(\mathbf{0}) = P(P^k(\mathbf{0}))$. We show that this iteration can be very slow to converge.

Remarkably however, we show that a multi-dimensional Newton's method converges monotonically to the LFP on a decomposed version of the system, and constitutes a rapid "acceleration" of the standard iteration. Note that in other contexts, in general Newton's method is not guaranteed to converge; but when it does converge, typically it converges very fast. We thus believe that in our context Newton provides an efficient practical method for numerically estimating these probabilities for all RMCs.

Lastly, for "lower bounds", we show that one can not hope to improve our PSPACE upper bounds for RMCs without a major breakthrough, by establishing a connection to a fundamental open problem in the complexity of numerical computation: the square-root sum problem. This problem is known to be in PSPACE, but its containment even in NP is a longstanding open problem first posed in 1976 ([16]), with applications to a number of areas including computational geometry. We show the square-root sum problem is polynomial-time reducible to the problem of determining for an SCFG whether the termination probability is $\leq p$ for some $p \in [0, 1]$, and also to the problem of determining whether a 2-exit RMC terminates with probability 1.

Due to space limitations, all proofs are omitted. Please see the full paper [15].

Related Work. The work in the verification literature on algorithmic analysis of pushdown systems is extensive (see, e.g., [3, 10]). Recursive state machines were introduced in [1, 4] as a more direct graphical model of procedural programs, and their algorithmic verification questions were thoroughly investigated. A recent work directly related to ours is by Esparza, Kucera, and Mayr [12]. They consider model checking for probabilistic pushdown systems (pPDSs). pPDSs and RMCs are intimately related models, and there are efficient P-time translations from one to the other. As part of their results [12] show decidability of reachability questions for pPDSs by appealing to results on the theory of reals. In particular, they derive EXPTIME upper bounds for reachability. Our work was done independently and concurrently with theirs (a preliminary draft of our work was made available to the authors of [12] after we learned of their work). In any case, although their work overlaps briefly with ours, their primary focus is on decidability (rather than precise complexity) of model checking problems for a probabilistic branching-time temporal logic, PCTL. Our work also answers several complexity questions raised in their work. As mentioned earlier, SCFGs have been studied extensively in the NLP literature (see, e.g., [23]). In particular, the problem of *consistency* of a SCFG (whether it terminates with probability 1) has been studied, and its connection to the extinction problem for MT-BPs is well known [18, 7, 17, 9]. However, none of the relevant references provide a complete algorithm and characterization for consistency. Another work on pPDSs is [2]. They do not address algorithmic questions for reachability.

2 Basics

A *Recursive Markov Chain (RMC)*, A, is a tuple $A = (A_1, \ldots, A_k)$, where each *component graph* $A_i = (N_i, B_i, Y_i, En_i, Ex_i, \delta_i)$ consists of:

- A set N_i of *nodes*.
- A subset of *entry* nodes $En_i \subseteq N_i$, and a subset of *exit* nodes $Ex_i \subseteq N_i$.
- A set B_i of *boxes*. Let $B = \cup_{i=1}^{k} B_i$ be the (disjoint) union of all boxes of A.
- A mapping $Y_i : B_i \mapsto \{1, \ldots, k\}$ that assigns to every box (the index of) one of the components, $A_1, \ldots, A_k$. Let $Y = \cup_{i=1}^{k} Y_i$ denote the map $Y : B \mapsto \{1, \ldots, k\}$ which is consistent with each Y_i, i.e., $Y|_{B_i} = Y_i$, for $1 \leq i \leq k$.
- To each box $b \in B_i$, we associate a set of *call ports*, $Call_b = \{(b, en) \mid en \in En_{Y(b)}\}$, and a set of *return ports*, $Return_b = \{(b, ex) \mid ex \in Ex_{Y(b)}\}$.
- A transition relation δ_i, where transitions are of the form $(u, p_{u,v}, v)$ where:
  1. the source u is either a non-exit node $u \in N_i \setminus Ex_i$, or a return port $u = (b, ex) \in Return_b$, where $b \in B_i$.
  2. The destination v is either a non-entry node $v \in N_i \setminus En_i$, or a call port $v = (b, en) \in Call_b$, where $b \in B_i$.
  3. $p_{u,v} \in \mathbb{R}_{>0}$ is the transition probability from u to v. (We assume $p_{u,v}$ is rational.)
  4. *Consistency of probabilities*: for each u, $\sum_{\{v' | (u, p_{u,v'}, v') \in \delta_i\}} p_{u,v'} = 1$, unless u is a call port or exit node, neither of which have outgoing transitions, in which case by default $\sum_{v'} p_{u,v'} = 0$.

We will use the term *vertex* of A_i to refer collectively to its set of nodes, call ports, and return ports, and we denote this set by Q_i, and we let $Q = \bigcup_{i=1}^{k} Q_i$ be the set of all vertices of the RMC A. That is, the transition relation δ_i is a set of probability-weighted directed edges on the set Q_i of vertices of A_i. Let $\delta = \cup_i \delta_i$ be the set of all transitions of A.

An RMC A defines a global denumerable Markov chain $M_A = (V, \Delta)$ as follows. The global *states* $V \subseteq B^* \times Q$ of M_A are pairs of the form $\langle \beta, u \rangle$, where $\beta \in B^*$ is a (possibly empty) sequence of boxes and $u \in Q$ is a *vertex* of A. More precisely, the states V and transitions Δ are defined inductively as follows:

1. $\langle \epsilon, u \rangle \in V$, for $u \in Q$. (ϵ denotes the empty string.)
2. if $\langle \beta, u \rangle \in V$ and $(u, p_{u,v}, v) \in \delta$, then $\langle \beta, v \rangle \in V$ and $(\langle \beta, u \rangle, p_{u,v}, \langle \beta, v \rangle) \in \Delta$
3. if $\langle \beta, (b, en) \rangle \in V$, $(b, en) \in Call_b$, then $\langle \beta b, en \rangle \in V$ and $(\langle \beta, (b, en) \rangle, 1, \langle \beta b, en \rangle) \in \Delta$
4. if $\langle \beta b, ex \rangle \in V$, $(b, ex) \in Return_b$, then $\langle \beta, (b, ex) \rangle \in V$ and $(\langle \beta b, ex \rangle, 1, \langle \beta, (b, ex) \rangle) \in \Delta$

Item 1 corresponds to the possible initial states, 2 corresponds to a transition within a component, 3 is when a new component is entered via a box, 4 is when the process exits a component and control returns to the calling component.

Some states of M_A are *terminating states* and have no outgoing transitions. These are states $\langle \epsilon, ex \rangle$, where ex is an exit node. If we wish to view M_A as a proper Markov chain, we can consider the terminating states as absorbing states of M_A, with a self-loop of probability 1.

RMCs where the *call graph* between components forms an acyclic graph are called *Hierarchical Markov Chains* (HMCs). In this special case M_A is finite, but can be exponentially larger than the HMC which specifies it.

The central reachability questions. Our goal is to answer termination and reachability questions for RMCs. Given a vertex $u \in Q_i$ and an exit $ex \in Ex_i$,

both in the same component A_i, let $q^*_{(u,ex)}$ denote the probability of eventually reaching the terminating state $\langle \epsilon, ex \rangle$, starting at the initial state $\langle \epsilon, u \rangle$. Computing probabilities $q^*_{(u,ex)}$ will allow us to efficiently obtain other probabilities.

For a given pair of vertices $u, v \in Q$ of the RMC, let $[u, v]$ denote the probability that starting at state $\langle \epsilon, u \rangle$ we will eventually reach a state $\langle \beta, v \rangle$ for some $\beta \in B^*$. We can obtain the probabilities $[u, v]$ based on the probabilities $q^*_{(u,ex)}$. One way to do this is as follows: add a new special exit $ex^\star_i$ to every component A_i of the RMC, remove the out-edges from $v \in Q_j$ and instead add a transition $v \xrightarrow{1} ex^\star_j$, and add transitions $w \xrightarrow{1} ex^\star_h$, for every return port $w = (b, ex^\star_k)$, where $b \in B_h$. Now, for $u \in Q_i$, $[u, v]$ in the original RMC is equal to $q^*_{(u,ex^\star_i)}$ in the revised RMC. (Intuitively, when we encounter v we "raise an exception", pop the entire call stack, and exit the system.) There is also a more involved way to obtain the probability $[u, v]$ from the probabilities $q^*_{(u,ex)}$ without increasing the number of exits in any component. We can thus focus on finding efficient algorithms for the following central questions:

(1) *Qualitative* reachability problem: Is $\mathbf{q}^*_{(u,ex)} = 1$?
(2) *Quantitative* reachability problems: Given $r \in [0, 1]$, is $q^*_{(u,ex)} \geq r$? Is $q^*_{(u,ex)} = r$? Compute or approximate the exact probabilities $q^*_{(u,ex)}$.

Single-exit RMCs and Stochastic Context-Free Grammars. A *Stochastic Context-Free Grammars* (SCFG) is a tuple $G = (T, V, R, S_1)$, where T is a set of *terminal* symbols, $V = \{S_1, \ldots, S_k\}$ is a set of *non-terminals*, and R is a set of rules $S_i \xrightarrow{p} \alpha$, where $S_i \in V$, $p \in [0, 1]$, and $\alpha \in (V \cup T)^*$, such that for every non-terminal S_i, $\sum_{\langle p_j | (S_i \xrightarrow{p_j} \alpha_j) \in R \rangle} p_j = 1$. Let $p(S_j)$ denote the probability that the grammar, started at S_j, will terminate and produce a finite string. SCFGs are "equivalent" to single-exit RMCs in the following sense.

Proposition 1. *Every SCFG G can be transformed to a 1-exit RMC A, such that $|A| \in O(|G|)$, and there is a bijection from non-terminals S_j in G to components A_j of A, each with a single entry en_j and exit ex_j, such that $p(S_j) = \mathbf{q}^*_{(en_j, ex_j)}$, for all j. Conversely, every 1-exit RMC A can be transformed to a SCFG G of size $O(|A|)$, such that there is a map from vertices u to non-terminals S_u of G, such that if ex is u's component exit, then $\mathbf{q}^*_{(u,ex)} = p(S_u)$.*

The system of nonlinear equations associated with an RMC. Consider the probabilities $q^*_{(u,ex)}$ as unknowns. We can set up a system of (nonlinear) polynomial equations, such that these probabilities must be a solution of the system, and in fact precisely the *Least Fixed Point* solution (which we define). Let us use a variable $x_{(u,ex)}$ for each unknown probability $q^*_{(u,ex)}$. We will often find it convenient to index the variables $x_{(u,ex)}$ according to a fixed order, so we can refer to them also as $x_1, \ldots, x_n$, with each $x_{(u,ex)}$ identified with x_j for some j. We thus obtain a vector of variables: $\mathbf{x} = (x_1\ x_2 \ldots x_n)^T$.

Definition 1. *Given RMC $A = (A_1, \ldots, A_k)$, we define a system of polynomial equations, S_A, over the variables $x_{(u,ex)}$, where $u \in Q_i$ and $ex \in Ex_i$, for $1 \leq i \leq$*

k. The system contains one equation of the form $x_{(u,ex)} = P_{(u,ex)}(\mathbf{x})$*, for each variable* $x_{(u,ex)}$*. Here* $P_{(u,ex)}(\mathbf{x})$ *denotes a multivariate polynomial with positive rational coefficients. There are 3 cases to distinguish, based on the "type" of vertex* u*:*

1. *Type I:* $u = ex$*. In this case:* $x_{(ex,ex)} = 1$.
2. *Type II: either* $u \in N_i \setminus \{ex\}$ *or* $u = (b, ex')$ *is a return port. In these cases:*
$$x_{(u,ex)} = \sum_{\{v|(u,p_{u,v},v)\in\delta\}} p_{u,v} \cdot x_{(v,ex)}.$$
(If u *has no outgoing transitions, this equation is by definition* $x_{(u,ex)} = 0$*.)*
3. *Type III:* $u = (b, en)$ *is a call port. In this case:*
$$x_{((b,en),ex)} = \sum_{ex' \in Ex_{Y(b)}} x_{(en,ex')} \cdot x_{((b,ex'),ex)}$$

In vector notation, we denote $S_A = (x_j = P_j(\mathbf{x}) \mid j = 1, \ldots, n)$ *by:* $\mathbf{x} = P(\mathbf{x})$.

Note we can easily construct the system $\mathbf{x} = P(\mathbf{x})$ from A in polynomial time: $P(\mathbf{x})$ has size $O(|A||Ex|^2)$, where $|Ex|$ denotes the maximum number of exits of any component of A. We will now identify a particular solution to the systems $\mathbf{x} = P(\mathbf{x})$, called the *Least Fixed Point* (LFP) solution, which gives us precisely the probabilities we are after. For vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, define the partial-order $\mathbf{x} \preceq \mathbf{y}$ to mean that $x_j \leq y_j$ for every coordinate j. For $D \subseteq \mathbb{R}^n$, we call a mapping $H : \mathbb{R}^n \mapsto \mathbb{R}^n$ *monotone* on D, if: for all $\mathbf{x}, \mathbf{y} \in D$, if $\mathbf{x} \preceq \mathbf{y}$ then $H(\mathbf{x}) \preceq H(\mathbf{y})$. Define $P^1(\mathbf{x}) = P(\mathbf{x})$, and define $P^k(\mathbf{x}) = P(P^{k-1}(\mathbf{x}))$, for $k > 1$.

Recall that $q^*_{(u,ex)}$ denotes the probability of eventually reaching $\langle \epsilon, ex \rangle$ starting at $\langle \epsilon, u \rangle$ in M_A. Let $\mathbf{q}^* \in \mathbb{R}^n$ denote the corresponding n-vector of probabilities (using the same indexing as used for $\mathbf{x}$). For $k \geq 0$, let $\mathbf{q}^k$ denote the n-vector of probabilities where $q^k_{(u,ex)}$ is the probability of reaching $\langle \epsilon, ex \rangle$ starting at $\langle \epsilon, u \rangle$ *in at most* k *steps* of M_A, meaning via a path in M_A of length at most k. Let $\mathbf{0}$ ($\mathbf{1}$) denote the n-vector consisting of 0 (respectively, 1) in every coordinate. Define $\mathbf{x}^0 = \mathbf{0}$, and for $k \geq 1$, define $\mathbf{x}^k = P(\mathbf{x}^{k-1}) = P^k(\mathbf{0})$.

Theorem 1. *Let* $\mathbf{x} = P(\mathbf{x})$ *be the system* S_A *associated with RMC* A*.*

1. $P : \mathbb{R}^n \mapsto \mathbb{R}^n$ *is monotone on* $\mathbb{R}^n_{\geq 0}$*. Hence, for* $k \geq 0$, $\mathbf{0} \preceq \mathbf{x}^k \preceq \mathbf{x}^{k+1}$.
2. *For all* $k \geq 0$, $\mathbf{q}^k \preceq \mathbf{x}^{k+1}$.
3. $\mathbf{q}^* = P(\mathbf{q}^*)$*. In other words,* $\mathbf{q}^*$ *is a* fixed point *of the map* P*.*
4. *For all* $k \geq 0$, $\mathbf{x}^k \preceq \mathbf{q}^*$.
5. $\mathbf{q}^* = \lim_{k\to\infty} \mathbf{x}^k$.
6. *For all* $\mathbf{q}' \in \mathbb{R}^n_{\geq 0}$*, if* $\mathbf{q}' = P(\mathbf{q}')$*, then* $\mathbf{q}^* \preceq \mathbf{q}'$.
In other words, $\mathbf{q}^*$ *is the* Least Fixed Point*,* LFP(P)*, of* $P : \mathbb{R}^n_{\geq 0} \mapsto \mathbb{R}^n_{\geq 0}$.

We have thus identified $\mathbf{q}^*$ as $\text{LFP}(P) = \lim_{k\to\infty} \mathbf{x}^k$. We can (naively) view Theorem 1 as giving an iterative algorithm to compute LFP(P), by computing the iterates $\mathbf{x}^k = P^k(\mathbf{0})$, $k \to \infty$, until we think we are "close enough". (But how do we know? Please read on.) We now observe several unfortunate properties of S_A that present obstacles for efficiently computing LFP(P).

Theorem 2. *All following RMCs, except the HMCs in (4), have one component, one entry* en*, and one exit* ex*.*

1. Irrational probabilities*: there is a RMC, A, such that the probability* $\mathbf{q}^*_{(en,ex)}$ *is an irrational number, and is in fact not "solvable by radicals". Thus, computing* LFP(P) exactly *is not possible in general.*
2. Slow convergence*: there is a RMC such that it requires an exponential number of iterations,* 2^{k-3}*, to gain* k *bits of precision.*
3. Qualitative questions not purely structure-dependent*: there are 2 "structurally" identical RMCs,* A' *and* A''*, that only differ in values of non-zero transition probabilities, but* $\mathbf{q}^*_{(en,ex)} = 1$ *in* A'*, while* $\mathbf{q}^*_{(en,ex)} < 1$ *in* A''*.*
4. Very small & very large probabilities*: There is a HMC, with* $m+1$ *components, and of total size* $O(m)$*, where component* A_m *has entry* en_m *and two exits* ex'_m *and* ex''_m*, such that* $\mathbf{q}^*_{(en_m,ex'_m)} = \frac{1}{2^{2^m}}$ *and* $\mathbf{q}^*_{(en_m,ex''_m)} = 1 - \frac{1}{2^{2^m}}$*.*

These facts illustrate some of contrasts between RMCs and finite Markov chains (MCs). For example, for finite MCs reachability probabilities are rational values that are efficiently representable and computable; moreover, qualitative questions, such as whether a state is reached with probability 1, only depend on the structure (edges) of the MC, and not on values of transition probabilities.

For RMC $A = (A_1, \ldots, A_k)$, let $\theta = \max_{i \in \{1,\ldots,k\}} \min\{|En_i|, |Ex_i|\}$. Note that $q^*_{(u,ex)} = 0$ iff there is no path in the *graph* of M_A from $\langle \epsilon, u \rangle$ to $\langle \epsilon, ex \rangle$. Reachability in RSMs was studied in [1, 4], where it was shown that the problem can be decided in $O(|A|\theta^2)$ time, thus:

Theorem 3. *(see [1, 4]) Given RMC* A*, we can determine in time* $O(|A|\theta^2)$*, for all vertices* u *and exits* ex*, whether or not* $\mathbf{q}^*_{(u,ex)} = 0$*.*

3 RMCs and the Existential Theory of Reals

We now show that the central reachability questions for RMCs can be answered by appealing to algorithms for deciding the *Existential Theory of the Reals*, $\boldsymbol{ExTh}(\mathbb{R})$. This consists of sentences in prenex form: $\exists x_1, \ldots, x_n R(x_1, \ldots, x_n)$, where R is a boolean combination of "atomic predicates" of the form $f_i(\mathbf{x}) \Delta 0$, where f_i is a multivariate polynomial with rational coefficients over the variables $\mathbf{x} = x_1, \ldots, x_n$, and Δ is a comparison operator $(=, \neq, \geq, \leq, <, >)$.

Beginning with Tarski, algorithms for deciding the First-Order Theory of Reals, $\boldsymbol{Th}(\mathbb{R})$, and its existential fragment $\boldsymbol{ExTh}(\mathbb{R})$, have been deeply investigated. In the current state of the art, it is known that $\boldsymbol{ExTh}(\mathbb{R})$ can be decided in PSPACE [8, 25, 5]. Furthermore it can be decided in exponential time, where the exponent depends (linearly) only on the number of variables; thus for a fixed number of variables the algorithm runs in polynomial time.

Suppose we want to decide whether a rational vector $\mathbf{c} = [c_1, \ldots, c_n]^T$ is $LFP(P)$. Consider the sentence: $\varphi \equiv \exists x_1, \ldots, x_n \bigwedge_{i=1}^n P_i(x_1, \ldots, x_n) = x_i \wedge \bigwedge_{i=1}^n x_i = c_i$. φ is true iff $\mathbf{c} = P(\mathbf{c})$. To guarantee that $\mathbf{c} = \text{LFP}(P)$, we additionally need: $\psi \equiv \exists x_1, \ldots, x_n \bigwedge_{i=1}^n P_i(x_1, \ldots, x_n) = x_i \wedge \bigwedge_{i=1}^n 0 \leq x_i \wedge \bigvee_{i=1}^n x_i < c_i$. ψ is false iff there is no solution $\mathbf{z} \in \mathbb{R}^n_{\geq 0}$ to $\mathbf{x} = P(\mathbf{x})$ such that $\mathbf{c} \not\preceq \mathbf{z}$. Hence, to decide whether $\mathbf{c} = \text{LFP}(P)$, we only need two queries to a decision procedure for $\boldsymbol{ExTh}(\mathbb{R})$. Namely, we check that φ is true, and hence $\mathbf{c} = P(\mathbf{c})$, and we

then check that ψ is false, and hence $\mathbf{c} = \mathrm{LFP}(P)$. Note that all multi-variate polynomials in our systems $\mathbf{x} = P(\mathbf{x})$ have (multivariate) degree $d \leq 2$.

Theorem 4. *Given a RMC A and given a vector of rational probabilities $\mathbf{c}$, there is a PSPACE algorithm to decide whether $\mathrm{LFP}(P) = \mathbf{c}$, as well as to decide whether $\mathbf{q}_j^* \; \Delta c_j$, for any comparison operator Δ. Moreover, the running time of the algorithm is $O(|A|^{O(1)} \cdot 2^{O(n)})$ where n is the number of variables in the system $\mathbf{x} = P(\mathbf{x})$. Hence the running time is polynomial if n is bounded.*

ExTh$(\mathbb{R})$ gives us a way to ask questions like: "Is there a solution to $\mathbf{x} = P(\mathbf{x})$ where $a \leq x_i \leq b$?" for any rational numbers a and b, and if we wish, with either inequality replaced by strict inequality. Since $\mathbf{0} \preceq \mathrm{LFP}(P) \preceq \mathbf{1}$, we can use such queries in a "binary search" to "narrow in" on the value of each coordinate of $\mathrm{LFP}(P)$. Via simple modifications of sentences like ψ, we can gain one extra bit of precision on the exact value of c_i with each extra query to ***ExTh***$(\mathbb{R})$. So, if we want j bits of precision for each c_i, $i = 1, ...n$, we need to make $j \cdot n$ queries. The sizes of the queries do not vary by much: only with an additive factor of at most j-bits, to account for the constants a and b. This discussion yields:

Theorem 5. *Given RMC A, and a number j in unary, there is an algorithm that approximates the coordinates of $\mathrm{LFP}(P)$ to within j bits of precision in PSPACE. The running time is $O(j \cdot |A|^{O(1)} \cdot 2^{O(n)})$, where n is the number of variables in $\mathbf{x}$.*

With a more involved construction we can handle in polynomial time all RMCs that have a constant number of components, each with a constant number of entries and exits; the components themselves can be arbitrarily large.

Theorem 6. *Given an RMC with a bounded total number of entries and exits, we can decide in polynomial time whether $\mathrm{LFP}(P) = \mathbf{c}$, or whether $\mathbf{q}_j^* \; \Delta c_j$, for any comparison operator Δ, and we can approximate each probability to within any given number of bits of precision. In particular, this applies to SCFGs with a bounded number of terminals and MT-BPs with a bounded number of types.*

4 RMCs and Newton's method

This section approaches efficient numerical computation of $\mathrm{LFP}(P)$, by studying how a classical numerical solution-finding method performs on the systems $\mathbf{x} = P(\mathbf{x})$. Newton's method is an iterative method that begins with an initial guess of a solution, and repeatedly "revises" it in an attempt to approach an actual solution. In general, the method may not converge to a solution, but when it does, it is typically fast. For example, for the bad RMC of Theorem 2.2, where the Iterative algorithm converges exponentially slowly, one can show that Newton's method converges exponentially faster, gaining one bit of precision per iteration. Recall that, given a univariate polynomial $f(x)$ (or more generally, a univariate differentiable function), and an initial guess x_0 for a root of $f(x)$, Newton's method computes the sequence $x_0, x_1, \ldots, x_k$, where $x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}$. There

is a natural n-dimensional version of Newton's method (see, e.g, [26] and [24]). 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. For each $\mathbf{c} \in \mathbb{R}^n$, $F'(\mathbf{c})$ is a real-valued matrix whose (i, j) entry is the polynomial $\frac{\partial F_i}{\partial x_j}$ evaluated at $\mathbf{c}$. For the method to be defined, $F'(\mathbf{x}_k)$ must be invertible at each point $\mathbf{x}_k$ in the sequence. Even when the $\mathbf{x}_k$'s are defined and a solution exists, Newton's method need not converge, and diverges even for some univariate polynomials of degree 3. We already know one convergent iterative algorithm for computing LFP(P). Namely, computing the sequence $\mathbf{x}^j = P^j(\mathbf{0})$, $j \to \infty$. Unfortunately, we saw in Thm. 2 that this algorithm can be very slow. The question arises whether Newton's method, applied to $F(\mathbf{x}) = P(\mathbf{x}) - \mathbf{x}$, can guarantee convergence to LFP(P), and do so faster. That is essentially what we establish in this section.

- Preprocess the system $\mathbf{x} = P(\mathbf{x})$, eliminating all variables $x_{(u,ex)}$ where $\mathbf{q}^*_{(u,ex)} = 0$.
- Construct the DAG of SCCs, H, based on the remaining system of equations.
- While (there is a sink SCC, C, remaining in the DAG H)
  - If C is the trivial SCC, $C = \{1\}$, then associate the value 1 with this node.
    Else, run Newton's method, starting at $\mathbf{0}$, on the equations for the set of variables in C, where these equations are augmented by the values of previously computed variables.
    Stop if a fixed point is reached, or when approximate solutions for C are considered "good enough".
    Store these final values for the variables in C and substitute these values for those variables in all remaining equations.
  - remove C from the DAG.

Fig. 1. Decomposed Newton's method

We cannot in general obtain convergence of Newton's method for the entire system $\mathbf{x} = P(\mathbf{x})$ at once, because for instance the condition on invertibility of the Jacobian may not hold in general. But it turns out that such anomalous cases can be avoided: we first preprocess the system (in linear time in its size, by Theorem 3) to remove all variables $x_{(u,ex)}$ where $\mathbf{q}^*_{(u,ex)} = 0$. Then we form a graph G whose nodes are the remaining variables x_i and the constant 1, and whose edges are (x_i, x_j) if x_j appears in $P_i(\mathbf{x})$, and edge $(x_i, 1)$ if $P_i(\mathbf{x}) \equiv 1$. We *decompose* the graph (and the system) into strongly connected components (SCCs) and apply Newton's method separately on each SCC bottom-up, as shown in Fig.1. In Fig.1 we have not specified explicitly how many iterations are performed. For concreteness in the following theorem, suppose that we perform k iterations for every SCC. Let $\mathbf{x}_k$ be the resulting tuple of values.

Theorem 7. *In the Decomposed Newton's Method of Fig. 1, the sequence* $\mathbf{x}_k, k \to \infty$*, monotonically converges to* $\mathbf{q}^*$*. Moreover, for all* $k \geq 0$*,* $\mathbf{x}_k \succeq P^k(\mathbf{0})$*.*

From our proof it actually follows that Newton's method in general constitutes a rapid "acceleration" of the standard iteration, $P^k(\mathbf{0})$, $k \to \infty$. In particular, for finite MCs, which generate linear systems, the decomposed Newton's method converges in one iteration to LFP(P).

Input: A SCFG G, with start non-terminal S_1.

1. Remove all nonterminals unreachable from S_1.
2. If there is any "useless" nonterminal left (i.e., a nonterminal that does not derive any terminal string), return NO.
3. For the remaining SCFG, let ρ be the maximum eigenvalue of the matrix $B(1)$ (the Jacobian matrix of $P(\mathbf{x})$, evaluated at the all 1-vector).
   If $\rho > 1$ then return NO; otherwise (i.e., if $\rho \leq 1$) return YES.

Fig. 2. SCFG consistency algorithm

5 1-exit RMCs and consistency of SCFGs

An SCFG is called *consistent* if it generates a terminal string with probability 1. We provide a simple, concrete efficient algorithm to check consistency using the connection of SCFG's to 1-exit RMC's and to multi-type Branching Processes. MT-BPs model the growth of a population of objects of a number of distinct types. The probability of extinction of a type in a MT-BP is related to the probability of the language generated by a SCFG. Using this connection and classical results on branching processes, one can "characterize" the question of termination of a SCFG as a question related to eigenvalues of certain matrices associated with the SCFG (see, e.g., [18] and [7, 17]). These "characterizations" unfortunately often omit special uncovered cases (and sometimes contain errors, eg. [7]) and do not give a complete algorithm.

Our algorithm for checking SCFG consistency is outlined in Fig. 2. The matrix $B(1)$ in the algorithm is precisely the Jacobian matrix of the system of polynomials $P(\mathbf{x})$, from section 4, where we substitute 1 for every variable x_i. To finish the algorithm, we only need to show that we can test in polynomial time whether the spectral radius of a non-negative rational matrix $B(1)$ is > 1. There are a number of ways to show this. One is by appealing to the existential theory of the reals. By the Perron-Frobenius theorem (see [21]), the maximum magnitude eigenvalue of a non-negative matrix is always real. Recall that the eigenvalues of a matrix M are the roots of the characteristic polynomial $h(x) = Det(M - xI)$. This univariate polynomial can be computed in polynomial time, and we can test whether $\rho(B(1)) > 1$ by testing the 1-variable sentence in $\boldsymbol{ExTh}(\mathbb{R})$: $\exists x(x > 1 \wedge h(x) = 0)$. More efficiently, for the non-negative matrices $B(1)$ we can also use Linear Programming to decide whether $\rho(B(1)) > 1$. Furthermore, with a more involved algorithm (see [15]) we can classify in one pass the termination probability of all the nonterminals (and all vertices of a 1-exit RMC).

Theorem 8. *Given a 1-exit RMC, A, there is a polynomial time algorithm to determine, for each vertex u and exit ex, which of the following three cases holds: (1) $\mathbf{q}^*_{(u,ex)} = 0$, or (2) $\mathbf{q}^*_{(u,ex)} = 1$, or (3) $0 < \mathbf{q}^*_{(u,ex)} < 1$. In particular, we can test SCFG consistency in polynomial time.*

6 RMCs and the Square-Root Sum Problem

We show that the square-root sum problem is reducible to the SCFG quantitative reachability problem, and to the general RMC qualitative reachability problem. Let SQUARE-ROOT-SUM be the following problem: given $(d_1, \ldots, d_n) \in \mathbb{N}^n$

and $k \in \mathbb{N}$, decide whether $\sum_{i=1}^{n} \sqrt{d_i} \leq k$. The complexity of this problem is open since 1976. It is known to be contained in PSPACE (e.g., by appeal to $\boldsymbol{ExTh}(\mathbb{R})$), however, it is not even known to be contained in NP. It is a major open problem in the complexity of exact numerical algorithms, with applications in computational geometry and elsewhere. (See, e.g., [16, 29, 22].)

Let SCFG-PROB be the following problem: given a SCFG (with rational edge probabilities) and given a rational number $p \in [0, 1]$, decide whether the SCFG terminates (i.e., produces a finite string) with probability $\geq p$.

Theorem 9. *SQUARE-ROOT-SUM is P-time reducible to SCFG-PROB.*

Let 2-EXIT-SURE be the following problem: given a 2-exit RMC with rational edge probabilities, and an entry-exit pair en and ex of some component, decide whether $q^*_{(u,ex)} = 1$. We can modify the above construction to show:[1]

Theorem 10. *SQUARE-ROOT SUM is P-time reducible to 2-EXIT-SURE.*

7 Conclusions

We introduced Recursive Markov Chains, and studied basic algorithmic problems in their analysis involving termination and reachability. A wide variety of techniques came into play, from the existential theory of the reals, theory of branching processes, numerical computing, etc. A number of questions remain open, both for the problems we have investigated and for further directions. For example, we proved that Newton's method converges monotonically, and dominates the iterative algorithm. We expect that this is the practical way to approximate the probabilities, and we believe that in fact Newton gains i bits of precision in a polynomial number of iterations in the unit-cost real RAM model. Moreover, the reductions from the square-root sum problem to deciding whether a probability is $\leq p$ does not preclude the possibility that these probabilities can be *approximated* to i bits of precision in P-time without yielding a P-time solution to the square-root sum problem. Indeed, sum of square-roots itself can be so approximated using Newton's method.

A number of further directions are worth pursuing, building upon this work. We have extended our methods to algorithms for the verification of linear time properties of RMC's ([14]). Another direction we are pursuing is the analysis of Recursive Markov Decision Processes and Recursive Stochastic Games.

References

1. R. Alur, K. Etessami, and M. Yannakakis. Analysis of recursive state machines. In *Proc. of 13th Int. Conf. on Computer-Aided Verification*, pp. 304–313, 2001.
2. S. Abney, D. McAllester, and F. Pereira. Relating probabilistic grammars and automata. *Proc. 37th Ann. Ass. for Comp. Linguistics*, pp. 542–549, 1999.

[1] Theorem 10 has also been observed independently by J. Esparza & A. Kucera ([11]) based on a preliminary draft of this paper which included Theorem 9.

3. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: App's to model checking. In *CONCUR'97*, pp. 135–150, 1997.
4. M. Benedikt, P. Godefroid, and T. Reps. Model checking of unrestricted hierarchical state machines. In *Proc. of ICALP'01*, LNCS 2076, pp. 652–666, 2001.
5. S. Basu, R. Pollack, and M. F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *J. of the ACM*, 43(6):1002–1045, 1996.
6. S. Basu, F. Pollack, and M. F. Roy. *Algorithms in Real Algebraic Geometry.* Springer, 2003.
7. T. L. Booth and R. A. Thompson. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, 22(5):442–450, 1973.
8. J. Canny. Some algebraic and geometric computations in PSPACE. In *Prof. of 20th ACM STOC*, pp. 460–467, 1988.
9. Z. Chi and S. Geman. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 1997.
10. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *12th CAV*, LNCS 1855, pp. 232–247, 2000.
11. J. Esparza and A. Kučera. *personal communication*, 2004.
12. Javier Esparza, Antonín Kučera, and Richard Mayr. Model checking probabilistic pushdown automata. In *LICS 2004*, 2004.
13. C. J. Everett and S. Ulam. Multiplicative systems, part i., ii, and iii. Technical Report 683,690,707, Los Alamos Scientific Laboratory, 1948.
14. K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic systems. Technical report, School of Informatics, U. of Edinburgh, 2004. *Submitted.*
15. K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of non-linear equations. Technical report, School of Informatics, University of Edinburgh, 2004. (Full paper).
16. M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *8th ACM STOC*, pp. 10–22, 1976.
17. U. Grenander. *Lectures on Pattern Theory, Vol. 1.* Springer-Verlag, 1976.
18. T. E. Harris. *The Theory of Branching Processes.* Springer-Verlag, 1963.
19. P. Jagers. *Branching Processes with Biological Applications.* Wiley, 1975.
20. A. N. Kolmogorov and B. A. Sevastyanov. The calculation of final probabilities for branching random processes. *Dokaldy*, 56:783–786, 1947. (Russian).
21. P. Lancaster and M. Tismenetsky. *The Theory of Matrices.* Academic Press, 2nd edition, 1985.
22. G. Malajovich. An effective version of Kronecker's theorem on simultaneous diophantine equations. Technical report, City U. of Hong Kong, 1996.
23. C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.
24. J. M. Ortega and W.C. Rheinbolt. *Iterative solution of nonlinear equations in several variables.* Academic Press, 1970.
25. J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. parts i,ii, iii. *J. of Symbolic Computation*, pp. 255–352, 1992.
26. J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer, 1993.
27. Y. Sakakibara, M. Brown, R Hughey, I.S. Mian, K. Sjolander, R. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, 22(23):5112–5120, 1994.
28. B. A. Sevastyanov. The theory of branching processes. *Uspehi Mathemat. Nauk*, 6:47–99, 1951. (In Russian).
29. P. Tiwari. A problem that is easier to solve on the unit-cost algebraic ram. *Journal of Complexity*, pages 393–397, 1992.