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

Kousha Etessami
University of Edinburgh

Mihalis Yannakakis
Columbia University

Abstract

We define *Recursive Markov Chains* (RMCs), a class of finitely presented denumerable Markov chains, and we study algorithms for their analysis. Informally, an RMC consists of a collection of finite-state Markov chains with the ability to invoke each other in a potentially recursive manner. RMCs offer a natural abstract model for probabilistic programs with procedures. They generalize, in a precise sense, a number of well studied stochastic models, including Stochastic Context-Free Grammars (SCFG) and Multi-Type Branching Processes (MT-BP).

We focus on algorithms for *reachability* and *termination* analysis for RMCs: what is the probability that an RMC started from a given state reaches another target state, or that it terminates? These probabilities are in general irrational, and they arise as (least) fixed point solutions to certain (monotone) systems of nonlinear equations associated with RMCs. We address both the *qualitative problem* of determining whether the probabilities are 0, 1 or in-between, and the *quantitative problems* of comparing the probabilities with a given bound, or approximating them to desired precision.

We show that all these problems can be solved in PSPACE using a decision procedure for the Existential Theory of Reals. We provide a more practical algorithm, based on a decomposed version of multi-variate Newton's method, and prove that it always converges monotonically to the desired probabilities. We show this method applies more generally to any monotone polynomial system. We obtain polynomial time algorithms for various special subclasses of RMCs. Among these: for SCFGs and MT-BPs (equivalently, for *1-exit* RMCs) the qualitative problem can be solved in P-time; for *linearly-recursive* RMCs the probabilities are rational and can be computed exactly in P-time.

We show that our PSPACE upper bounds cannot be substantially improved without a breakthrough on longstanding open problems: the *square-root sum* problem and an arithmetic circuit decision problem which captures P-time on the unit-cost rational arithmetic RAM model. We show that these problems reduce to the qualitative problem and to the approximation problem (to within any nontrivial error) for termination probabilities of general RMCs, and to the quantitative decision problem for termination (extinction) of SCFGs (MT-BPs).

*To appear in J. ACM. A preliminary version of this paper appeared in *Proc. of 22nd Symp. on Theoretical Aspects of Computer Science*, 2005, pp. 340-352.

1 Introduction

We study and provide algorithms for analysis of *Recursive Markov Chains* (RMCs), a natural model for systems that involve both probability and recursion. Informally, a Recursive Markov Chain consists of a collection of finite-state component Markov chains which can call each other in a potentially recursive manner. RMCs are a probabilistic version of Recursive State Machines (RSMs) ([AEY01, BGR01]). These and other expressively equivalent non-probabilistic models, e.g., Pushdown Systems (PDSs), have been studied extensively in recent research on software model checking and program analysis, because of their applications to modeling and verification of sequential programs with procedures (i.e., subroutines). RMCs are in turn a natural abstract model for probabilistic procedural programs. Probabilistic models of programs are of interest for a number of 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 behavior or on the behavior of its input environment. Under such assumptions, we may want to determine the probability of properties of interest, e.g., that the program terminates, and/or that it terminates in a certain state. Beyond the analysis of probabilistic programs, the RMC model is of interest in its own right as a basic model that combines two very common modelling primitives: probability and recursion.

We now give an example and a brief description of RMCs to facilitate intuition (formal definitions are provided in section 2). Figure 1 visually depicts an example of a Recursive Markov Chain with two *component* Markov chains, A_1 and A_2. Each component has certain designated *entry* and *exit* nodes. For example, component A_1 has one entry, en, and two exits, ex_1 and ex_2. In this example, A_2 also has one entry and two exits, but in general, the components of an RMC may have different numbers of entries and exits. These components may have other, ordinary, nodes, e.g., u in A_1. In addition to ordinary nodes, each component may also contain *boxes*, e.g., the box b_1 in A_1, and the boxes b'_1 and b'_2 in A_2. The boxes model recursive calls. Each box is mapped to a component, and acts (just like a "function call") as a place-holder for a copy of the component to which it is mapped. For example, component A_1 has one box, b_1, which is mapped to A_2. Every box has *call ports* and *return ports*, which correspond 1-1 to the entries and exits, respectively, of the component to which the box is mapped. In this case note that the call ports and return ports of box b_1 correspond to the entries and exits of A_2, respectively. A transition into a box goes to a specific call port and models the invocation of the component to which the box is mapped, starting at the entry node corresponding to the call port; this can be viewed as a function call where the call port is the parameter value passed to the function. When (and if) the called component terminates at an exit, then the execution of the calling component resumes from the corresponding return port of the box; this is like a return from a function, where the exit at which the call terminated is the returned value. Probabilities label the transitions of an RMC, as shown in the figure. Intuitively, "macro-expanding" each box using the component to which it corresponds, and doing this "for ever" as long as there are boxes remaining, generates the underlying denumerable Markov chain which is described by the RMC in a concise, finite, fashion. We are interested in the properties of this underlying denumerable Markov chain.

A basic computational question that will concern us in this paper, and which forms the backbone of many other analyses for RMCs, is the following: given an RMC, and

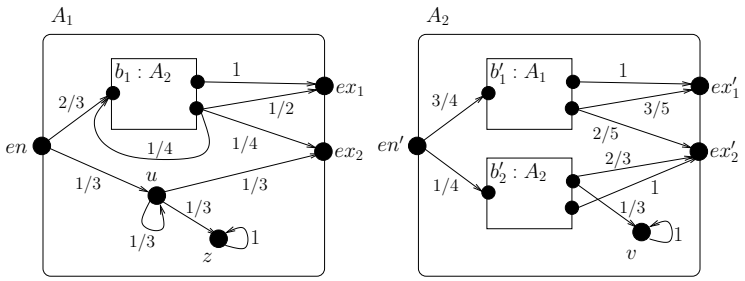

Figure 1: *A sample Recursive Markov Chain*

given a vertex u and exit ex, both from the same component, what is the probability that starting at u (in the empty calling context, i.e., not inside any boxes), we will eventually terminate at ex (in the empty calling context)? This is what we call the *termination probability*, $q^*_{u,ex}$, associated with the pair u and ex. Computation of such probabilities is crucial to many other analyses of RMCs. As we shall see, such probabilities can be irrational, so we can not hope to compute them "exactly". We must instead either be content with approximating the probabilities, or answering decision questions about them, such as whether the probability is at least a desired rational value. We shall also see that the problem of computing/approximating such probabilities encounters a number of other difficulties not encountered in the case of finite-state Markov Chains.

It turns out that basic analysis questions about RMCs generalize related questions about several classic probabilistic models that have been studied for decades. These include *(multi-type) Branching Processes* (MT-BPs) an important class of stochastic processes first defined by Kolmogorov, and studied by him and Sevastyanov and others beginning in the 1940s and continuing to the present (see, e.g., [Har63, KS47, Sev51, AN72, Jag75, KA02, HJV05]). The theory of Branching Processes dates back (in the single-type case) to the 19th century and the work of Galton and Watson on population dynamics. Multi-type BPs and their variants have been applied in a wide variety of stochastic contexts, including population genetics ([Jag75]), models in molecular biology ([KA02, HJV05]), and the study of nuclear chain reactions ([EU48]). Many variants and extensions of MT-BPs have also been studied. The problem of computing *extinction probabilities* for MT-BPs, which was already considered by Kolmogorov and Sevastyanov in the 1940s [KS47], is in fact equivalent to that of computing termination probabilities for a special restricted class of RMCs, namely *1-exit* RMCs, where every component has exactly one exit node. Another directly related class is Stochastic Context-Free Grammars (SCFGs). As we shall show, the problem of computing the probability of the language generated by a SCFG is also equivalent to computing the termination probability of a 1-exit RMC. SCFGs have been studied extensively since the 1970s, particularly in the Natural Language Processing (NLP) community where they are a core model (see, e.g., [MS99]) as well as well as in biology sequence analysis (see, e.g., [DEKM99, SBH$^+$94]). (For definitions of MT-BPs and SCFGs, see section 2.3.)

As we shall see, general RMCs are a more expressive model and have different complexity characteristics. A model that is expressively equivalent to general RMCs is probabilistic Pushdown Systems (pPDSs), introduced independently and concurrentlly in [EKM04]. As we'll see, there are linear time translations between RMCs and pPDSs.

Despite the extensive study of both MT-BPs and SCFGs over many decades, a number of basic algorithmic questions about them have not been satisfactorily answered. For example, is the probability of the language of a given SCFG or the extinction probability of a MT-BP $\geq p$? 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 the rich branching process literature.

Our results, and the structure of this paper. The goal of this paper is to develop the basic theory and explore the fundamental algorithmic properties of Recursive Markov Chains. The focus will be on computation of termination probabilities and its complexity, because, as explained, computation of these probabilities forms the core of many other important analyses.[1] We shall observe that we can easily reduce to termination more general reachability questions: what is the probability that the RMC starting from a given state (with empty context, i.e. no pending recursive calls) will reach another target state (with empty context, or for some context)?

As we mentioned, the termination probabilities are generally irrational, and hence cannot be computed exactly. We address the *qualitative problem* of determining whether the probabilities are 0, 1 or in-between, and the *quantitative problems* of (i) approximating these probabilities to desired precision (the *approximation problem*), and (ii) comparing the probabilities to given bounds (the *decision problem*). We provide both upper and lower bounds, for the general class of RMCs and for several important subclasses.

We first give a brief overview of the results, and we then present a more detailed summary. For upper bounds, we show that the qualitative and quantitative problems can be solved in PSPACE, using a decision procedure for the Existential Theory of Reals, and we provide a more practical numerical algorithm, based on a decomposed version of multi-variate Newton's Method, which we show converges monotonically to the desired probabilities. We obtain more efficient algorithms for several important subclasses of RMCs: hierarchical, linearly-recursive, bounded, and 1-exit RMCs (these classes are defined formally in Section 2). For lower bounds, we show that our PSPACE bounds cannot be substantially improved upon without a breakthrough on the square-root sum problem (`SQRT-SUM`), a long-standing open problem in the complexity of numerical computation, and on another more powerful and fundamental problem, called `PosSLP`, which is complete for the class of decision problems that can be solved in polynomial time on models with unit-cost exact rational arithmetic. We show that these problems reduce to the quantitative decision problem for 1-exit RMCs (and SCFGs and MT-BPs) of comparing the termination probabilities with a given bound. They also reduce to both the qualitative and quantitative problems for general RMCs, and furthermore to the approximation of the termination probabilities with any nontrivial constant error.
We now summarize in more detail the main results of this paper, and we outline the organization to help guide the reader.

Section 2: Basic definitions and background: We give the formal definition of Recursive Markov Chains, and define, in one place for easy reference, several special subclasses of

[1] Indeed, as described in the conclusions, a number of papers have appeared since the publication of the conference version of this paper, which use the analyses described here as a basis for other analyses, such as for model checking, analysis of expected termination time (hitting time), and more.

RMCs: hierarchical, linear, bounded, and 1-exit RMCs (Section 2.1). We define formally the main (qualitative and quantitative) problems addressed in the paper concerning the termination probabilities of an RMC, and observe that the computation of more general reachability probabilities reduces to computation of termination probabilities (Section 2.2). We then give (in Section 2.3) the definition of SCFGs and MT-BPs, and establish formally their relationship to 1-exit RMCs: we present polynomial time translations between SCFGs and 1-exit RMCs in both directions, such that the probability of the language of the SCFG is equal to a termination probability of the RMC (Theorem 2.3). Similar translations are presented for MT-BPs and 1-exit RMCs, establishing the equality between the extinction probabilities of a branching process and the termination probabilities of the corresponding 1-exit RMC (Theorem 2.4).

Section 3: The nonlinear equation system for RMCs: From an RMC we construct a system of equations, denoted $\mathbf{x} = P(\mathbf{x})$ in vector notation, where $\mathbf{x}$ is an n-vector of variables corresponding to the termination probabilities, and where each of the n equations, $\mathbf{x}_i = P_i(\mathbf{x})$ (one for each termination probability), has on its right hand side a multi-variate polynomial, $P_i(\mathbf{x})$, with only positive coefficients. The particular systems associated with RMCs have the additional property that they always have a non-negative solution, and in fact a *least* non-negative solution, i.e., a non-negative solution which is smallest in every coordinate than any other solution. This solution is the *Least Fixed Point* (LFP) of the monotone operator $P : \mathbb{R}^n_{\geq 0} \mapsto \mathbb{R}^n_{\geq 0}$. We show that the LFP is precisely the vector $\mathbf{q}^*$ of termination probabilities for the RMC (Theorem 3.1).

The monotone nonlinear system $\mathbf{x} = P(\mathbf{x})$ for an RMC gives rise to a natural iterative numerical algorithm with which to approximate the LFP. Namely, $\mathbf{q}^* = \lim_{k\to\infty} P^k(\mathbf{0})$, where $P^1(\mathbf{0}) = P(\mathbf{0})$ and $P^{k+1}(\mathbf{0}) = P(P^k(\mathbf{0}))$. We show that this standard iteration can be exponentially slow to converge to within i bits of $\mathbf{q}^*$, and this holds even for a fixed 1-exit RMC (Theorem 3.2.) (We give a superior iterative numerical algorithm later in Section 6.) We also present a number of examples (also in Theorem 3.2) to illustrate other numerical and computational pathologies which make the problem of computing/approximating these probabilities very different than that of finite-state Markov chains. For example, we observe that the termination probabilities can be irrational (even for MT-BPs and SCFGs), and not solvable by radicals. Thus we can't hope to compute them exactly.

Section 4: Basic upper bounds: We show that for general RMCs we can decide in PSPACE whether a termination 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 (Theorems 4.2 and 4.3.) These results are shown by appealing to the deep results on the complexity of decision procedures for the Existential Theory of Reals ([Can88, Ren92, BPR03]). Better results are obtained later in Section 8 for important special classes of RMCs.

Section 5: "Lower Bounds": We show that one can not hope to improve substantially on the PSPACE upper bounds for probabilities of RMCs without a major breakthrough in the complexity of exact numerical computation. We do this by establishing reductions from two important open problems. The first of these is the *square-root sum problem* (`SQRT-SUM`), which asks, given natural numbers $(d_1, \ldots, d_n)$ and another natural number k, whether $(\sum_i \sqrt{d_i}) \geq k$. This problem is known to be in PSPACE, but its containment even in NP is a longstanding open problem first posed in 1976 ([GGJ76]). This problem arises often especially in geometric computations where the square root sum represents the sum

of Euclidean distances between given pairs of points with integer (or rational) coordinates; for example, determining whether the length of a spanning tree or a TSP tour of given points on the plane is bounded by a given threshold amounts to answering an instance of the `SQRT-SUM` problem. The second problem, which is harder than square-root sum via P-time Turing reductions (but not known to be harder via P-time many-one reductions), is the `PosSLP` (Positive Straight-Line-Program) problem considered recently by Allender et. al. in [ABKPM06]. The `PosSLP` problem asks whether a given arithmetic circuit (equivalently, a straight-line program) with integer inputs, and gates $\{+, *, -\}$, outputs a positive number or not. The importance of `PosSLP` was highlighted in [ABKPM06] which showed that this problem is hard (under P-time Turing reductions) for the entire class of decision problems that can be decided in polynomial time in the Blum-Shub-Smale model of computation over the reals using rational constants [BCSS98] or equivalently, a *unit-cost algebraic RAM* model in which all operations on rationals take unit time, no matter how large the numbers. Importantly, the division operation is exact rational division, not integer division; it is known that with integer division (the floor function) all of PSPACE can be decided in the unit-cost model in polynomial time [Sch79, BMS81]. It is an open question whether the unit-cost RAM model without integer division can decide in polynomial time any problem that is not in P in the usual Turing machine model (equivalently, RAM model with logarithmic cost); Tiwari showed in [Tiw92] that the square-root sum problem can be solved in polynomial time in this model, as an example of a problem that we currently do not know how to solve in P in the standard model. Allender et. al. [ABKPM06] showed that `PosSLP` can be decided in the 4th level of the *Counting Hierarchy* (CH), an analog of the polynomial-time hierarchy for counting classes like $\#P$, and since `SQRT-SUM` is P-time Turing reducible to `PosSLP` they were also able to conclude that `SQRT-SUM` is in CH. Thus it is unlikely that either of these problems is PSPACE-hard, but it remains an important open question whether either problem can be decided in P or even in NP.

We show that the `SQRT-SUM` and `PosSLP` problems reduce (both via P-time many-one reductions) to the qualitative termination problem for RMCs (even for 2-exit RMCs), i.e., determining whether a 2-exit RMC terminates with probability 1. Furthermore, even any non-trivial approximation of the termination probability for a 2-exit RMC is at least as hard as the `SQRT-SUM` and `PosSLP` problems. Specifically, for any $\epsilon > 0$, both these problems are reducible to the following problem: given a 2-exit RMC which is guaranteed to either terminate with probability 1 or with probability $\leq \epsilon$, decide which of the two is the case (Theorem 5.2). We also show that the `SQRT-SUM` and `PosSLP` problem are polynomial-time reducible to the decision problem for 1-exit RMCs (Theorems 5.1 and 5.3), i.e., determining whether the termination probability is $\leq p$ for given $p \in [0, 1]$; this applies in particular to the problem of bounding the extinction probability of a branching process or the probability of the language generated by a SCFG. We in fact show that `PosSLP` is (many-one) reducible to the termination problem for the further restricted class of 1-exit hierarchical RMCs (Theorem 5.3), which have rational termination probabilities (but which can require exponential size).

Section 6: RMCs and decomposed Newton's method: the available PSPACE decision procedures for the Existential Theory of the Reals (see [Can88, Ren92, BPR03]) yield the current best worst-case complexity upper bounds for quantitative termination decision problems for RMCs. However, there are large constants in the (exponential) running time of these algorithms, and they are impractical for large instances. In practice, exact

decision procedures aren't always necessary for such quantitative problems, and it would be desirable to have efficient numerical procedures for estimating the termination probabilities for RMCs. We provide an iterative numerical algorithm, based on a decomposed Newton's method, for estimating termination probabilities of RMCs. The algorithm converges monotonically to the termination probabilities, and in practice it converges very quickly for even fairly large instances, although the worst-case number of iterations required is still not fully understood. (We discuss the current state of our understanding of the behavior of this algorithm in the concluding section 10.) It is important to point out that such numerical algorithms are not a substitute for exact decision procedures such as those for the existential theory of reals, because in general they do not yield a method to decide exact quantitative questions (e.g., is the termination probability at least 1/2?) but rather they only allow iterative approximation of the desired values. In more detail, we show that a decomposed version of (multi-dimensional) Newton's method, where Newton's method is applied separately to each strongly connected component of the equation system, converges monotonically to the Least Fixed Point solution (and thus to the termination probabilties) starting from the all zero vector (Theorem 6.1). The proof shows, furthermore, that this method constitutes a rapid "acceleration" of the standard iteration of the monotone system of nonlinear equations. Note that in general Newton's method is not guaranteed to converge for arbitrary nonlinear polynomial systems, but when it does converge it typically converges very fast. We show that it always converges monotonically to the LFP in our setting. We thus believe that in our context Newton provides an efficient practical method for numerically estimating these probabilities for typical RMCs. (As discussed in the conclusions, more recent work [KLE07] has revealed examples where even our decomposed Newton's method can converge slowly. But implementation and experimental studies have confirmed that, over a wide range of examples, our decomposed Newton's method performs well in practice [WE07, NS06].)

Section 7: General monotone systems of polynomial equations: We show that essentially all our analyses for the nonlinear systems for RMCs generalize to any system of equations $\mathbf{x} = P(\mathbf{x})$, where P is a vector of multivariate polynomials with positive coefficients. These more general systems may not have any finite non-negative solutions, but if they do then they will have a finite least fixed point (LFP). We show that the techniques developed for analysis and computation of the LFP of RMC equations are also applicable to these more general systems, including Newton's method , i.e., if the system has a solution, then the decomposed Newton's algorithm will converge monotonically to the LFP (Corollary 7.5).

Section 8: P-time algorithms for special classes of RMCs: We give efficient polynomial time algorithms for analysis of several special classes of RMCs.

1. We show that for 1-exit RMCs, MT-BPs, and SCFGs, we can solve in polynomial time the qualitative problem, of deciding whether the probability of termination (resp., extinction, language) is exactly 1, i.e., *almost sure* termination (Theorem 8.1).

2. We show that the quantitative problems can be solved in polynomial time for *bounded* RMCs, which are RMCs that have a bounded number of components, and each component has a bounded number of entries and exits (Theorem 8.8). These correspond to programs with a constant ('small') number of procedures that pass in and out a constant amount of information; the components (the procedures) themselves can be arbitrarily large and complex.

3. Finally, we consider the class of *linearly recursive* RMCs, where there are no positive

probability paths from the return port of some box to the call port of a box in the same component (this corresponds to linear recursion in programs), and the class of *hierarchical Markov chains* (HMC), where the call graph between the components is acyclic. These classes inherit some of the nice properties of finite-state Markov chains. We show that for both of these classes, the termination probabilities are rational. In the case of linearly recursive RMCs the probabilities have polynomial bit size and can be computed exactly in polynomial time (Theorem 8.9). For hierarchical MCs we can solve the qualitative problem in polynomial time. Furthermore, if the number of levels in the hierarchy is bounded then the probabilities have polynomial bit size and can be computed exactly in polynomial time (Theorem 8.11). If the number of levels is unbounded, then the probabilities may have exponential bit size, hence cannot be computed in polynomial time in the standard Turing machine model. We show (Corollary 8.12) that the decision problem for hierarchical Markov chains is complete (under P-time Turing reductions) for the class of problems that can be decided in polynomial time in the BSS model mentioned earlier, with unit-cost exact rational arithmetic.

Section 9: Relation to other models: We detail the relationship between RMCs and several other probabilistic models. In particular, we show that RMCs are expressively equivalent to *probabilistic Pushdown Systems* (pPDSs), a model introduced independently in [EKM04], and we provide linear-time translations in both directions between the two models (Theorem 9.1). We also observe that the *Random Walk with Backbutton* model studied in [FKK$^+$00] as a probabilistic model of web browsing/crawling, constitutes a special subclass of 1-exit RMCs.

Related Work. There is extensive work in the verification and program analysis literature on algorithmic analysis of non-probabilistic models of procedural programs, based on Pushdown Systems and related models (see, e.g., [BEM97, EHRS00, Rep98]). Recursive state machines were introduced in [AEY01, BGR01] (see the journal version [ABE$^+$05]) as a more direct graphical model of procedural programs, expressively equivalent to Pushdown Systems, and their algorithmic verification questions were thoroughly investigated.

A conference version of this paper appeared in [EY05b]. (Some results given here were not stated in [EY05b]. Specifically, hardness results with respect to the `PosSLP` problem were not in [EY05b], Theorem 5.2 appeared for the first time (stated without proof) in a later conference paper [EY07], and Theorem 5.3 appears for the first time in this paper.) A work directly related to this paper, done independently and concurrently, is that of Esparza, Kucera, and Mayr [EKM04] who considered model checking for probabilistic pushdown systems (pPDSs). pPDSs and RMCs are expressively equivalent models: as we show in section 9 there are efficient linear-time translations between the two. Among the results in [EKM04], they showed decidability of reachability questions for pPDSs by constructing essentially the same nonlinear system of equations for pPDSs that we construct and associate with RMCs. They then appealed to results on the theory of reals to derive EXPTIME upper bounds for reachability in pPDSs. As we point out, the known results for the existential theory of reals can actually be used to obtain PSPACE upper bounds for reachability. That is essentially the main overlap between their results and our results in this paper. Their main focus was decidability (rather than precise complexity) of model checking problems for pPDSs for properties expressed by deterministic Büchi automata, and those expressed in a probabilistic branching-time temporal logic, PCTL. Subsequent papers by ourselves and others, which build on this paper, have de-

veloped improved model checking algorithms and complexity bounds for all linear-time properties (expressed by nondeterministic Büchi automata or Linear Temporal Logic) [BKS05, EY05a, YE05]. Since the conference publication of our paper [EY05b] and of [EKM04], a number of conference papers have been published that build on and extend this work in various directions (see [BKS05, EKM05, BEK05, KLE07, EKL08] and see our papers [EY05a, EY05c, YE05, EY06a, EY06b]). We will give a brief description of this work in the concluding section.

As mentioned earlier, SCFGs have been studied extensively in the Natural Language Processing literature (see, e.g., [MS99]). In particular, the problem of *consistency* of a SCFG (whether the language that it generates has probability 1) has been studied, and its connection to the extinction problem for branching processes is well known [Har63, BT73, Gre76, CG98]. However, none of the relevant references provide a complete algorithm, characterization, and proof for consistency.

The branching process literature on computing extinction probabilities is old and extensive (see [Har63, AN72, Mod71] for thorough expositions). However, even there, no reference provides a complete algorithm and proof for deciding almost sure termination for all branching processes in polynomial time. The most comprehensive results (in Russian) are due to Sevastyanov and Kolmogorov [Sev51, KS47] (see [Har63]). We elaborate in detail on those results in section 8.1.

Another related work is [AMP99]. They study probabilistic Pushdown Automata (pPDA), and their relationship to SCFGs and weighted CFGs. Among their results they show that for every pPDA there is a SCFG which yields the same probability distribution on strings. However, that construction is not computationally useful in the following sense: the resulting SCFG uses grammar rules whose probabilities are given by the termination probabilities of the original pPDA, and thus in order to actually "construct" this SCFG one first has to compute these termination probabilities for pPDSs, so this computational problem is not addressed. Note also that these probabilities may be irrational, so constructing the resulting SCFG exactly is in fact problematic. The paper [AMP99] does not address the computation of these probabilities for pPDAs, nor other algorithmic questions for analysis of SCFGs and pPDA.

Another case of a model that has probabilistic and recursive features is that of Fagin, et. al. [FKK$^+$00], on Random walks with "back buttons". They study a probabilistic model of surfing/crawling on the WWW, where from each web page, the user with some probability either follows a link to go to a new page or pushes the back-button to return to the previous page. They study both steady-state/limit distributions and termination probabilities for these models, and show that semi-definite programming can be used to approximate these probabilities in polynomial time. It turns out, as we will explain in section 9, that the back-button model corresponds to a (strict) subclass of 1-exit RMCs; the subclass is strict and cannot generate various distributions that 1-exit RMCs can generate.

2 Basic definitions and background

In Section 2.1 we will define formally Recursive Markov Chains, and several special subclasses. In Section 2.2 we define the problems that we will consider in this paper concerning termination probabilities, and show that more general reachability probabilities can be

reduced to them. In Section 2.3 we will give the definitions of Stochastic Context-Free Grammars and Multitype Branching Processes, and relate them to 1-exit RMCs.

2.1 The RMC Model

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*, and a mapping $Y_i : B_i \mapsto \{1, \ldots, k\}$ that assigns to every box (the index of) one of the components, $A_1, \ldots, A_k$. To each box $b \in B_i$, we associate a set of *call ports*, $Call_b = \{(b, en) \mid en \in En_{Y_i(b)}\}$ corresponding to the entries of the corresponding component, and a set of *return ports*, $Return_b = \{(b, ex) \mid ex \in Ex_{Y_i(b)}\}$, corresponding to the exits of the corresponding component.
- 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 $u = (b, ex)$ of a box $b \in B_i$,
  2. The destination v is either a non-entry node $v \in N_i \setminus En_i$, or a call port $u = (b, en)$ of a box $b \in B_i$,
  3. $p_{u,v} \in \mathbb{R}_{>0}$ is the transition probability from u to v,
  4. *Consistency of probabilities*: for each u, $\sum_{\{v' \mid (u, p_{u,v'}, v') \in \delta_i\}} p_{u,v'} = 1$, unless u is a call port or exit node, neither of which have outgoing transitions, in which case by default $\sum_{v'} p_{u,v'} = 0$.

For an example, see Figure 1 and its description in the Introduction.

For computational purposes, we assume that the transition probabilities $p_{u,v}$ are rational, and we measure their size (bit-complexity) by the number of bits in the numerator and denominator. 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. Thus, the transition relation δ_i is a set of probability-weighted directed edges on the set Q_i of vertices of A_i. We will use all the notations without a subscript to refer to the union over all the components of the RMC A. Thus, $N = \cup_{i=1}^{k} N_i$ denotes the set of all the nodes of A, $Q = \cup_{i=1}^{k} Q_i$ the set of all vertices, $B = \cup_{i=1}^{k} B_i$ the set of all the boxes, $Y = \cup_{i=1}^{k} Y_i$ the map $Y : B \mapsto \{1, \ldots, k\}$ of all boxes to components, and $\delta = \cup_i \delta_i$ 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. The sequence β represents the stack of pending recursive calls and u is the current vertex. 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 $\langle \beta, (b, en) \rangle \in V$, $(b, en) \in \mathit{Call}_b$, then
   $\langle \beta b, en \rangle \in V$ and $(\langle \beta, (b, en) \rangle, 1, \langle \beta b, en \rangle) \in \Delta$

4. if $\langle \beta b, ex \rangle \in V$, $(b, ex) \in \mathit{Return}_b$, 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, item 2 corresponds to a transition within a component, item 3 corresponds to a recursive call when a new component is entered via a box, and item 4 correspond to the termination of a recursive call 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. To simplify notation, we will sometimes use in place of $\langle \epsilon, u \rangle$ the notation $\langle u \rangle$ or simply u; so for example, we will often say that a path of the RMC A starts at vertex u and terminates at an exit ex of the component of u, to mean a path of M_A from state $\langle \epsilon, u \rangle$ to state $\langle \epsilon, ex \rangle$.

Special Classes of RMCs.

We give here, together in one place for easy reference, definitions of several special classes of RMCs. As with procedural programs, from an RMC A we can define the *call graph* of A: the graph has one node $i = 1, \ldots, k$ for each component A_i of A and has a directed edge (i, j) if a box of A_i is mapped to A_j. RMCs whose call graph is acyclic are called *Hierarchical Markov Chains* (HMCs). These are the probabilistic version of Hierarchical State Machines [AY01]. Hierarchy is often used to structure large models and represent them in a succinct, modular fashion. In this special case M_A is finite, but can be exponentially larger than the HMC which specifies it.

We say that an RMC is *linearly recursive* or simply a *linear RMC*, if there is no positive probability path in any component (using only transitions of that component) from a return port of any box to a call port of any box (neither the same nor another box). This corresponds to the usual notion of linear recursion in procedures. As an example, the RMC of Fig. 1 is not linear because of the back edge from the second return port of box b_1 to its call port; if this edge was not present then the RMC would be linear.

The class of *bounded* RMCs (for some fixed bound c) is the set of RMCs that have a bounded number (at most c) of components, each of which has a bounded number (at most c) of entries and exits. The components themselves can be arbitrarily large, i.e. have an arbitrary number of vertices, boxes, and edges. These correspond to recursive programs with a bounded number of different procedures, which pass a bounded number of input and output values.

The class of *1-exit RMCs* is the set of RMCs all of whose components have only one exit; there can be an arbitrary number of components with an arbitrary number of entries. As we will see, this class encompasses well-studied models, such as Stochastic Context-Free Grammars and (Multi-type) Branching Processes.

The number of exits measures the amount of information (number of different return values) that a component returns when it terminates to the component that called it; one exit means that the component does not return any information beyond the fact that it terminated. There appears to be a distinct difference in expressiveness and complexity between 1-exit and multiexit RMCs; for example we'll show how to test in polynomial

time for almost sure termination in 1-exit RMCs, but we do not know how to do it for multi-exit RMCs.

2.2 The central reachability and termination questions.

Our focus in this paper is on answering 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 $\langle \epsilon, u \rangle$. We let $q^*_u = \sum_{ex \in Ex_i} q^*_{(u,ex)}$ be the probability that the process eventually terminates (at some exit) when started at vertex u.

Computing the termination probabilities $q^*_{(u,ex)}$ allows us to efficiently obtain other reachability probabilities, in the following sense: from a RMC A, in which we want to compute a specified reachability probability, we can construct efficiently another RMC A' such that the desired reachability probability in A is equal to a termination probability in A'. We show this for two types of reachability probability. Suppose we are given an RMC A and two vertices $u, v \in Q$ of A. The first reachability probability, denoted $Pr[u, v]$, is the probability that the RMC A started at vertex u with empty context (no pending recursive calls) will reach eventually vertex v with empty context, i.e., $Pr[u, v]$ is the probability that the infinite Markov chain M_A induced by A, started at state $\langle \epsilon, u \rangle$ will reach eventually state $\langle \epsilon, v \rangle$; it is easy to see that this probability can be nonzero only if u, v are in the same component of A. The second reachability probability, denoted $Pr'[u, v]$, is the probability that the RMC A started at vertex u with empty context will reach eventually vertex v with some context; i.e., $Pr'[u, v]$ is the probability that the infinite Markov chain M_A, started at state $\langle \epsilon, u \rangle$ will reach eventually a state of the form $\langle \beta, v \rangle$ for some $\beta \in B^*$; the vertices u, v could be from different components in this case.

Proposition 2.1 *Given a RMC A and two vertices u, v of A, we can construct in linear time two other RMCs C, C' such that the reachability probability $Pr[u, v]$ is equal to a termination probability in C, and the probability $Pr'[u, v]$ is equal to a termination probability in C'.*

Proof. For $Pr[u, v]$, we may assume that u, v are vertices in the same component A_i of A (otherwise, the probability is 0). Let C be the RMC that has all the components of A, and an additional component C_0 that is the same as A_i, except that we add self-loops with probability 1 to all the exits nodes of A_i and make them non-exit nodes, we remove all outgoing edges from v, and make v the only exit node of C_0. All the boxes of C_0 are mapped to the same components as in A_i. It is easy to see that the probability $Pr[u, v]$ in A is equal to the termination probability $q^*_{(u,v)}$ in the RMC C.

For the probability $Pr'[u, v]$, assume that u is in component A_i and v in A_j. We do the following transformation to A to obtain a new RMC C'. Add a new special exit $ex^\star_h$ to every component A_h of the RMC A. Remove the out-edges from v and instead add a probability 1 transition $v \xrightarrow{1} ex^\star_j$ to the new exit of its component. For every box b of every component A_h, add a probability 1 transition from the new return port $w = (b, ex^\star_{Y(b)})$ of b to the new exit $ex^\star_h$ of the component. Intuitively, the effect of the new exits and the new transitions is that when we encounter vertex v in any context, we "raise an exception", pop the entire call stack, and exit the system. It follows easily that $Pr[u, v]$ in the original RMC A is equal to the termination probability $q^*_{(u,ex^\star_i)}$ in the new RMC C'. ∎

The above transformation for the probability $Pr'[u, v]$ increases the number of exits by 1. We remark that there is also a more involved way to reduce it to a termination probability without increasing the number of exits as shown in [EY05a]. There we consider the more general question of model checking RMCs, and show that the termination probabilities lie at the heart of the analysis of general properties of RMCs.

Determining whether a termination probability $q^*_{(u,ex)}$ is 0, can be done efficiently (in polynomial time) using the algorithms for the analysis of (non-probabilistic) Recursive State Machines. Note that $q^*_{(u,ex)}$ is the sum of the probabilities of all the paths of M_A that start at state $\langle \epsilon, u \rangle$ and end in state $\langle \epsilon, ex \rangle$, where the probability of a path is the product of the probabilities of its edges (transitions). Thus, $q^*_{(u,ex)} = 0$ if and only if there is no such path. The actual values of the transition probabilities are clearly irrelevant in this regard, and this is simply a reachability question on the Recursive State Machine that underlies the RMC A. For RMC $A = (A_1, \ldots, A_k)$, let $\xi = \max_{i \in \{1,\ldots,k\}} |Ex_i|$ be the maximum number of exits in any component, and let $\theta = \max_{i \in \{1,\ldots,k\}} \min\{|En_i|, |Ex_i|\}$. Reachability in RSMs was studied in [ABE$^+$05], where it was shown that the problem can be decided in $O(|A|\theta^2)$ time for a given vertex-exit pair (u, ex). If the RMC has v vertices and e edges, then we can compute all the reachable vertex-exit pairs (u, ex) in time $O(e\xi + v\theta\xi)$. The theorem is stated in a slightly different way in [ABE$^+$05], but what we have stated here follows from basically the same analysis. More specifically, [ABE$^+$05] constructs in total time $O(e\theta + v\theta^2)$, a relation R_i for each component A_i of A, which contains all reachable vertex-exit pairs of A_i if $|Ex_i| \leq |En_i|$ (i.e., $R_i = \{(u, v) | u \in Q_i, v \in Ex_i, \langle \epsilon, u \rangle$ can reach $\langle \epsilon, v \rangle\}$), and R_i contains all reachable entry-vertex pairs if $|En_i| < |Ex_i|$. Once these relations are computed, [ABE$^+$05] shows that other reachability information can be computed easily by replacing every box b with directed edges from the call ports (b, en) to the exit ports (b, ex) such that $(en, ex) \in R_{Y(b)}$ and performing standard graph search in the resulting ordinary (nonrecursive) graph. If a component A_i has fewer exits than entries, then R_i gives already the information we want for this component. Otherwise, replace as above all the boxes of A_i with edges from the call ports to the reachable exit ports to get an ordinary graph G_i on the same vertex set Q_i and perform a search from the exit nodes to determine all the reachable vertex-exit pairs; if A_i has v_i vertices and e_i edges, then G_i has at most $e_i + v_i\theta$ edges, and since there are at most ξ exit nodes, the search takes time $O(e_i\xi + v_i\theta\xi)$ time. Summing over all the components, it follows that all the reachable vertex-exit pairs of the RMC can be computed in time $O(e\xi + v\theta\xi)$.

Theorem 2.2 *(see [ABE$^+$05]) Given RMC A, in time $O(|A|\theta\xi)$ we can determine for all vertex-exit pairs (u, ex), whether or not $q^*_{(u,ex)} = 0$.*

We distinguish between the qualitative (almost sure) reachability problem and the quantitative problem. For the latter problem, as we will see, in general the termination probabilities are irrational, thus we cannot compute them exactly. Therefore, we will consider two types of quantitative problems: the decision and the approximation problem. More formally, we focus here on finding efficient algorithms for the following central questions:

(1) *Qualitative termination problems*: Given an RMC A, vertex u and exit ex of the same component, is $q^*_{(u,ex)} = 1$? Is $q^*_u = 1$?

(2) *Quantitative termination problems*:
a. *Decision Problem.* Given an RMC A, vertex u and exit ex of the same component,

and a rational $r \in [0,1]$ compare $q^*_{(u,ex)}$ to r, i.e. determine whether $q^*_{(u,ex)} < r$, $= r$ or $> r$. Same question for the probability q^*_u.

b. *Approximation Problem.* Given an RMC A, vertex u and exit ex of the same component, and a number j in unary, approximate $q^*_{(u,ex)}$ to j bits of precision (i.e. compute a value that is within an additive error $\leq 2^{-j}$ of $q^*_{(u,ex)}$).

Clearly, the qualitative problem is a special case of the quantitative decision problem. Furthermore, it is easy to see that if we have an algorithm for the quantitative decision problem, then we can use it to solve the approximation problem with polynomial overhead: Simply do a binary search in the interval [0,1] to narrow down the possible range for the probability $q^*_{(u,ex)}$; after j iterations the interval of uncertainty for $q^*_{(u,ex)}$ is at most 2^{-j}.

2.3 Single-exit RMCs, Stochastic Context-Free Grammars, and Branching Processes.

A *Stochastic Context-Free Grammar* (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 nonterminal S_i, $\sum_{\langle p_j | (S_i \xrightarrow{p_j} \alpha_j) \in R \rangle} p_j = 1$. S_1 is specified as the starting nonterminal. A SCFG G generates a language $L(G) \subseteq T^*$ and associates a probability $p(\tau)$ to every terminal string τ in the language, according to the following stochastic process. Start with the starting nonterminal S_1, pick a rule with left hand side S_1 at random (according to the probabilities of the rules) and replace S_1 with the string on the right-hand side of the rule. In general, in each step we have a string $\sigma \in (V \cup T)^*$; take the leftmost nonterminal S_i in the string σ (if there is any), pick a random rule with left-hand side S_i (according to the probabilities of the rules) and replace this occurrence of S_i in σ by the right-hand side of the rule to obtain a new string σ'. The process stops only when (and if) the current string σ has only terminals. The probability $p(\tau)$ of a terminal string is the probability that the process terminates with the string τ.

According to this definition, $p(\tau)$ is the sum of the probabilities of all leftmost derivations of τ, where the probability of a (leftmost) derivation is the product of the probabilities of the rules used in the derivation. An alternative, equivalent definition is that $p(\tau)$ is the sum of the probabilities of all the parse trees of the string τ, where again the probability of the parse tree is the product of the probabilities of the rules used in the parse tree. The probability of the language $L(G)$ of the SCFG G is $p(L(G)) = \sum_{\tau \in L(G)} p(\tau)$. Note that $L(G)$ is the probability that the stochastic process that we described above, starting with S_1 terminates. More generally, we can define for each nonterminal $S_j \in V$ an associated probability $p(S_j)$, which is the probability that the process starting with S_j terminates. Note that, even though the probabilities of the rules of every nonterminal sum to 1, the probability of the language $L(G)$ may be less than 1; G is called *consistent* if $p(L(G)) = 1$.

We will present reductions between 1-exit RMCs and SCFGs, showing the following Theorem.

Theorem 2.3

1. Every SCFG G can be transformed in linear time 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

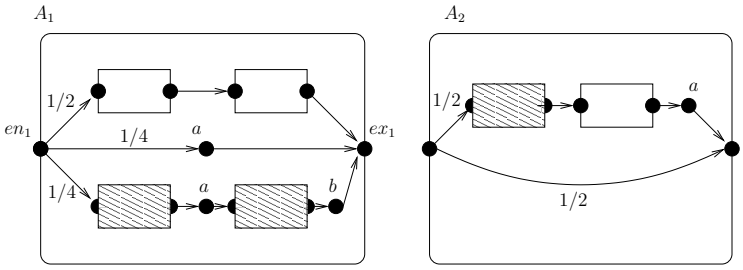

Figure 2: *RMC of a SCFG*

A, each with a single entry en_j and single exit ex_j, such that $p(S_j) = q^*_{(en_j,ex_j)}$, for all j.
2. Conversely, every 1-exit RMC A can be transformed in linear time 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 $q^*_u = p(S_u)$.

Proof. Given a SCFG G, we can define in a natural way a recursive Markov chain A whose termination probability is equal to $p(L(G))$. The RMC A has one component A_i for every nonterminal S_i of G, the component A_i has one entry en_i and one exit ex_i, and has a path from en_i to ex_i for each rule of G with left-hand-side (lhs) S_i, where the path contains a box for every nonterminal on the right-hand-side (rhs) of the rule, the first edge has probability equal to the probability of the rule and the other edges have probability 1. As an example, Figure 2.3 shows the RMC corresponding to the grammar G with nonterminals $V = \{S_1, S_2\}$, terminals $T = \{a, b\}$ and rules $R = \{S_1 \stackrel{1/2}{\to} S_1S_1,\ S_1 \stackrel{1/4}{\to} a,\ S_1 \stackrel{1/4}{\to} S_2aS_2b,\ S_2 \stackrel{1/2}{\to} S_2S_1a,\ S_2 \stackrel{1/2}{\to} \epsilon\}$. The unshaded boxes of the figure are mapped to A_1 and the shaded boxes are mapped to A_2. All edges that do not have an attached probability label have probability 1. Observe that there is a 1-to-1 correspondence between the leftmost derivations of terminal strings in G starting from S_1, and terminating paths in the RMC A starting at the entry en_1 of component A_1, i.e. paths reaching the exit ex_1 of A_1, and the correspondence obviously preserves the probabilities. Thus, the probability $q^*_{en_1} = q^*_{(en_1,ex_1)}$ of termination of the RMC A, starting at en_1, is equal to the probability $p(L(G))$ of the language of G. And generally, $p(S_i) = q^*_{(en_i,ex_i)}$ for all nonterminals S_i.

The RMCs derived from a SCFG have some special properties: all components are acyclic, have 1 entry and 1 exit. The restriction of 1 entry is not a significant one: every RMC A can be easily transformed, at polynomial cost, to another equivalent RMC whose components have 1 entry each. However, the restriction of 1 exit is significant: 1-exit RMCs are weaker than general RMCs.

There is a reverse transformation from every 1-exit RMC A (whether acyclic or cyclic) to a SCFG G, of size linear in A, such that the probability of termination of A starting at a vertex is equal to $p(L(G))$. Let A be an 1-exit RMC. We can assume also that each node of A is labelled by a letter in a terminal alphabet T or by ϵ as in the figure (for simplicity we don't label call and return ports); then the set of terminating paths starting at any vertex u defines a language $L(u)$ of terminal strings with an associated probability. The reduction is as follows. For each vertex u of A the grammar G has a nonterminal S_u. If u is the exit of its component and has label a then G contains the rule $S_u \stackrel{1}{\to} a$. If u is a call port $u = (b, en) \in Call_b$ and $v = (b, ex)$ is the return port

of the box, then G contains the rule $S_u \xrightarrow{1} S_{en} S_v$. Otherwise, i.e. if u is not an exit or a call port and thus has probabilistic outgoing edges, then for each edge $(u, p_{u,v}, v) \in \delta$, the SCFG G has a rule $S_u \xrightarrow{p_{u,v}} a S_v$, where a is the label of u. It is easy to see that there is a 1-to-1 correspondence between terminating paths of the RMC A starting at a vertex u (i.e. paths in the corresponding Markov chain M_A that start at state $\langle \epsilon, u \rangle$ and end at state $\langle \epsilon, ex \rangle$ where ex is the exit of u's component) and leftmost derivations in G of a terminal string starting from the nonterminal S_u. The correspondence preserves probabilities (the product of the probabilities of the edges on the path is equal to the product of the probabilities of the rules used in the derivation), and the concatenation of the labels of the nodes on the path is equal to the terminal string that is derived. Thus, the termination probability q_u^* is equal to the probability $p(S_u)$ of the language generated by G starting from S_u. ■

Note that, the SCFG G constructed above from a 1-exit RMC has a very special form. Thus, every SCFG and 1-exit RMC can be transformed to an equivalent SCFG in *Generalized Chomsky Normal Form* , where every nonterminal S_u has rules that come in one of three types:

- *Type*$_1$: S_u has one rule associated with it, $S_u \xrightarrow{1} a$, where $a \in T \cup \{\epsilon\}$.
- *Type*$_{rand}$: S_u has "linear" rules associated with it of the form $S_u \xrightarrow{p_{u,v}} a S_v$.
- (*Type*$_{call}$): S_u has one rule associated with it of the form: $S_u \xrightarrow{1} S_v S_w$.

We remark that it is similarly possible to define a kind of generalized normal form for arbitrary (multi-exit) RMCs, but we refrain from doing so in the interest of space.

A similar proof can be used to show that the same tight relationship holds between 1-exit RMCs and finite *Multi-Type Branching Processes* [Har63]. An MT-BP $G = (V, R)$ consists of a (finite) set $V = \{S_1, \ldots, S_k\}$ of *types*, and a (finite) set R of rules $S_i \xrightarrow{p} \alpha$, where $S_i \in V$, $p \in (0, 1]$, and α is a (finite) multi-set whose elements are in V, and such that for every type S_i, $\sum_{\langle p_j | (S_i \xrightarrow{p_j} \alpha_j) \in R \rangle} p_j = 1$. The rule $S_i \xrightarrow{p} \alpha$ specifies the probability with which an entity of type i generates the multiset α of offsprings in the next generation. The stochastic process generated by such an MT-BP is intuitively described as follows: we start with an initial set of entities of given types; we will usually start with one entity of some type S_i. In each generation we have a set of entities of various types, and from them we produce the set in the next generation as follows. For each entity in the current set, independently and simultaneously, we probabilistically choose a rule whose left hand side is the type of the entity, according to that rule's probability, and replace the entity with a new set of entities whose types are specified by the right hand side of the rule. The process continues as long as the current set of entities is not empty and terminates if and when it becomes empty.

Let $p(S_j)$ denote the probability that, starting with one entity of type S_j, the process will terminate, i.e., we will eventually reach extinction of all objects; $p(S_j)$ is called the probability of extinction of type S_j. Clearly, given these probabilities we can easily compute the probability of termination for any initial set of entities: if there are initially n_j entities of type $j = 1, \ldots, k$ then the termination probability is $\Pi_j (p(S_j))^{n_j}$.

Formally, the specification of a branching process is very similar to a SCFG, where types correspond to nonterminals. The difference is that in a branching process there are

no terminals, and the right-hand sides of the rules are multisets rather than strings, i.e. the order of the offsprings is not important. Also, in a branching process the rules are applied simultaneously to all the entities in each generation, whereas in a SCFG the rules are applied sequentially in a derivation. These differences however are clearly immaterial as far as termination of the process is concerned. If we view the branching process G as a SCFG with nonterminals V and rules R, then the extinction probability $p(S_j)$ is equal to the probability of the language generated by the grammar with starting nonterminal S_j. Clearly, the translation from a SCFG to a MT-BP where we ignore the terminals on the right-hand sides of the rules and change strings to multisets, is a polynomial (in fact, linear) translation. Conversely, if the rules of the branching process specify the multisets on the right-hand side explicitly by listing the elements as many times as they occur in the multiset, or equivalently specify the numbers of offsprings of the various types in unary notation, then the translation from a MT-BP to a SCFG, where we just change multisets to strings in the rules by listing the elements in an arbitrary order, is polynomial.

If the multiplicities of the multi-sets on the right hand sides of the rules of the MT-BP are specified in binary, in other words, if the multi-set is specified by set of pairs, (S_j, m_j) specifying each distinct type S_j in the multi-set, together with the number of occurences m_j of S_j, then this straight-forward translation is not polynomial, but we can nevertheless obtain a polynomial translation as follows. We introduce additional nonterminals. Specifically, if n_j is the maximum multiplicity of a type S_j on the right-hand side of any rule of the MT-BP, then we introduce $l_j + 1$ new nonterminals $U_{jt}, t = 0, 1, \ldots, l_j$, where $l_j = \lfloor \log n_j \rfloor$, and include in the SCFG rules $U_{j0} \to S_j$, and $U_{jt} \to U_{j,t-1}U_{j,t-1}$ for $t = 1, \ldots, l_j$ with probabilitiy 1. Note that by these rules, U_{jt} generates a string of 2^t S_j's with probability 1. For each rule of the MT-BP, we have a corresponding rule in the SCFG with the same left-hand-side and same probability; if the right-hand-side of the MT-BP rule contains a type S_j with positive multiplicity m_j that has binary representation $a_{l_j} \ldots a_1 a_0$, then the string on the right hand side of the corresponding rule in the SCFG contains a substring that includes a (single) occurrence of U_{jt} iff $a_t = 1$, for each t; clearly this substring generates with probability 1 a string of m_j S_j's. It follows then easily from the construction that the extinction probability $p(S_j)$ in the MT-BP is equal to the probability of the language generated by the grammar with starting nonterminal S_j. Combining with the translation between SCFG's and 1-exit RMCs we have:

Theorem 2.4
1. Every MT-BP G (even when the MT-BP's rules are presented by giving the multi-sets in binary representation) can be transformed in polynomial time to a 1-exit RMC A, such that there is a mapping from types S_j in G to components A_j of A, each with a single entry en_j and exit ex_j, such that $p(S_j) = q^*_{(en_j, ex_j)}$, for all j.
2. Conversely, every 1-exit RMC A can be transformed in linear time to a MT-BP G of size $O(|A|)$, such that there is a map from the vertices u of A to types S_u of G, such that $q^*_u = p(S_u)$.

3 The system of nonlinear equations associated with an RMC, and basic properties of its Least Fixed Point solution.

Given a Recursive Markov Chain A, we would like to compute the termination probabilities $q^*_{(u,ex)}$ for all pairs (u, ex) where u is a vertex of A and ex is an exit of the component that contains u. We will 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). There is one variable $x_{(u,ex)}$ for each unknown probability $q^*_{(u,ex)}$, i.e. for each vertex u and each exit ex of the component that contains u, and one polynomial $P_{(u,ex)}$ in this set of variables. We will often find it convenient to index the variables $x_{(u,ex)}$ according to an arbitrary 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 exactly one equation of the form $x_{(u,ex)} = P_{(u,ex)}(\mathbf{x})$, for each variable $x_{(u,ex)}$, where $P_{(u,ex)}(\mathbf{x})$ is a multivariate polynomial with positive rational coefficients, and is defined as follows. There are 3 cases, based on the* Type *of vertex u:*

1. Type$_1$*: $u = ex$. In this case, the equation is $x_{(ex,ex)} = 1$.*
2. Type$_{rand}$*: either $u \in N_i \setminus \{ex\}$ or $u = (b, ex')$ is a return port. In this case:*
$$x_{(u,ex)} = \sum_{\{v | (u, p_{u,v}, v) \in \delta\}} p_{u,v} \cdot x_{(v,ex)}.$$
(If u has no outgoing transitions, i.e., u is an exit other than ex then this equation is by definition $x_{(u,ex)} = 0$.)
3. Type$_{call}$*: $u = (b, en)$ 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 the system of equations $S_A = (x_{u,ex} = P_j(\mathbf{x}) \mid u \in Q_i, ex \in Ex_i, i = 1, \ldots, k)$ 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 $O(|A|\xi^2)$, where ξ denotes the maximum number of exits of any component of A. We will now identify a particular solution to the system $\mathbf{x} = P(\mathbf{x})$, 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 usual partial-order $\mathbf{x} \leq \mathbf{y}$ 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 $\mathbf{x} \leq \mathbf{y}$ then $H(\mathbf{x}) \leq H(\mathbf{y})$. 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 3.1 *Let $\mathbf{x} = P(\mathbf{x})$ be the system S_A associated with RMC A, and let $\mathbf{q}^*$ be the vector of termination probabilities of A.*

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

Proof. We prove each assertion in turn:

1. That P is monotone on $\mathbb{R}^n_{\geq 0}$ follows immediately from the fact that all coefficients in the polynomials P_j defining P are non-negative. Thus, if $\mathbf{0} \leq \mathbf{x} \leq \mathbf{y}$ then $\mathbf{0} \leq P(\mathbf{x}) \leq P(\mathbf{y})$. By induction on $k \geq 0$, $\mathbf{0} \leq \mathbf{x}^k \leq \mathbf{x}^{k+1}$.

2. By induction on $k \geq 0$. For $k = 0$: $\mathbf{x}^1 = P(\mathbf{0})$ is an n-vector where $P_{(u,ex)}(\mathbf{0}) = 1$ if $u = ex$, and $P_{(u,ex)}(\mathbf{0}) = 0$ otherwise. Hence, for each (u, ex), $\mathbf{x}^1_{(u,ex)} = \mathbf{q}^0{}_{(u,ex)}$, the probability of reaching $\langle \epsilon, ex\rangle$ from $\langle \epsilon, u\rangle$ in at most 0 steps.

   Inductively, suppose $\mathbf{q}^k \leq \mathbf{x}^{k+1}$. Consider the probability $\mathbf{q}^{k+1}_{(u,ex)}$. There are three cases, based on what type of vertex u is:

   - *Type*$_1$. If $u = ex$, then clearly $\mathbf{q}^k_{(u,e)} = \mathbf{q}^{k+1}_{(u,ex)} = 1$. Note that since $P_{(ex,ex)}(\mathbf{x}) = 1$, $\mathbf{x}^k_{(ex,ex)} = P^k_{(ex,ex)}(\mathbf{0}) = 1$, for all $k \geq 1$. Thus $\mathbf{q}^{k+1}_{(u,ex)} = \mathbf{x}^{k+2}$.
   - *Type*$_{rand}$. In this case, $\mathbf{q}^{k+1}_{(u,ex)} = \sum_v p_{u,v}\, \mathbf{q}^k_{(v,ex)}$. Thus

$$\begin{aligned}
\mathbf{x}^{k+2}_{(u,ex)} &= P_{(u,ex)}(\mathbf{x}^{k+1}) \\
&= \sum_v p_{u,v}\, \mathbf{x}^{k+1}_{(v,ex)} \\
&\geq \sum_v p_{u,v}\, \mathbf{q}^k{}_{(v,ex)} \quad \text{(by inductive hypothesis)} \\
&= (\mathbf{q}^{k+1})_{(u,ex)}
\end{aligned}$$

   - *Type*$_{call}$. In this case[2], $u = (b, en) \in \mathit{Call}_b$, and

$$\mathbf{q}^{k+1}_{(u,ex)} \leq \sum_{ex' \in Ex_{Y(b)}} \mathbf{q}^k{}_{(en,ex')} \cdot \mathbf{q}^k_{((b,ex'),ex)}$$

     To see that this inequality holds, note that in order to get from $u = (b, en)$ to ex in at most k steps, we must first get from the entry en of the component

[2]This is the only case where inequality, as opposed to equality, in the inductive hypothesis becomes necessary.

labeling box b to some exit ex' in at most some number $m \leq k$ step, and then get from that box-exit (b, ex') to ex in at most $m' \leq k$ steps, such that, $m + m' \leq k$. In the formula for the upper bound, we have relaxed the requirements and only require that each of m and m' is $\leq k$. Hence the inequality. Now, by the inductive assumption, $\mathbf{x}^{k+1} \geq \mathbf{q}^k$. Hence, using the inequality, and substituting, we get

$$\mathbf{q}^{k+1}{}_{(u,ex)} \leq \sum_{ex' \in Ex_{Y(b)}} \mathbf{x}^{k+1}_{(en,ex')} \mathbf{x}^{k+1}_{((b,ex'),ex)} = P(\mathbf{x}^{k+1})_{(u,ex)} = \mathbf{x}^{k+2}{}_{(u,ex)}.$$

We have established assertion (2).

3. Assertion (3) follows from the definition of $\mathbf{q}^*$. The equations for vertices of Type_1, Type_{rand}, and Type_{call}, can be used to define precisely the probabilities $\mathbf{q}^*{}_{(u,ex)}$ in terms of other probabilities $\mathbf{q}^*{}_{(v,ex)}$. Hence $\mathbf{q}^*$ is a fixed-point of P.

4. Note that P is monotonic, and that $\mathbf{q}^*$ is a fixed-point of P. Since $\mathbf{x}^0 = 0 \leq \mathbf{q}^*$, it follows, by induction on $k \geq 0$, that $\mathbf{x}^k \leq \mathbf{q}^*$, for all $k \geq 0$.

5. Note $\lim_{k\to\infty} \mathbf{q}^k = \mathbf{q}^*$, and $\mathbf{q}^k \leq \mathbf{x}^{k+1} \leq \mathbf{q}^*$. Thus, $\lim_{k\to\infty} \mathbf{x}^k = \mathbf{q}^*$.

6. Suppose $\mathbf{q}' \geq 0$ is a fixed-point of P. By the same argument as for $\mathbf{q}^*$, we know that for all $k \geq 0$, $\mathbf{x}^k \leq \mathbf{q}'$. But since $\lim_{k\to\infty} \mathbf{x}^k = \mathbf{q}^*$, it must be that $\mathbf{q}^* \leq \mathbf{q}'$.

■

We have thus identified $\mathbf{q}^*$ as $\text{LFP}(P) = \lim_{k\to\infty} \mathbf{x}^k$. We can view Theorem 3.1 as giving an iterative algorithm to compute $\text{LFP}(P)$, by computing the iterates $\mathbf{x}^k = P^k(\mathbf{0})$, $k \to \infty$, until we think we are "close enough". How many iterations do we need to gain k bits of precision? We show below that we need at least an exponential number.

We furthermore give several simple examples to illustrate some of the difficulties of analysing recursive Markov chains, and we point out some of the important differences between RMCs and ordinary finite Markov chains. For example, for finite Markov chains with rational transition probabilities, the reachability probabilities are rational, whereas for RMCs they are in general irrational. In the finite Markov chain case, qualitative questions, such as whether a state is reached from another state with probability 1, only depend on the structure (the edges) of the Markov chain, and not on the values of transition probabilities, whereas for RMCs they may depend on the actual values. For finite Markov chains given explicitly in the input, the reachability probabilities have polynomial bit complexity. By contrast, even in the case of hierarchical Markov chains, which have rational reachability probabilities, an exponential number of bits is required to differentiate a reachability probability from 1 or from another rational number; thus approximation to a polynomial number of bits is not sufficient to answer a decision problem or a qualitative problem. Some of these examples will be used later on as gadgets in our lower bound proofs.

Theorem 3.2 *There are RMCs with the following properties. Furthermore, all the following RMCs, except the HMCs in (4.), have one component, one entry en, and one exit ex.*

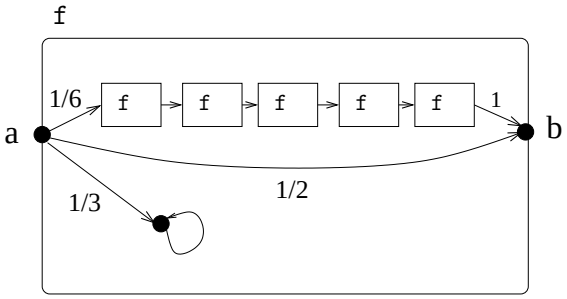

Figure 3: RMC A

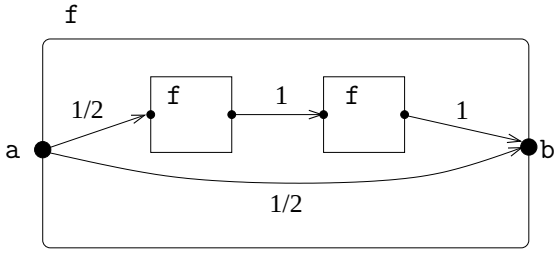

Figure 4: RMC A'

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 fixed RMC such that it requires an exponential number of iterations, $m = 2^{k-3}$, of $P^m(\mathbf{0})$ to obtain $\mathbf{q}^*$ to within 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 $\mathbf{q}^*_{(en_m,ex'_m)} = 1/(2^{2^m})$ and $\mathbf{q}^*_{(en_m,ex''_m)} = 1 - (1/(2^{2^m}))$.*

Proof.

1. Consider the RMC, A, in Figure 3. There is only one component with one entry and one exit. All boxes refer to that one component. All edges without probability labels are probability 1 edges. We can simplify the system $\mathbf{x} = P(\mathbf{x})$ for this RMC to the single equation in one variable, $x = (1/6)x^5 + 1/2$, by observing that the only probability that needs to be determined in order to determine all others is $q^*_{(a,b)}$. It can easily be shown that this probability is the LFP solution of the above equation.

Thus, LFP(P) is a root of the polynomial $(1/6)x^5 - x + (1/2)$. Multiplying this polynomial by 6, which doesn't change the roots, we get the polynomial: $q(x) = x^5 - 6x + 3$. $q(x)$ is an irreducible quintic polynomial. It is well-known that this polynomial has Galois group S_5, and hence, by Galois' theorem, all roots are irrational and not "solvable by radicals" (see, e.g., [Ste89], Theorem 14.8). ($\mathbf{q}^*_{(en,ex)}$ happens to be $\approx .50550123\ldots$.)

2. Consider the RMC A' in Figure 4. The RMC A' can again be "solved" using the simplified equation $x = (1/2)x^2 + 1/2$. The LFP solution is 1, because both roots of the polynomial are 1. Let $y_k = (1 - x_k)$ be the distance between x_k and 1, where x_k is the k-th approximant. We wish to find a recursive equation for y_k instead of x_k. Substituting into the original equation, we will see that $y_{k+1} = (1/2) * (1 - x_k^2) = y_k - (y_k^2/2)$.

   Now, suppose that for some $i \geq 0$, j is the first iteration where $y_j \leq 2^{-i}$. First, we claim that $2^{-(i+1)} < y_j \leq 2^{-i}$. This is easy to see by induction. For $i = 0$ and $i = 1$ it is obvious, since $y_0 = 1$ and $y_1 = 1/2$. For $i > 1$, if in the $j - 1$ iteration $2^{-i} < y_{j-1} \leq 2^{i-1}$, then $y_j > 2^{-i} - 2^{-(2i-1)}$. Since $i > 1$, $y_j > 2^{-i} - 2^{-(i+1)} = 2^{-(i+1)}$.

   Now, let j' be the first iteration where $y_{j'} \leq 2^{-(i+2)}$. Note that $y_{j'} > 2^{-(i+3)}$. We show that $j' \geq 2^{i-1} + j$.

   Consider the values at iterations $y_j, y_{j+1}, \ldots, y_{j'}$. In each such iteration, to compute y_{k+1}, $j \leq k < j'$ we subtract $(y_k)^2/2$ from y_k. Since $y_k \leq 2^{-i}$ in every such iteration, we know that $(y_k)^2/2 \leq 2^{-(2i+1)}$. Thus, $y_{j'} \geq y_j - (j' - j) * 2^{-(2i+1)}$. Thus $y_j - y_{j'} \leq (j' - j) * 2^{-(2i+1)}$. But since $y_{j'} \leq 2^{-(i+2)}$, and $y_j \geq 2^{-(i+1)}$, we have $2^{-(i+2)} \leq y_j - y_{j'}$, and we get $2^{-(i+2)} \leq (j' - j) * 2^{-2i-1}$. But then $j' - j \geq 2^{i-1}$. Thus, if we are within i bits of precision to 1, it will take 2^{i-1} iterations to gain 2 extra bits of precision. Thus, to gain $k + 2$ bits of precision, we will need at least 2^{k-1} iterations.

3. Consider again the RMC A' of Figure 4. Suppose we increase the probability of the edge from a to the first box-entry to $c > 1/2$, and we reduce the probability of the edge from a to b to $d < 1/2$, so that again $c + d = 1$. Our equation S for such a RMC becomes $x = cx^2 + d$. Substituting $(c + d)$ for 1, note that the roots of the polynomial $cx^2 - (c + d)x + d$ are given by:

$$\frac{(c+d) \pm \sqrt{(c+d)^2 - 4cd}}{2c} = \frac{(c+d) \pm \sqrt{(c-d)^2}}{2c} = 1 \text{ or } d/c.$$

   Since $d/c < 1$, the LFP solution is d/c. We can make d/c as small as we want by choosing d close to 0 and c close to 1, while still $(c + d) = 1$.

4. Consider the HMC, i.e., hierarchical RMC, A, (i.e., no cycles in the call graph), which has $m+1$ components $A_0, \ldots, A_m$. Figure 5 depicts A_0 and A_i, for $i > 0$. A_i, $i > 0$, has two boxes, b_1^i and b_2^i, both of which map to A_{i-1}. All edge probabilities in A_i are 1. A_0 has just two edges, each with probability 1/2, from the entry to two exits. It is easy to show by induction on i, that $q_{(en_i, ex'_i)} = 1/(2^{2^i})$, and $q^\star_{(en_i, ex''_i)} = 1 - (1/(2^{2^i}))$. Note that $|A| \in O(m)$.[3]

[3]The probability $\frac{1}{2^{2^m}}$ can easily be obtained with a 1-exit HMC of size $O(m)$, by a minor modification

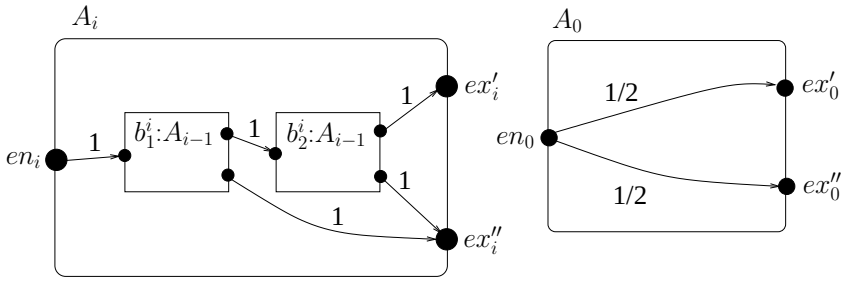

Figure 5: Hierarchical RMC, with very high & low probabilities.

■

4 Upper bounds: RMCs and the Existential Theory of Reals

We now show that the central reachability questions for general RMCs can be answered by appealing to algorithms for deciding the *Existential Theory of the Reals*, $\boldsymbol{ExTh}(\mathbb{R})$. A sentence in $\boldsymbol{ExTh}(\mathbb{R})$ is a prenex form sentence: $\exists x_1, \ldots, x_n R(x_1, \ldots, x_n)$, 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 [Can88] (see also [Ren92, BPR96, BPR03]). 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. More specifically, the following is known.[4]

Theorem 4.1 *([Can88, Ren92, BPR96]) Let φ be an existential first-order sentence in prenex form in $\boldsymbol{ExTh}(\mathbb{R})$, which uses m distinct polynomials f_i in atomic predicates, over n variables $x_1, \ldots, x_n$, each f_i having degree $\leq d$, and with all rational coefficients in φ describable in at most L bits.*
There is an algorithm that decides whether φ is true over the real numbers, and that runs in PSPACE and in time: $O(\ (L \log L \log\log L)\ (md)^{O(n)}\ )$.

Suppose we want to decide whether a nonnegative rational vector $\mathbf{c} = [c_1, \ldots, c_n]^T$ is $LFP(P)$. Consider the sentence: $\varphi \equiv \exists x_1, \ldots, x_n \bigwedge_{i=1}^n P_i(x_1, \ldots, x_n) = x_i \wedge \bigwedge_{i=1}^n x_i = c_i$.

of the construction. However, we note without proof here that it is impossible to design a 1-exit RMC of size $O(m)$ such that some probability $\mathbf{q}^*_{(u,ex)} = 1 - \frac{1}{2^{2^m}}$. Thus, ≥ 2 exits are required to get probabilities "exponentially close" (but not equal) to 1 with HMCs or RMCs.

[4]For simplicity, we assume in the statement of Theorem 4.1 that given truth values for the atomic predicates, the truth of the boolean combination R can be evaluated in constant time. This assumption does not have a significant effect on the running times we state, but serves only to simplify the statement.

The sentence φ is true iff $\mathbf{c} = P(\mathbf{c})$. To guarantee that $\mathbf{c} = \mathrm{LFP}(P)$, we additionally need to check: $\psi \equiv \exists x_1, \ldots, x_n \bigwedge_{i=1}^{n} P_i(x_1, \ldots, x_n) = x_i \wedge \bigwedge_{i=1}^{n} 0 \leq x_i \wedge \bigvee_{i=1}^{n} x_i < c_i$. ψ is false iff there is no solution $\mathbf{z} \in \mathbb{R}^n_{\geq 0}$ to $\mathbf{x} = P(\mathbf{x})$ such that $\mathbf{c} \not\leq \mathbf{z}$. Hence, to decide whether $\mathbf{c} = \mathrm{LFP}(P)$, 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 $\mathbf{c} = \mathrm{LFP}(P)$.

If we want to determine whether a particular probability q_k^* is smaller than a given rational number c_k then we form the sentence $\varphi_k \equiv \exists x_1, \ldots, x_n \bigwedge_{i=1}^{n} P_i(x_1, \ldots, x_n) = x_i \wedge \bigwedge_{i=1}^{n} 0 \leq x_i \wedge x_k < c_k$. This sentence says that the system $x = P(x)$ has a nonnegative solution with the property that $x_k < c_k$. Clearly, this is the case if and only if the LFP has this property. Similarly we can test if $q_k^* \leq c_k$ or not (i.e. $q_k^* > c_k$).

Theorem 4.2 *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 $\mathbf{q}_k^* \, \Delta c_k$, for any comparison operator Δ. Moreover, the running time of the algorithm is $O(|A|^{O(n)})$ 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.*

$\boldsymbol{ExTh}(\mathbb{R})$ gives us a way to ask questions like: "Is there a solution to $\mathbf{x} = P(\mathbf{x})$ where $a \leq x_k \leq b$?" for any rational numbers a and b, and if we wish, with either inequality replaced by strict inequality. Since $\mathbf{0} \leq \mathrm{LFP}(P) \leq \mathbf{1}$, 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 φ_k we can gain one extra bit of precision on the exact value of q_k^* with each extra query to $\boldsymbol{ExTh}(\mathbb{R})$. So, if we want j bits of precision for each q_k^*, $k = 1, ...n$, 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 4.3 *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 $O(j \cdot |A|^{O(n)})$, where n is the number of variables in $\mathbf{x}$.*

5 "Lower" bounds: RMCs, the Square-Root Sum Problem, and arithmetic circuit decision problems

The last section showed that problems related to the termination probability of RMCs can be decided in PSPACE. In this section we show that any substantial improvement of those PSPACE upper bounds will have to overcome major obstacles. Namely, we show that even approximating the termination probability of a given RMC to within any nontrivial additive factor is at least as hard as a long standing open problem in the complexity of exact numerical computation, the *square-root sum problem* (`SQRT-SUM`), and an even more difficult (and fundamental) problem on arithmetic decision circuits.

Formally `SQRT-SUM` is the following problem: given natural numbers $(d_1, \ldots, d_n) \in \mathbb{N}^n$ and another number $k \in \mathbb{N}$, decide whether $\sum_{i=1}^{n} \sqrt{d_i} \leq k$. The `PosSLP` (positive Straight-Line Program) decision problem asks whether a given a straight-line program or, equivalently, arithmetic circuit with operations $+, -, *$, and inputs 0 and 1, and a designated output gate, outputs a positive integer or not. In the introduction, we gave

some background on the significance and the current status of these problems. Recall that both problems can be solved in PSPACE, and their complexity was recently lowered slightly to the 4th level of the *Counting Hierarchy*, an analog of the polynomial-time hierarchy for counting classes like $\#P$.

We show in this section that these problem reduce to the qualitative and quantitative termination problems for RMCs. First we give a simple direct reduction from the `SQRT-SUM` problem to the quantitative decision problem for 1-exit RMC (equivalently, SCFG); we include this reduction since it is quite simple. We then show that the `SQRT-SUM` and `PosSLP` problems are reducible to the problem of distinguishing for a given 2-exit RMC between the case that the RMC terminates with probability 1 and the case that it terminates with probability $\leq \epsilon$ where ϵ is any positive constant. This means that the qualitative termination problem (is the termination probabiility 1), and approximation to within any nontrivial error for 2-exit RMCs are as hard as `SQRT-SUM` and `PosSLP`. Finally, we show that these problems reduce also to the quantitative decision problem for 1-exit hierarchical RMCs. (As we will show later, the qualitative termination problem can be solved in polynomial time for both, the class of 1-exit and the class of hierarchical RMCs).

Let SCFG-DEC be the following decision problem: given a SCFG G (with rational rule probabilities) and given a rational number $p \in [0, 1]$, decide whether the language generated by the SCFG has probability $\geq p$ (i.e., G produces a terminal string with probability $\geq p$). From Theorems 2.3 and 2.4, this problem is equivalent to the corresponding decision problems, denoted 1-EXIT-DEC (respectively, MT-BP-DEC) of determining for a given 1-exit RMC (resp. MT-BP) and rational p, whether the termination probability of the RMC starting at a given vertex (resp. the extinction probability of a given type in the MT-BP) is $\geq p$.

Theorem 5.1 `SQRT-SUM` *is P-time reducible to SCFG-DEC, 1-EXIT-DEC and MT-BP-DEC.*

Proof. It suffices to prove it for SCFG-DEC. Given $(d_1, \ldots, d_n) \in \mathbb{N}^n$ and $k \in \mathbb{N}$, we construct an SCFG, G, as follows. Let $m = \max_{1 \leq i \leq n} d_i$. Our SCFG G, will have non-terminals $\{S, S_1, \ldots, S_n\}$, and a single terminal $\{z\}$. S is the start non-terminal. The production rules associated with S are $S \xrightarrow{1/n} S_i$, for $i = 1, \ldots n$. Let $c_i = (1 - (d_i/m^2))/2$.

There are two productions associated with each S_i:

$$\begin{array}{lll} S_i & \xrightarrow{1/2} & S_i S_i \\ S_i & \xrightarrow{c_i} & z \end{array}$$

Note that $1/2 + c_i$ need not necessarily sum to exactly 1. If desired, we can make this a "proper" SCFG, where production probabilities out of each non-terminal sum to 1, by adding an extra production $S_i \to A$, with the residual probability $1 - (1/2 + c_i)$, such that A is a new "dead end" non-terminal, with only one production $A \to A$, having probability 1.

For a non-terminal N, let p_N be the probability that N terminates. Using the standard formula for roots of quadratic polynomials, we see that:

$$p_S = \sum_{i=1}^{n}(1/n)p_{S_i} = (1/n)\sum_{i=1}^{n}(1-\sqrt{(1-2c_i)})$$
$$= 1-(1/n)\sum_{i=1}^{n}\sqrt{(d_i/m^2)} = 1-(\,(1/(nm))\sum_{i=1}^{n}\sqrt{d_i})$$

Thus, $\sum_{i=1}^{n}\sqrt{d_i} \leq k$ if and only if $p_S \geq (1-k/(nm))$. ■

We next show that the qualitative decision problem as well as any non-trivial approximation of the termination probabilities for 2-exit RMCs, are both SQRT-SUM-hard and PosSLP-hard. Specifically, let the *Promised Gap Decision Problem*, PGD(a,b), be the following: Given an RMC, vertex u and exit ex, and rationals $a < b$, and the promise that the termination probability $q^*_{u,ex}$ has the property that either $q^*_{u,ex} \leq a$ or $q^*_{u,ex} \geq b$, decide which of the two is the case.

Theorem 5.2 *For every $\epsilon > 0$, the* SQRT-SUM *and* PosSLP *problems are P-time (many-one) reducible to the promised gap problem PGD(ϵ,1) for the termination probability of 2-exit RMCs.*

Proof. Here we prove PosSLP-hardness. It follows from [ABKPM06] that SQRT-SUM is also (Turing) reducible to the PGD(ϵ,1) problem for RMCs. A direct reduction from SQRT-SUM, which we omit due to space, shows that SQRT-SUM is many-one reducible to PGD(ϵ,1) for RMCs.

We are given an arithmetic circuit C over basis $\{+,-,*\}$ with fan-in 2 and with inputs 0,1, and we want to determine whether the output of the circuit, denoted $val(C)$, is positive. First, we observe that in the PosSLP problem we can assume, w.l.o.g., that the only use of a subtraction gate occurs at the topmost gate of the arithmetic circuit. We can transform any arithmetic circuit over basis $\{+,-,*\}$ to this form by replacing each gate g of the original circuit with two gates g^+ and g^- for the positive and negative parts, such that $g = (g^+ - g^-)$. Viewing the transformation bottom-up, each addition gate $g_k := g_i + g_j$ in the original circuit can be replaced by two gates: $g_k^+ := g_i^+ + g_j^+$ and $g_k^- := g_i^- + g_j^-$. Likewise, a subtraction gate $g_k := g_i - g_j$ can be replaced by $g_k^+ := g_i^+ + g_j^-$ and $g_k^- := g_i^- + g_j^+$. Finally, a multiplication gate $g_k := g_i * g_k$ can be replaced by $g_k^+ := g_i^+ * g_j^+ + g_i^- * g_j^-$ and $g_k^- := g_i^+ * g_j^- + g_i^- * g_j^+$ (note that we need two multiplication gates and one addition gate for each of these). Clearly, the transformation only blows up the circuit linearly. For the output gate, g_{out}, we can add a subtraction gate $g_{out} := g_{out}^+ - g_{out}^-$, thus computing the same output as the original circuit.

Thus, the PosSLP problem is equivalent to the following problem: given two monotone arithmetic circuits (SLPs), S_1 and S_2, over $\{+,*\}$ with inputs 0 or 1, determine whether $val(S_1) > val(S_2)$, where $val(S_1), val(S_2)$ are the output values computed by the two circuits.

We shall define two 2-exit RMCs, A_1 and A_2, such that the probability of terminating at exit 1 of A_i is $val(S_i)/M$ and the probability of terminating at exit 2 is $(1-(val(S_i)/M))$, where M is sufficiently large such that these define legitimate probabilities.

We first need a normal form for the arithmetic circuits. We can assume, wlog, that the circuits S_i have the following normal form:

1. the depth of both circuits S_i is the same number, say k. We can do this by inserting dummy gates, which may have only 1 incoming edge.

2. The circuits are "leveled", and they alternate between a $+$ level and $*$ level. We can again do this by inserting dummy gates.

3. We can assume furthermore that each gate has actually two incoming edges (both incoming edges can be from the same gate at the lower level). For the $+$ gates, we do this by carrying to the i'th level a gate $\texttt{zero}_i$, whose value is 0, and adding a second incoming edge from $\texttt{zero}_i$ to any $+$-gate at level $i+1$ that had only 1 incoming edge before. (Note that $\texttt{zero}_i$ itself can be built with alternating levels of $+$ and $*$ gates starting from the 0 input.) For the $*$ gates, we do this by carrying to the i'th level a gate $\texttt{one}_i$, whose value is 1, and adding a second incoming edge from $\texttt{one}_i$ to any $*$-gate at level $i+1$ that had only one incoming edge before. We can easily build $\texttt{one}_i$ itself using alternating $+$ and $*$ gates starting from 1 input and using the $\texttt{zero}_i$ gates.

Given circuits S_1 and S_2 in this normal form, we will construct corresponding RMCs A_1, A_2. Each RMC contains one component B_i for every gate g_i of the corresponding circuit; every component has one entry and two exits.

We proceed bottom up in each circuit, as follows. For each $+$ gate of the form $g_i = g_j + g_k$, we include a component B_i in the RMC, whose entry has a $1/2$ transition to a box labeled by component B_j and a $1/2$ transition to a box labeled by component B_k. From return port 1 of both the boxes labeled B_j and B_k, we go with probability 1 to the first exit of component B_i, and from the second return port of B_j and B_k we go with probability 1 to the second exit of B_i. For each $*$-gate $g_i = g_j * g_k$, we include a component B_i in the RMC: whose entry transitions with probability 1 to a sequence of two boxes labeled by B_j and B_k. From return port 1 of the box labeled by B_j we go to the call port of box B_k with probability 1. From the second return port of box B_j we go to the second exit of B_i with probability 1. From the first return port of box B_k we go to the first exit of B_i with probability 1; from the second return port of box B_k we go to the second exit of B_i with probability 1.

Let g_m be a gate at level r. It is easy to prove by induction that the probability, starting at the entry of B_m, of termination at exit 1 of B_m is $val(g_m)/M_r$, and the probability of termination at exit 2 is $1-(val(g_m))/M_r)$ where $M_r = 2^{a_r}$ and the exponent a_r is defined by the recurrence $a_r = 2a_{r-1}$ if level $r \geq 1$ consists of $*$ gates, $a_r = a_{r-1}+1$ if level $r \geq 1$ consists of $+$ gates, with $a_0 = 0$ (corresponding to the inputs considered as being at level 0). Thus, if we assume wlog that odd levels of the circuits consist of $*$ gates and even levels of $+$ gates, then $a_{2i-1} = 2^i - 2$ and $a_{2i} = 2^i - 1$ for all $i \geq 1$.

Let B_{m_1} and B_{m_2} be the components associated with the output gates of S_1 and S_2, respectively. We will use the components B_{m_1}, B_{m_2} to construct a new component A with one entry and one exit, such that the termination probability of A is 1 if $val(S_2) \geq val(S_1)$ and otherwise (i.e., if $val(S_1) > val(S_2)$) its termination probability is $\leq (M-1)/M < 1$, where $M = M_k = 2^{a_k}$ is the denominator corresponding to the common depth k of the circuits S_1 and S_2. The component A is based on the RMC of part 3 of Theorem 3.2, i.e., the RMC depicted in Figure 4, where the transitions from the entry node have probabilities c and d instead of $1/2$. Our component A here has one entry, one exit, and four boxes b_1, b_2, b_3, b_4. Boxes b_3, b_4 are both labeled by A and are connected sequentially

in the same way as the two boxes labeled f in Figure 4: there is a probability 1 transition from the return port of b_3 to the call port of b_4 and a probability 1 edge from the return port of b_4 to the exit of A. The two boxes b_1 and b_2 are labeled by B_{m_1} and B_{m_2}, respectively. From the entry of A there is $1/2$ probability transition to the call ports of both b_1 and b_2. There are probability 1 edges from return port 1 of b_1 and return port 2 of b_2 to the call port of box b_3, and probability 1 edges from the other return ports of b_1 and b_2, i.e., return port 2 of b_1 and return 1 of b_2, to the exit of A. The component A starting from the entry node, will transition to one of the boxes b_1, b_2 and after that box returns, it will either transition to the call port of b_3, or to the exit; let c be the probability of the first event and $d = 1 - c$ the probability of the second. Then c is $1/2$ times the probability that B_{m_1} terminates at exit 1 and the probability that B_{m_2} terminates at exit 2, thus, $c = 1/2[(val(S_1)/M) + 1 - (val(S_2))/M)] = 1/2 + (val(S_1) - val(S_2))/2M$. The probability $d = 1 - c$ is $d = 1/2 - (val(S_1) - val(S_2))/2M$. Thus, A is in effect equivalent to the RMC analyzed in part 3 of Theorem 3.2 with the above values c, d for the transition probabilities of the entry node. From that analysis we know that the termination probability of A is 1 if $c \leq 1/2$, and d/c if $c > 1/2$. Thus, the termination probability of A is 1 if $val(S_1) \leq val(S_2)$, and it is $\frac{M-val(S_1)+val(S_2)}{M-val(S_2)+val(S_1)} \leq \frac{M-1}{M} < 1$ if $val(S_1) > val(S_2)$.

We will now use the component A in order to construct another RMC. From part 4 of Theorem 3.2, we know how to construct a (hierarchical) RMC, C, with n levels such that the top component of C has two exits, and the termination probability at exit 1 is $b = 1/2^{2^n}$, and at exit 2 it is $1 - b$. Construct an RMC G whose top component contains an A box and a C box. We take n associated with the C box to be much larger than the depth k of the circuits, specifically, $n = c * k$, for some constant c, such that $b = 1/2^{2^n} < \epsilon/M$; since $M < 2^{2^k}$ it suffices to take for example $c \geq \log\log(1/\epsilon)$. The entry of G goes to the entry of the C box with probability 1. Exit 1 of the C box goes to the exit of the G component, and exit 2 goes to the entry of the A box. The exit of the A box goes back to the entry of the C box. Let z be the probability of termination for G. If the termination probability of A is 1, then $z = 1$. If it is $1 - a$, then z satisfies: $z = b + (1 - b)(1 - a)z$. So $z(a + b - ab) = b$. Therefore $z \leq b/a < \epsilon$, because $a \geq 1/M$. ■

We now show hardness of the decision problem for hierarchical 1-exit RMCs.

Theorem 5.3 *The* `PosSLP` *problem (*`SQRT-SUM` *problem) is P-time many-one (Turing, respectively) reducible to the decision problem for hierarchical 1-exit RMCs, i.e. determining whether the termination probability is greater than p for a given rational* $p \in (0, 1)$.

Proof. Since `SQRT-SUM` is P-time Turing reducible to `PosSLP` ([ABKPM06]), it suffices to provide the claimed reduction from `PosSLP`. As in the proof of the previous theorem, given a circuit C over basis $\{+, -, *\}$ with inputs 0,1, we first construct two circuits S_1, S_2 over $\{+, *\}$ such that $val(C) = val(S_1) - val(S_2)$, and furthermore the circuits S_1, S_2 have the same depth and are in the same normal form: each level has gates of the same type ($*$ or $+$) with inputs from the immediately previous level, and the levels alternate. Let c be any positive rational constant < 1, for example $c = 1/2$. We will construct a hierarchical RMC which contains two components B_i, B_i' for each gate g_i of the circuits S_1, S_2, each component has one entry and one exit, and the termination probability of (the entry node of) B_i is $p_i = \alpha_r val(g_i)$, where α_r depends on the depth r of the gate

g_i, and the termination probability of B'_i is $p'_i = c - p_i$. Then we will add one more component on top of these to get the result.

We perform the construction bottom-up in the circuits. To begin with, we consider the inputs 1 and 0 as being 'gates' g_{-1} and g_0 at level 0, let $\alpha_0 = c$, and construct trivial components $B_0 = B'_{-1}$ with termination probability 0, and $B'_0 = B_{-1}$ with termination probability c. Consider a level $r \geq 1$ consisting of + gates, and define $\alpha_r = \alpha_{r-1}/2$. For an addition gate $g_i = g_j + g_k$ at level r we include a component B_i in the RMC, whose entry has a probability 1/2 transition to a box labeled by component B_j and a 1/2 transition to a box labeled by component B_k; the return ports of both boxes have probability 1 transitions to the exit of B_i. Clearly the termination probability of B_i is $p_i = (p_j + p_k)/2 = \alpha_{r-1}(val(g(j) + g(k))/2 = \alpha_r val(g_i)$. We include also a component B'_i which is similar to B_i except that its two boxes are mapped to B'_j and B'_k. Its termination probability is $p'_i = (p'_j + p'_k)/2 = (c - p_j + c - p_k)/2 = c - p_i$.

Consider a level r consisting of $*$ gates, and let $g_i = g_j * g_k$ be a gate at level r. Let $\rho = (1 - c)/(2 - c^2)$; note that $0 < \rho < 1/2$. Define $\alpha_r = \rho(\alpha_{r-1})^2$. The component B_i corrsponding to the gate g_i has two boxes b_1, b_2 that are labeled B_j, B_k and are connected in series . The entry of B_i transitions with probability ρ to the call port of b_1 and with the remaining probability $1 - \rho$ to a dead state. There is a probability 1 transition from the return port of b_1 to the call port of b_2 and from the return port of b_2 to the exit of B_i. The termination probability of B_i is clearly $p_i = \rho p_j p_k = \rho\alpha_{r-1} val(g_j)\alpha_{r-1} val(g_k) = \alpha_r val(g_i)$.

The component B'_i for the $*$ gate is a little more complex. It has four boxes b_1, b'_1, b_2, b'_2 mapped respectively to B_j, B'_j, B_k, B'_k. The entry of B'_i has transitions with probability ρ to the call ports of b'_1 and b'_2 and a transition with the remaining probability $1 - 2\rho$ to the exit of the component. The return port of b'_1 transitions with probability 1/2 to the call port of box b_2, with probability $c/2$ to the exit and with the remaining probability $(1 - c)/2$ to a dead state. The return port of box b'_2 has the symmetric transitions: probability 1/2 to the call port of b_1, probability $c/2$ to the exit and $(1 - c)/2$ to the dead state. The return ports of both boxes b_1, b_2 have probability 1 transitions to the exit. We can calculate the termination probability p'_i of the entry node of B'_i. It is $p'_i = \rho(c - p_j)(p_k + c)/2 + \rho(c - p_k)(p_j + c)/2 + (1 - 2\rho) = \rho c^2 - \rho p_j p_k + 1 - 2\rho$. Since $\rho(c^2 - 2) = c - 1$, we have $p'_i = c - \rho p_j p_k = c - p_i$.

Let g_{m_1} and g_{m_2} be the output gates of S_1 and S_2, respectively, both at the same depth k. We add a top component C to our hierarchical Markov chain. The component C has two boxes b_1, b_2, mapped respectively to the components B_{m_1} and B'_{m_2}. The entry of C has probability 1/2 transitions to the call ports of the two boxes, and the return ports of both boxes have probability 1 transitions to the exit of C. The termination probability of C is $p_C = 1/2(p_{m_1} + c - p_{m_2}) = c/2 + (\alpha_k/2)(val(S_1) - val(S_2))$. Thus, $p_C > c/2$ iff $val(S_1) > val(S_2)$. ■

6 Numerical algorithms: RMCs and Newton's method

This section approaches efficient numerical computation of 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 3.2, part 2, (see Figure 4) where the standard iterative algorithm, $P^k(\mathbf{0})$, $k \to \infty$, converges exponentially slowly, requiring roughly 2^i iterations to gain i bits of precision, one can easily show that Newton's method converges exponentially faster, gaining one bit of precision per iteration (we will make this observation more precise later). 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, [SB93] and [OR70]). 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

$$F'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} \cdots \frac{\partial F_1}{\partial x_n} \\ \vdots \vdots \vdots \\ \frac{\partial F_n}{\partial x_1} \cdots \frac{\partial F_n}{\partial x_n} \end{bmatrix}$$

In other words, for each $\mathbf{c} \in \mathbb{R}^n$, $F'(\mathbf{c})$ is a real-valued matrix whose (i, j) entry is the function $\frac{\partial F_i}{\partial x_j}$ 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. 3.2 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, after some preprocessing and using a suitable decomposition of the system.

We first preprocess the system (in polynomial time in its size, by Theorem 2.2) 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 call G the *dependency graph* of the RMC A and the system S_A. 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.6. Namely, let $C_1, \ldots, C_m$ be the strongly connected components of G, and let H be the DAG whose nodes are the SCCs and whose edges are (C_i, C_j), $i \neq j$, iff there is an edge in G from some vertex in C_i to some vertex in C_j. Assume $C_1, \ldots, C_m$ are topologically sorted, so an edge (C_j, C_k) implies $k > j$. Note that the only scc with no outgoing edges is the scc that consists of the constant node 1, because we have eliminated all the variables that have value 0: if there was another scc with no outgoing edges, then all the variables in it will have value 0 in the least fixed point. We can obtain, a sequence of systems $\langle S^1_A, \ldots, S^m_A \rangle$ from these SCCs as follows: If we fix an assignment of values to variables in each C_k, $k > j$, then C_j can be seen as a system of equations S^j_A involving only the variables in C_j. Each such system has a LFP, which we write as LFP(S^j_A). This corresponds to a "piece" of the overall solution, i.e., there is an easy 1-1 correspondence between the combined variables of S^j_A, $j = 1, \ldots, m$, and the coordinates of $\mathbf{x} = P(\mathbf{x})$. The algorithm in Fig.6 specifies how we process SCCs and apply Newton's method to

1. Preprocess $\mathbf{x} = P(\mathbf{x})$, eliminating all variables x_i where $q_i^* = 0$;
2. Construct the DAG of SCCs, H, based on the remaining system of equations.
3. While (there is a sink SCC, C, remaining in the DAG H)
   (a) If C is the trivial SCC, $C = \{1\}$, then assign value 1 to the node in C.
   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.
      i. Stop if a fixed point is reached and assign the variables in C the values of the fixed point.
      ii. Stop after a specified number of iterations, k, or if the solutions for C are considered "good enough". Assign these final values to the variables in C and substitute these values for those variables in all remaining equations.
   (b) remove C from the DAG.

Figure 6: Decomposed Newton's method

the decomposed system. In Fig.6 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.[5]

Theorem 6.1 *Given an RMC with associated system of equations, $x = P(x)$, in the Decomposed Newton's Method of Fig. 6, the sequence $\mathbf{x}_k, k \to \infty$, monotonically converges to $\mathbf{q}^*$. Moreover, for all $k \geq 0$, $\mathbf{x}_k \geq P^k(\mathbf{0})$.*

For convenience, we will often avoid boldface and use x for $\mathbf{x}$.

Proof. We will show by bottom-up induction on the SCCs that the algorithm is well-defined (i.e. does no attempt to invert a singular matrix) and the vector x_k computed by performing k iterations on each SCC lies between $P^k(0)$ and q^*. This implies then that $x_k \to q^*$ as $k \to \infty$ since $P^k(0) \to q^*$.

Clearly the above statement is true at the beginning after we eliminate (assign 0 to) all variables x_i with $q_i^* = 0$ and we process the bottom SCC with the singleton 1. Suppose that we are now processing SCC C_j after having processed all the successor SCCs. If C_j is a trivial SCC, i.e. it contains a single node x_i and no edges, which means that x_i depends only on variables in lower SCCs, then the statement follows trivially from the induction hypothesis. Suppose that C_j is nontrivial, that is, the corresponding system of equations S_A^j is recursive. Since S_A^j is obtained from the corresponding equations in the original system S_A by substituting in the right hand side (rhs) the values obtained for

[5]We note that Theorem 6.1 is related to the *Monotone Newton Theorem* (MNT) in [OR70] (their Theorem 13.3.4) where sufficient conditions are given under which monotone convergence of Newton's method holds. However, Theorem 6.1 does not follow from MNT. This is because in our setting, without decomposing our system into SCCs, and in particular removing all 0 variables, the Jacobian $F'(\mathbf{x})$ may not be invertible and thus Newton iterates may not even be well defined, and furthermore because MNT requires that the Jacobian $F'(\mathbf{x})$ be invertible even at the solution itself, in our case at the LFP. As we shall see, this does not hold for us even on individual SCCs, and it causes considerable complications in our proofs.

the variables in lower SCCs, and these values are less than or equal to the corresponding values in $q^* = \text{LFP}(S_A)$, it follows that S_A^j has a least fixpoint $\text{LFP}(S_A^j)$ and it is less than or equal the projection of q^* on the variables in C_j. On the other hand, if we start with the 0 vector for the variables in C_j and apply to it, k times, the operator on the right hand side of S_A^j , then the vector x^k that we will obtain is greater than or equal to the projection of $P^k(0)$ on these variables, because by induction hypothesis the values that we substituted for the variables in lower SCCs have this property. We are going to show that when we perform k iterations of Newton's method on the system S_A^j, we do not encounter a singular matrix, and the computed vector lies between x^k and $\text{LFP}(S_A^j)$, which implies the statement for the induction step. In fact we will show in particular that if S_A^j is linear, then one iteration of Newton yields $\text{LFP}(S_A^j)$, and if S_A^j is nonlinear (i.e. the rhs of some equation contains a nonlinear term) then the computed vector is strictly smaller than $\text{LFP}(S_A^j)$ in all coordinates.

Thus, for the rest of the proof we will restrict attention to one SCC C_j, after having processed the lower SCCs. To simplify notation and avoid too many subscripts, we we will still use $x = P(x)$ to denote the system S_A^j of the current SCC C_j, use q^* to denote its least fixpoint $\text{LFP}(S_A^j)$, use x^k to denote the vector obtained after k iterations of Newton on S_A^j, and $x^i = P^i(0)$ to denote the vector obtained after i iterations of the rhs operator P of the system S_A^j on the 0 vector.

For $F(x) = P(x) - x$, the Newton iterates are defined by:

$$\begin{aligned} x_{k+1} &= x_k - (F'(x_k))^{-1}F(x_k) \\ &= x_k + (-F'(x_k))^{-1}(P(x_k) - x_k) \end{aligned}$$

We have to show several things. First, we have to show the sequence $x_0, x_1, \ldots$ is well defined. In other words, we have to show that the inverse matrix $(-F'(x_k))^{-1}$ exists for each x_k. We can break things into two cases: In the first case the system S_A^j for the SCC C_j being solved is a linear system; for example this is the case if all the vertices of the SCC are of type *rand* and thus correspond to an ordinary finite Markov chain, but the system S_A^j for a SCC may be linear even if there are some Type_{call} variable because some of the variables on the right hand side of their equations belong to lower SCCs and thus we have substituted their values. In any case, if S_A^j is a linear system then $-F'(x_0) = -F'(x_1) = \ldots$ is a constant matrix $I - B$, where B is the matrix of coefficients on the rhs. It will turn out in this case that $(I - B)$ is invertible and that $x_1 = q^*$.

In the second case, the system S_A^j contains at least one nonlinear term. In this case we will show that $x_0 \leq x_1 \leq \ldots$, that $x_k < q^*$ (i.e., strictly less in every coordinate) and that $(-F'(x_k))^{-1}$ exists for all $k \geq 0$, and $\lim_{k\to\infty} x_k = \mathbf{q}^*$.

We will see that the Newton iterates grow at least as fast as the iterates in the standard LFP algorithm.

Let us now consider $F'(\mathbf{x})$ in more detail. Recall that the (i, j) entry of the Jacobian matrix F' is $a_{i,j} = \frac{\partial F_i}{\partial x_j}$. Depending on the type (1, *rand* or *call*) of the equation $x_i = P_i(\mathbf{x})$ from S_A^j, we have the following cases:

- Type_1. This cannot happen for a nontrivial SCC, because in this case the SCC would contain just a single node x_i with an edge to the 1 node.
- Type_{rand}. For $i \neq j$, then $a_{i,j} = p_{i,j}$, where $p_{i,j}$ is the coefficient of x_j in P_i ($p_{i,j}$ could be 0). $a_{i,i} = (p_{i,i} - 1)$, where $p_{i,i}$ is the coefficient of x_i in P_i. Note that the

equation for x_i in the original system S_A may contain some variables that belong to lower SCCs, and which are replaced by constants in S_A^j.

- *Type$_{call}$*. Then $P_i(\mathbf{x})$ has the form $P_i(\mathbf{x}) = \sum_{k=1,\ldots,l} x_{i_k} * x_{j_k} + \sum_{k=l+1,\ldots,m} p_{i,j_k} x_{j_k} + c$. In the original system S_A the rhs of the equation is a sum of binomials; however some of the variables may belong to lower SCCs, hence when we substitute their computed values we may obtain some linear terms and a constant term.

  In this case $a_{i,i} = -1$, since we can assume that x_i itself does not appear in P_i for equations of type *call*. This is because a call port can be assumed to never be a component entry, and also never a return port. For $i \neq j$, $a_{i,j}$ is either x_{i_k}, if $j = j_k$, for some $k \leq l$, or is x_{j_k} if $j = i_k$ for some $k \leq l$, or a number $p_{i,j_k} \in [0,1]$ if $j = j_k$ for some $k \leq l$, and is otherwise 0.

Thus, we note that $F'(x)$ has a particular form. Namely:

$$F'(x) = B(x) - I$$

where I is the identity matrix, and the matrix $B(x)$ is a non-negative matrix whose (i,j) entry $b_{i,j}$ is either a probability $p_{i,j}$ which is the coefficient of x_j in the polynomial P_i or else is x_r, for some index r.

For $x \geq 0$, $-F'(x) = I - B(x)$, where $B(x)$ is a non-negative matrix. We use the following fact from matrix analysis:

Theorem 6.2 *(see, e.g., [LT85], p. 531) Let $B \in R^{n \times n}$ with $B \geq 0$. The matrix $I - B$ is invertible (non-singular) and $(I-B)^{-1} \geq 0$ if and only if $\rho(B) < 1$, where $\rho(B)$ is the spectral radius of B. If this is the case then,*

$$(I-B)^{-1} = (I + B + B^2 + \ldots)$$

We will show that the spectral radius of $B(x)$ is less than 1 for any vector x such that $x < \mathbf{q}^*$ (i.e., strictly less in all coordinates). It will follow that for such x, $(I - B(x))^{-1} = \sum_{i=0}^{\infty} (B(x))^i$ exists (i.e., is finite) and is non-negative. First, some key lemmas.

Lemma 6.3 *For $0 \leq x \leq y$,*

1. $$B(x)(y-x) \leq P(y) - P(x) \leq B(y)(y-x) \tag{1}$$

2. *Moreover, if S_A^j is a linear system, then $B(x) = B(y)$, and thus $B(x)(y-x) = P(y) - P(x) = B(y)(y-x)$*

3. *If we let q^* denote $\mathrm{LFP}(S_A^j)$, then $B(x^i)(q^* - x^i) \leq (q^* - x^{i+1})$, for all $i = 1, 2, \ldots$, where $x^i = P^i(\mathbf{0})$*

4. *And, also $(q^* - x^{i+1}) \leq B(q^*)(q^* - x^i)$, where again $x^i = P^i(\mathbf{0})$.*

5. $$F'(x)(y-x) \leq F(y) - F(x) \leq F'(y)(y-x) \tag{2}$$

Proof. We only need to prove (1.), since (2.), (3.), (4.), and (5.) follow directly from (1.). In particular, for (2.) note that in that case $B(x)$ is a constant matrix, and thus $B(x) = B(y)$. For (3.) and (4.), take $y = q^*$ and $x = x^i$. Lastly, for (5.), note that $F'(x) = (B(x) - I)$, and $F(y) - F(x) = (P(y) - P(x)) - (y - x)$. By subtracting $(y - x)$ from each part of the inequality (1), we obtain the inequality (2).

Now we prove (1.). To show $B(x)(y - x) \leq P(y) - P(x)$, consider a variable x_i and the corresponding entry of $B(x)(y-x)$. There are two non-trivial cases to consider, based on the kind of the equation of x_i:
Case 1: $P_i(x)$ is linear. Then
$[B(x)(y - x)]_i = \sum_j p_{i,j}(y_j - x_j) = \sum_j p_{i,j}y_j - \sum_j p_{i,j}x_j = P_i(y) - P_i(x)$.
Note that equality holds in this case, establishing (2).
Case 2: $P_i(x)$ is nonlinear. Then

$$[B(x)(y-x)]_i = \sum_{k=1}^{l}[x_{i_k}(y-x)_{j_k} + x_{j_k}(y-x)_{i_k}] + \sum_{k=l+1}^{m} p_{ij_k}(y-x)_{j_k}$$

The expression in the brackets is

$$\begin{aligned} &\leq y_{i_k}(y-x)_{j_k} + x_{j_k}(y-x)_{i_k} \\ &= y_{i_k}y_{j_k} - x_{i_k}x_{j_k} \end{aligned}$$

Thus,

$$\begin{aligned} [B(x)(y-x)]_i &\leq \sum_{k=1}^{l} y_{i_k}y_{j_k} - \sum_{k=1}^{l} x_{i_k}x_{j_k} + \sum_{k=l+1}^{m} p_{ij_k}y_{j_k} - \sum_{k=l+1}^{m} p_{ij_k}x_{j_k} \\ &= P_i(y) - P_i(x) \end{aligned}$$

The proof that $P(y) - P(x) \leq B(y)(y - x)$ is identical: it is the "mirror image" of this proof and is left to the reader. ■

Lemma 6.4 *Assume $0 \leq x \leq y$, and $x \leq P(x)$. Then*

1. *$(B(x))^d(y - x) \leq P^d(y) - P^d(x)$.*
2. *If, S_A^j is a linear system, then $B(x)^d(y - x) = P^d(y) - P^d(x)$.*
3. *If S_A^j is nonlinear, and $x < q^* = LFP(S_A^j)$ (in every coordinate), then for each variable index s, there is some $d_s \geq 1$, such that $(B(x)^{d_s}(P(x) - x))_s < (P^{d_s+1}(x) - P^{d_s}(x))_s$.*

Proof.

1. By induction on d using Lemma 6.3.

   Base case, $d = 1$: $B(x)(y - x) \leq P(y) - P(x)$.

   Inductively, assuming $B(x)^d(y-x) \leq P^d(y) - P^d(x)$, then since $x \leq P(x)$ and $P(x)$ is monotone, $x \leq P^d(x)$ for all $d \geq 1$. Thus, since each entry of $B(x)$ is non-negative

and non-decreasing as a function of $x \geq 0$, we have $0 \leq B(x) \leq B(P^d(x))$. Thus, by Lemma 6.3,
$(B(x))^{d+1}(y-x) \leq B(P^d(x))(B(x))^d(y-x) \leq B(P^d(x))(P^d(y)-P^d(x)) \leq (P^{d+1}(y)-P^{d+1}(x))$.

2. Noting that in this case $B(x)$ is a constant matrix, independent of x, use part (2.) of Lemma 6.3, and part (1.) of this Lemma.

3. Suppose there is at least one nonlinear term in some equation. This means that $B(x)$ depends non-trivially on some coordinate x_r of x. Suppose, moreover, that $x < q^*$.

   Since we are dealing with a system S_A^j for a strongly connected component C_j, the underlying weighted directed graph defined by the matrix $B(x)$ is strongly connected (it is equal to C_j), and it is thus easy to show that for each variable index s there exists some power $1 \leq d'_s \leq n$, such that the sth row of $B^{d'_s}$ has in some column a term that contains the variable x_r. In other words, there is some column index l, such that for any vector $x' \geq x$, where $x'_r > x_r$, $(B^{d'_s}(x))_{s,l} < (B^{d'_s}(x')_{s,l}$.

   Now, since $0 \leq x < q^*$, the sequence $P^m(x), m = 1, 2, \ldots$ converges to the LFP q^*. Thus, there exists some m such that $P^m(x) > x$ in every coordinate. Moreover, for all such m, and for each index l, there exists $m_l > m$, such that $(P^{m_l+1}(x) - P^{m_l}(x))_l > 0$. To see this note that, since we have a non-trivial SCC which means that every variable depends on some other variable, and since $q^* > x$, it follows by induction, that $q^* > P^m(x)$ for all m. Thus since $\lim_{m\to\infty} P^m(x) = q^*$, it must also be the case that for some $m_l > m$, $P^{m_l+1}(x)_l > P^{m_l}(x)_l$.

   Now, let $d_s = d'_s + m_l$.

$$\begin{aligned} B^{d_s}(x)(P(x)-x) &= B^{d'_s}(x)B^{m_l}(x)(P(x)-x) \\ &\leq B^{d'_s}(x)(P^{m_l+1}(x) - P^{m_l}(x)) \\ &\leq^\star B^{d'_s}(P^{m_l}(x))(P^{m_l+1}(x) - P^{m_l}(x)) \\ &\leq P^{d_s+1}(x) - P^{d_s}(x) \end{aligned}$$

   Note, moreover, that $B^{d'_s}(x)_{s,l} < B^{d'_s}(P^{m_l}(x))_{s,l}$ and $(P^{m_l+1}(x) - P^{m_l}(x)))_l > 0$. Thus the second inequality, $\leq^\star$, is strict in component s. Therefore, $(B^{d_s}(P(x) - x))_s < (P^{d_s+1}(x) - P^{d_s}(x))_s$.

■

Lemma 6.5 *If $x < \mathrm{LFP}(S_A^j)$, then $\rho(B(x)) < 1$.*

Proof. By standard facts about matrices ([LT85, HJ85]) $\rho(B(x)) < 1$ if and only if $\lim_{d\to\infty}(B(x))^d = 0$, (i.e., each coordinate of the matrix $(B(x))^d$ converges to 0 in the limit, as $d \to \infty$).

Let x^0, x^1, ... be the sequence of value vectors generated in the standard iterative LFP algorithm, i.e., $x^i = P^i(\mathbf{0})$. Having reduced our equations to one SCC, S_A^j, it is easily shown that the vector $x^i = P^i(0)$ is never equal to q^* in any coordinate, for any $i > 0$.

(This is because for any x^i, each coordinate has at least one "neighbor" that influences its probability which also hasn't reached its value in q^*.)

Thus $(q^* - x^i) > 0$. Now, by Lemma 6.4, part (1.), letting $y = q^*$, we have: $B^d(x^i)(q^* - x^i) \leq (q^* - x^{i+d})$. The right hand side goes to 0 as $d \to \infty$ since by Theorem 3.1 $\lim_{d\to\infty} x^{i+d} = q^*$. However, $(q^* - x^i) > 0$. Thus $\lim_{d\to\infty} B^d(x^i) = 0$.

Furthermore, note that since x^i converges monotonically to q^* as i increases, for any x which is less than q^* in every coordinate, there is some i such that $x \leq x^i$, and thus $B(x) \leq B(x^i)$. Hence $\lim_{d\to\infty} B^d(x) = 0$. Therefore $\rho(B(x)) < 1$. ∎

Thus, $(-F'(x_k))^{-1} = (I - B(x_k))^{-1}$ exists as long as $x_k < \mathbf{q}^*$, and if so, $(-F'(x_k))^{-1} = (I + B(x_k) + B(x_k)^2 + \ldots)$. Thus note again that, $(-F(x_k))^{-1}$ is always nonnegative.

Thus, the Newton iterates become, $x_0 = \mathbf{0}$, and:

$$x_{k+1} = x_k + (I + B(x_k) + B(x_k)^2 + \ldots)(P(x_k) - x_k) \tag{3}$$

We know that $B(x_k) \geq 0$. To establish that $x_{k+1} \geq x_k$, we want to show that $(P(x_k) - x_k) \geq 0$.

Lemma 6.6 *If $x_k < q^*$ is the k'th Newton iterate for an SCC, S_A^j, then $P(x_k) \geq x_k$.*

Proof. We prove by induction on k. For $k = 0$, we have $x_0 = \mathbf{0}$, and $P(x_0) \geq x_0$. Note that, because we have a nontrivial SCC which is not a bottom SCC, $x_0 < q^*$.

Inductively, suppose $P(x_{k-1}) \geq x_{k-1}$, and $x_{k-1} < q^*$. We have
$x_k = x_{k-1} + (I - B(x_{k-1}))^{-1}(P(x_{k-1}) - x_{k-1})$. Thus

$$x_k - x_{k-1} = (I - B(x_{k-1}))^{-1}(P(x_{k-1}) - x_{k-1})$$

Note that, by inductive assumption and Lemma 6.5, the right hand side is well-defined and nonnegative, and thus $x_k \geq x_{k-1}$. Using Lemma 6.3 part (5.), we have:

$$\begin{aligned}
(P(x_k) - x_k) - (P(x_{k-1}) - x_{k-1}) &= F(x_k) - F(x_{k-1}) \\
&\geq F'(x_{k-1})(x_k - x_{k-1}) \\
&= (B(x_{k-1}) - I)(x_k - x_{k-1}) \\
&= (B(x_{k-1}) - I)(I - B(x_{k-1}))^{-1}(P(x_{k-1}) - x_{k-1}) \\
&= -(P(x_{k-1}) - x_{k-1})
\end{aligned}$$

Adding $(P(x_{k-1}) - x_{k-1})$ to each side of the inequality, we get $(P(x_k) - x_k) \geq 0$. ∎

What remains to show is that, for all k, $x_k \leq q^*$, and, unless $x_k = q^*$, $x_k < q^*$.

Note that $x_{k+1} = x_k + (I + B(x_k) + B(x_k)^2 + \ldots)(P(x_k) - x_k)$. Since $B(x_k) \geq 0$, and $(P(x_k) - x_k) \geq 0$, we have an absolutely convergent series and by standard facts about infinite series we can rewrite this safely as:

$$x_{k+1} = x_k + \sum_{d=0}^{\infty} (B(x_k))^d (P(x_k) - x_k)$$

Now, using Lemma 6.4 part (1.), we have

$$\begin{aligned} x_{k+1} &= x_k + \sum_{d=0}^{\infty} (B(x_k))^d (P(x_k) - x_k) \\ &\leq x_k + \sum_{d=0}^{\infty} (P^{d+1}(x_k) - P^d(x_k)) \\ &= \lim_{m \to \infty} [x_k + \sum_{d=0}^{m} (P^{d+1}(x_k) - P^d(x_k))] \\ &= \lim_{m \to \infty} P^{m+1}(x_k) = q^* \end{aligned}$$

Thus $x_{k+1} \leq q^*$. Note also that $x_{k+1} \geq P(x_k)$, by using only the first term, $I(P(x_k) - x_k)$, in the sum $\Sigma_{d=0}^{\infty}(B(x_k))^d(P(x_k) - x_k)$. The other terms are non-negative. Thus, by induction, we have:

Lemma 6.7 *For all $k \geq 0$, $x^{k+1} \leq P(x_k) \leq x_{k+1} \leq q^*$.*

Thus, $x_0 \leq x_1 \leq \ldots$, converges monotonically to q^*, as long as every iteration is defined, and the $(k+1)$'st iteration is defined as long as $x_k < q^*$.

We now show that either $x_0 = 0 = \mathbf{q}^*$, or $x_1 = q^*$, or else $x_k < q^*$ for $k \geq 0$, i.e., strictly less in every coordinate, in which case, we have already seen that $(-F(x_k))^{-1}$ always exists, and thus the iterates are defined.

There are basically two cases to consider: either the system is linear and $B(x)$ is a constant matrix, or, the system is nonlinear and $B(x)$ contains some variable.

We will show that in the first case, one iteration of Newton's method is sufficient to reach q^*. This follows because, by Lemma 6.3 part (2.), in the case where the system is linear, we have $B(x)(y - x) = P(y) - P(x)$. Thus

$$\begin{aligned} x_1 &= x_0 + \Sigma_{d=0}^{\infty}(B(x_0))^d(P(x_0) - x_0) \\ &= x_0 + \Sigma_{d=0}^{\infty}(P^{d+1}(x_0) - P^d(x_0)) \\ &= \lim_{m \to \infty} P^m(x_0) \\ &= q^* \end{aligned}$$

Note in particular that this is the case with finite Markov chains: one iteration of Newton's method is enough to converge to the solution. This corresponds to the known fact that the solution for a Markov chain can be found by solving a linear system. Observe that by partitioning the system into SCCs, we have insured that the linear system can be solved by a matrix inversion for each component.

Next, let us consider the case where $B(x)$ is not constant. We have to show that $x_k < q^*$ for all k.

Lemma 6.8 *If $B(x)$ is not a constant matrix, and $0 < q^*$, then $x_k < q^*$ for all $k \geq 0$.*

Proof. This follows by combining Lemma 6.4 together with the sum formula $x_{k+1} = x_k + \sum_{d=0}^{\infty}(B(x_k))^d(P(x_k) - x_k)$. In particular, if $x_k < q^*$, then by Lemma 6.4 and the sum formula, $x_{k+1} = x_k + \sum_{d=0}^{\infty}(B(x_k))^d(P(x_k) - x_k) < x_k + \sum_{d=0}^{\infty} P^{d+1}(x_k) - P^d(x_k) = q^*$

The reason for the *strict* inequality is that, by Lemma 6.4, part (3.), for each variable index s, there is a power d_s such that $(B(x_k)^{d_s}(P(x_k) - x_k))_s < (P^{d_s+1}(x_k) - P^{d_s}(x_k))_s$. Since this is so for every index s, i.e., in every coordinate, we see that $x_{k+1} < q^*$ ■

We are done with the proof of Theorem 6.1, and have established the monotone convergence of the decomposed Newton's method for computing LFP(P). ■

From our proof it actually follows that Newton's method in general constitutes a rapid "acceleration" of the standard iteration, $P^k(\mathbf{0})$, $k \to \infty$. More specifically, by equation 3, $x_{k+1} = x_k + (\sum_{i=0}^{\infty} B(x_k)^i)(P(x_k) - x_k)$. Now, $B(x_k)$ is a non-negative matrix, and $(P(x_k) - x_k)$ is a non-negative vector. If we replace $(\sum_{i=0}^{\infty} B(x_k)^i)$ by only the first term of the infinite series, i.e., by the identity matrix, then the right hand side becomes $P(x_k)$, i.e., we get standard iteration. Thus $x_{k+1} \geq P(x_k)$, and when $B(x_k)$ is "large", x_{k+1} will potentially be much larger than $P(x_k)$.

In particular, for finite Markov chains, which generate linear systems, and where $B(x)$ is a constant matrix, note that the decomposed Newton's method converges in one Newton iteration to LFP(P). This is certainly not the case for finite MCs with standard iteration $P^k(\mathbf{0})$, where as long as there are cycles in the finite MC it will only converge in the limit.

Also, observe that for the RMC A' in Figure 4, for which we showed in Theorem 3.2, part 2, that $P^k(\mathbf{0})$ converges "exponentially slowly", Newton's method converges linearly: namely, for $(1/2)x^2 - x + 1/2 = 0$, Newton's method is the iteration $x_{k+1} = (1/2) + (1/2)x_k$. It is easy to see that, with $x_0 = 0$, $|1 - x_k| = 2^{-k}$.

7 General monotone systems of polynomial equations

In this section we observe the fact that in prior sections, when discussing the analysis of RMC equations via the Existential Theory of Reals and via Newton's method, we did not need to confine our discussion to RMC equations, but to any *Monotone System of Polynomial Equations* (MSPE), $x = P(x)$. These are systems of equations with one equation $x_i = P_i(x)$ for each variable, x_i, where $P_i(x)$ is a (multivariate) polynomial with positive coefficients. In this section we overview this generalization of the theory. Since the proofs are relatively simple modifications of the proofs given for RMCs, we forgo detailed proofs. First, we note that Theorem 3.1 can easily be modified to show the following:

Corollary 7.1 *Given any MSPE, $x = P(x)$,* if *the system has a nonnegative solution q', then it has a LFP, i.e., a least nonnegative solution $q^* = \mathrm{LFP}(P)$, such that $q^* = P(q^*)$ and $P^k(\mathbf{0}) \uparrow q^*$ as $k \to \infty$.*

Of course, a general monotone (non)linear system may not have any nonnegative solutions: consider $x_1 = x_1 + 1$. On the other hand, the MSPE's corresponding to RMCs certainly do have a nonnegative solution, and an LFP. Using identical arguments to those of section 4, we can employ the Existential Theory of Reals to answer queries about $x = P(x)$, and we obtain the following:

Corollary 7.2 *Given any monotone system of polynomial equations, $x = P(x)$, we can decide in PSPACE whether $x = P(x)$ has a nonnegative solution, and if so we can decide whether $\mathrm{LFP}(P) = \mathbf{c}$, for given rational probabilities $\mathbf{c}$, and we can decide whether $q_j^* \Delta c_j$*

for any comparison operator Δ. *Moreover, if* LFP(P) *exists we can approximate the coordinates of* LFP(P) *to within j bits of precision (j given in unary), also in PSPACE. The running time is $O(j \cdot |\mathbf{c}| \cdot |A|^{O(n)})$, where n is the number of variables in* $\mathbf{x}$.

Next, we observe that a decomposed Newton's method can also be used for finding the LFP of a general MSPE, $x = P(x)$, if any nonnegative solution exists. Suppose we are given such a MSPE. To simplify our discussion, we first show that without loss of generality we can assume our MSPE system comes in a certain normal form. Namely, a *canonical form* MSPE has only three types of equations, $x_i = P_i(x)$, which are completely analogous to the three types $Type_1$,$Type_{rand}$, and $Type_{call}$ for RMC equations:

1. $Type_1$: $x_i = 1$
2. $Type_{rand}$: $x_i = \sum_{j=1}^{n} a_j x_j$, with $a_j \geq 0$. (Note: a_j's need not be probabilities that sum to 1.)
3. $Type_{call}$: $x_i = x_j x_k$.
   (Note: because weights are arbitrary we don't need a sum of products, as we did for RMCs.)

Proposition 7.3 *Any MSPE system $x = P(x)$ can be converted in P-time to a system in canonical form with possibly more variables, such that there is a 1-1 correspondence between the nonnegative solutions of the two systems, which preserves the values of the original variables x.*

Proof.

We are given an MSPE $x = P(x)$. If some variable x_i occurs in some polynomial of $P(x)$ with exponent higher than 1 then introduce new variables to represent the powers of x_i with exponents that are powers of 2, and use them to eliminate the higher powers from $P(x)$. That is, introduce variables $x', y_1, y_1', y_2, y_2', \ldots, y_l, y_l'$ where l is the logarithm of the highest exponent of x_i in $P(x)$, and add equations: $x' = x$, $y_1 = xx'$, $y_1' = y_1$, $y_2 = y_1 y_1'$, $y_2' = y_2$, and so forth. Then replace every higher power x_i^k of x_i in $P(x)$ by the expression $x^{a_0} y_1^{a_1} y_2^{a_2} \ldots y_l^{a_l}$ where $a_l \ldots a_2 a_1 a_0$ is the binary representation of the exponent k. After this transformation, every term of every polynomial in $P(x)$ contains distinct variables.

If some polynomial $P_i(x)$ contains a term with two or more variables, and $P_i(x)$ is not just the product of two variables, then take two of these variables x_j, x_k that appear in such a term, introduce a new variable $y_{\{j,k\}}$ and replace the product $x_j x_k$ in every monomial in which it occurs (in the entire system) by $y_{\{j,k\}}$. We also add a new equation $y_{\{j,k\}} = x_j x_k$ to the system. Repeat this same procedure, adding new variables and equations, until every $P_i(x)$ is a constant or is linear or is the product of two variables. Finally, at the end of this process it may be the case that a number of equations $x_i = P_i(x, y)$ have a constant term a on the right hand side. Add a new $Type_1$ variable z, replace all such constant terms on right hand sides by az, and lastly add the equation $z = 1$. It is easy to see that by doing this we will end up with a system in canonical form. Furthermore, this system is "equivalent" to our original one in the following sense: any solution of the new system projected on to the original variables is a solution of the old system, and any solution of the old system can be extended uniquely to a solution of the new system. ■

Consider a canonical form MSPE. Since no solution may exist, we have to augment the decomposed Newton's Method of Figure 6, by adding the following extra stopping condition inside the while loop:

* Stop if ever $P(x_k) \not\geq x_k$, or if $(F'(x_k))^{-1}$ does not exist, and report "*no nonnegative solution*". (This check is unnecessary for RMCs.)

We also need to explain how some other steps in the algorithm in Figure 6 will be carried out for general systems. In particular, step 1 requires us to eliminate all variables x_i where $q_i^* = 0$. Since for general systems the LFP q^* might not even exist, we have to make sense of this step in a different way. Basically, as we saw in the proof of Theorem 6.1, the system $x = P(x)$ can be viewed as a graph G whose nodes are the variables x_i, and where each node is labeled according to the type of the variable. The only sink node in this graph is the $Type_1$ variable z, which we may assume to be unique.

The graph G can be viewed as a game graph, where $Type_{rand}$ nodes belong to player 1, and $Type_{call}$ nodes belong to player 2. Now, a variable x_i should be eliminated during preprocessing from $x = P(x)$ (because q_i^* must equal 0 if q^* exists) if and only if there is no strategy for player 1 to reach node z starting at node x_i. It should be intuitively obvious why the lack of such a strategy implies that $q_i^* = 0$ if LFP q^* exists. This is because the equations for such variables depend only on other such variables and do not contain any constant terms. Consequently, assigning 0 to all such variables necessarily satisfies the equation associated with all of them.

Checking whether such a strategy exists can be done easily in linear time in the size of the graph G, for all x_i. This is essentially identical to the algorithm and proof for Theorem 2.2. We thus have:

Proposition 7.4 *For a general MSPE in canonical form, $x = P(x)$, we can detect and eliminate in P-time those variables x_i such that if a non-negative solution to $x = P(x)$ exists, then $q_i^* = 0$, where $q^* = \mathrm{LFP}(P)$.*

After the preprocessing step, the rest of the decomposed Newton's method proceeds entirely as in Figure 6, with the added stopping condition given above. We obtain:

Corollary 7.5 *If a given MSPE (in canonical form), $x = P(x)$, has a non-negative solution, then the (revised) decomposed Newton's method will converge monotonically to the LFP, q^*, i.e., $\lim_{k\to\infty} \mathbf{x_k} \uparrow q^*$.*

(If a non-negative solution does not exist, then the algorithm will either report that "no non-negative solution exists", or will simply terminate after the maximum number of iterations k have been exhausted and report its final value.)[6]

The proof of this is a corollary of our proof for RMC equations. Modifying that proof, we need to note that, because q^* may not exist, in several places we have to condition the statements in that proof by the assumption that q^* exists. Importantly, in that proof we at no point rely on the fact the coefficients in RMC equations are probabilities. In the interests of space, we leave a detailed modification of the proof of Theorem 6.1 to the reader.

Finally, we mention without further elaboration that an MSPE in canonical form can easily be viewed as a *Weighted Context-Free Grammar* (WCFG) in generalized Chomsky Normal Form. A weighted CFG is like a CFG where the rules have associated positive

[6] In other words, the algorithm will not necessarily give the right answer if we don't iterate "enough" but it will never terminate unnaturally due to an iteration being ill defined because $(F'(x_k))^{-1}$ does not exist.

weights that are not necessarily probabilities, i.e. they do not have to be ≤ 1 and the weights of the rules with the same left hand side do not have to add up to 1. The question of the existence of a nonnegative solution to $x = P(x)$ corresponds to the question whether the total weight of all finite parse trees of the corresponding WCFG is finite. WCFG's and their relationship to pPDSs and SCFGs were considered in [AMP99], but they did not consider algorithms for analyzing them.

8 Polynomial-time algorithms for restricted classes of RMCs

8.1 Almost-sure termination/extinction for 1-exit RMCs, SCFGs, and MT-BPs

As shown in Section 2.3, 1-RMCs, MT-BPs, and SCFGs are all tightly related. In particular, the probability of extinction of an object of a given type in a MT-BP is the same as the probability of termination starting at the corresponding nonterminal in the corresponding SCFG, and it is the same as the probability of termination of the corresponding 1-exit RMC starting at the entry of the component corresponding to the given type of the MT-BP (nonterminal of the SCFG). In particular, an SCFG is called *consistent* if it generates a terminal string with probability 1. In this section we provide a simple, concrete, and efficient algorithm to check consistency of a SCFG, and almost sure (i.e. probability 1) termination of a 1-exit RMC, and extinction of a MT-BP.

Given the connection between these models, one can use classical results on branching processes [KS47, Sev51, Har63], to "characterize" the question of consistency of a SCFG as a question about the spectral radius of the *moment matrix* associated with the SCFG, namely, the nonnegative matrix $B(1)$ which is the Jacobian of $P(x)$ evaluated at the all 1 vector, i.e., where $(B(x))_{i,j} = \partial P_i(x)/\partial x_j$. See [Har63] for the classic text on Branching Processes, and see e.g., [BT73] for the connection to SCFGs.

These well known "characterizations" unfortunately often leave out some special uncovered cases (and sometimes contain errors or omit assumptions). None of the generally available references in English that we are aware of give a complete characterization and proof for all MT-BPs (and all SCFGs). The most comprehensive results, and also among the earliest results, are due to Kolmogorov and Sevastyanov [KS47, Sev51] (their papers are in Russian). These results are stated without proof in the classic text on Branching Processes by Harris [Har63], which refers to [Sev51] for the "quite complicated proofs". As quoted in [Har63], a general result that applies to all multitype branching processes is a necessary and sufficient condition for the probability of extinction to be 1 for *all* initial sets of entities, i.e. for all the types: this is the case if and only if $\rho(B(1)) \leq 1$ and the branching process has no so-called "final classes"; these are special classes of types from which the probability of extinction is zero. The textbook [Har63] itself proves the characterization for the case of MT-BPs for which the nonnegative moment matrix $B(1)$ is a *primitive* matrix (see, e.g., [HJ85]). This is what [Har63] calls the "*positively regular*" case. Primitive matrices allow [Har63] to apply the strongest form of the Perron-Frobenius theorem to the matrices $B(1)$.

More recent texts on Branching Processes, such as [Mod71, AN72, Jag75, HJV05], typically state (and in the case of [Mod71] reprove) the results in [Har63] for the positively-

regular (primitive) case of MT-BPs, and also often mention (e.g., [Jag75]) the kinds of difficulties that arise for generalizing the results to all MT-BPs. But none provide a complete algorithmic characterization and proof for all MT-BPs. It is worth noting also that there are occasional errors or missing assumptions in the statements of the theorems in some references. For example, [BT73] which is widely cited in the SCFG literature, states that if a SCFG has $\rho(B(1)) > 1$ then the SCFG is inconsistent; however, this is false for grammars where some nonterminals may be unreachable from the start nonterminal. A recent survey on Branching Processes ([AV01]) states without proof a result (Theorem 8) for general "irreducible" MT-BPs, namely that "for an irreducible MT-BP with moment matrix M, $q_i^* < 1$ for all i if and only if $\rho(M) > 1$", but the statement is in fact not quite correct, precisely because it ignores Sevastyanov's case of "final classes", which can exist in an irreducible MT-BP with $\rho(M) \leq 1$, but where the extinction probability is in fact 0.

Given this state of affairs, with no complete algorithmic characterization and proof available in English for all MT-BPs and SCFGs, in neither the Branching Process literature nor in the SCFG literature, and with several mistakes in both the MT-BP and SCFG literature, we feel that it is appropriate to provide a complete characterization, a concrete polynomial time algorithm, and a complete proof for general MT-BPs and SCFGs. That is what we do in this section.

Our proof is structurally similar to the proof of [Har63]. We extend [Har63]'s proof by decomposing the system $x = P(x)$ into SCCs, and analyzing each SCC by a further decomposition into its aperiodic parts. We also simplify and modify several aspects of [Har63]'s proof. In particular, it turns out that one direction of our proof of the characterization via the moment matrix $B(1)$ follows very easily from our results about $B(x)$ established for our analysis of the decomposed Newton's method in Section 6.

Given a SCFG we can write directly a system of polynomial equations $x = P(x)$, in the vector of variables x that contains one variable x_i for every nonterminal S_i, such that the probabilities $p(S_i)$ form the LFP of the system. The polynomial $P_i(x)$ has one term for every rule of the grammar with lhs S_i; if the rule has probability p and the right-hand-side of the rule contains ℓ_j occurrences of nonterminal S_j, $j = 1, \ldots, d$, then the corresponding term in P_i is $p \prod_j x_j^{\ell_j}$. If we form the Jacobian matrix $B(x)$ and evaluate it at $x = 1$, then the i, j entry of the resulting matrix $B(1)$ is equal to $\sum p\ell_j$ where the summation ranges over all rules with lhs S_i, thus, $B(1)_{i,j}$ is the expected number of occurrences of S_j on the rhs of a rule with lhs S_i. The matrix $B(1)$ is called the (first) moment matrix of the SCFG.

Our algorithm for checking SCFG consistency is outlined in Fig. 7. From classical algorithms for processing context-free grammars we know how to compute all reachable and all useless nonterminals in linear time (see eg. [HMU00]). To finish the algorithm, we only need to explain how one 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 do this. One is by appealing to the existential theory of the reals. By the Perron-Frobenius theorem (see [LT85]), 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 $\rho(B(1)) > 1$. We will return to this point

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

Output: YES if G is consistent, NO if it is not.

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 moment 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.

Figure 7: SCFG consistency algorithm

after we prove the correctness of the algorithm.

We will actually present a somewhat more involved algorithm, given in Figure 8, where we classify, in one pass, the termination probability of all vertices of a 1-exit RMC. It is not hard to show that the correctness of the algorithm in Figure 7 for classifying the start nonterminal of an SCFG follows from the correctness of the algorithm in Figure 8 for classifying all non-terminals of an SCFG (or, equivalently, all vertices of a 1-exit RMC). We will show this after the proof of the main theorem of this section.

Theorem 8.1 *Given a 1-exit RMC, A, or equivalently a SCFG or MT-BP, there is a polynomial time algorithm to determine, for each vertex u which of the following three cases holds: (1) $q_u^* = 0$, or (2) $q_u^* = 1$, or (3) $0 < q_u^* < 1$. In particular, we can test SCFG consistency in polynomial time.*

Proof. We are given an 1-exit RMC A and wish to determine, for all vertices u, whether $q_u^* = 1$, $q_u^* = 0$, or $0 < q_u^* < 1$. We will label each vertex of the RMC (and corresponding variable of the system) by 1, 0 or the symbol \$ respectively according to which of the three cases holds. The Algorithm is given in Fig. 8.

In step (1.), the preprocessing step, we identify and remove all the variables corresponding to vertices that cannot reach an exit, hence are 0. For the vertices that remain, all termination probabilities are > 0. Thus, we need to test if they are 1 or < 1.

We can view the system of equations $S_{A'}$ for this preprocessed RMC as a cyclic circuit that has one node for every variable. Let C_A denote the cyclic circuit over $\{1, +, *\}$, whose nodes are labelled by either a 1 ($Type_1$), + ($Type_{rand}$), or $*$ ($Type_{call}$). We can merge all the type I nodes into one node labelled 1, and label all the corresponding type I vertices of the RMC (and corresponding variables) by 1. The edges from a +-node to its successors are labeled by probabilities (that may not necessarily sum to 1, since we have already removed probability 0 nodes).

In step 2, we first break C_A into SCCs, and we topologically sort the underlying DAG to get SCCs $C_1, \ldots, C_k$, where if there is an edge from C_i to C_j, then $j \geq i$. Let C_k be the singleton node labelled 1 (it does not have any successors). Note that there is no other bottom SCC (i.e. SCC that has no outgoing edges), because if there was, all the corresponding variables must be 0 in the LFP, but we have already eliminated all the variables that are 0. Note also that if there is a path in C_A from a node x_u to a node

Input: A 1-exit RMC A

Output: Marking of each vertex by 0, 1, or \$ depending on whether its termination probability is 0, 1, or strictly between 0 and 1.

1. Construct the system $S_A : x = P(x)$ for A.

2. Preprocess $x = P(x)$, to determine and eliminate all x_u such that $q^*_u = 0$. Eliminate all such variables from S_A.
   This gives a new system, $S_{A'}$ in which the remaining reachability probabilities are either 1 or some p, where $0 < p < 1$.

3. Decompose the system $S_{A'}$ into Strongly Connected Components. Consider the DAG of SCCs, and topologically sort the SCCs as $C_1, \ldots, C_k$.
   Mark the trivial bottom SCC, containing one probability 1 node, as having probability 1.

4. Process the remaining SCCs "bottom up". Let C_r be the current lowest unprocessed SCC. C_r must have successors (because we have already eliminated all probability 0 nodes during preprocessing).

   (a) If C_r has some successors marked "\$", meaning their probability is strictly in between 0 and 1, then mark (all nodes in) C_r with \$.

   (b) Else, if C_r contains any +-node for which the sum of edge probabilities from that node to successors is < 1, then mark all of C_r with \$.

   (c) Else, if C_r is a singleton (contains only 1 node) and is also acyclic (i.e., there are no transitions from the node to itself), then: if all its successors are marked 1, then mark it also as 1 (otherwise, the singleton was already marked \$ in 3(a) or in 3(b)).

   (d) Else, i.e., if C_r is a nontrivial SCC all of whose successors are marked 1 and in which all + nodes have outedge probabilities summing to 1, let $\rho_r \in \mathbb{R}_{\geq 0}$ be the maximum eigenvalue of $B_r(1)$, the Jacobian matrix of C_r evaluated at the all 1-vector.
   If $\rho_r \leq 1$ then mark C_r with probability 1, otherwise mark C_r with \$.

Figure 8: **1-exit RMC algorithm: 0, 1, or "in between" probability.**

x_v, which indicates that the variable x_u depends (indirectly) on the variable x_v, and if $q_v^* < 1$, then also q_u^* must be less than 1; i.e., if v should be labelled \$ then also u should be labelled \$. It follows that in a SCC, all nodes must receive the same label, either all \$ or all 1. We shall process the SCCs "bottom up", starting at C_k, to compute whether the probabilities associated with nodes in each C_i are 1 or < 1. We then use this information to compute the same for prior components.

Now, in step 3, assume by induction that we are about to process C_r, where all components C_l with $l > r$ have already been processed and whose nodes have already been labelled with one of $\{1, \$\}$.

Clearly, if there is some node in C_r that depends on a node in a lower SCC that has been marked \$ ("in between"), then every node in C_r should also be marked \$ (because all of the probabilities in C_r depend on all others in C_r). That is what is done in step 3(a) of the algorithm.

Next, if there is some +-node in C_r for which the sum of edge probabilities from that node to successors is < 1, then that means that we have already eliminated a 0-successor of that node during preprocessing, and thus that again, since all probabilities in C_r depend on all others, we can mark every node in C_r as "\$" ("in between"). That is what is done in step 3(b).

Also clearly, if a remaining SCC C_r is a singleton that does not depend on itself, and if all of its successors have probability 1, then it also has probability 1. This is checked in step 3(c). Note we've already checked in 3(b) if the node has less than full probability coming out of its edges, and in 3(a) if it has a \$ successor, and in each of those cases would have marked it \$.

What remains is a nontrivial SCC, C_r, for whose nodes we have to determine whether the associated probability is exactly 1 or < 1. Suppose there are n nodes in C_r. For each variable x_i in C_r, define the equation $x_i = f_i(x)$ to be the equation $x_i = P_i(x)$ restricted to the variables in C_r. By this we mean that if a variable x_v appears in $P_i(x)$ which is not in C_r, then let $x_v = 1$ in $f_i(x)$. Note that, by the processing we have done before, for any such variable x_v, we will have already determined that $q_v^* = 1$. Because C_r is a non-trivial SCC, in each such equation the right hand side $f_i(x)$ does not reduce to a constant. Let $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ denote the underlying maps defined by this system of polynomials for each variable x_i in C_r, restricted to the variables in C_r. In other words, where the equation associated with x_i is $x_i = f_i(x)$, and $f(x) = (f_1(x), \ldots, f_n(x))$.

Let $B_r(1)$ be the Jacobian matrix of f, evaluated at the all 1 vector in $\mathbb{R}^n$ (see section 6). In other words, define the (i, j)'th entry of the matrix $B_r(x)$ to be $(B_r(x))_{i,j} := \frac{\partial f_i}{\partial x_j}$, and let $B_r(1)$ be the matrix where all of these entries are evaluated with $x_i := 1$ for all i. Let $\rho_r = \rho(B_r(1))$ be the spectral radius of $B_r(1)$, i.e., the maximum absolute value of an eigenvalue of the matrix $B_r(1)$. Since $B_r(1)$ is non-negative, by basic facts of Perron-Frobenius theory its maximum magnitude eigenvalue must be real and non-negative, and equal to ρ_r (see, e.g., [LT85, HJ85]).

We wish to show that if $\rho_r \leq 1$ then all nodes (i.e., variables) in C_r should get probability 1, whereas if $\rho_r > 1$, then all nodes in C_r have "in between" probability, and thus should get marked with \$. This is what is checked in the last step, 3(d).

First we establish the following "easier" direction of our claim about ρ_r, using the results established in section 6 on Newton's method.

Lemma 8.2 *If $\rho_r = \rho(B_r(1)) > 1$, then for every node marked x_u in the SCC C_r, the*

probability $q_u^* < 1$.

Proof. This is a simple corollary of Lemma 6.5 in Section 6, where we showed that for $\mathbf{x} < \mathrm{LFP}(C_r) = \mathbf{q}^*$, $\rho(B_r(\mathbf{x})) < 1$. Thus, by continuity of eigenvalues (see, e.g., [LT85]), $\rho(B_r(\mathbf{q}^*)) \leq 1$. Thus, if $\rho(B_r(1)) > 1$, then $\mathbf{q}^* \neq 1$. If there is some component $q_i^* < 1$ in $\mathbf{q}^*$, then all components must be < 1, because it is an SCC and every probability depends on every other. ■

Now we wish to establish that if $\rho_r \leq 1$ then every node in C_r should get probability 1. Although we don't actually need to do this in our algorithm, it is useful for the purposes of our proof to first consider partitioning the SCC C_r again, according to its "period". The *period* of a directed graph is the greatest common divisor of the lengths of all directed cycles in C_r. A digraph is *aperiodic* if its period is 1. If we wished to do so, there are simple algorithms (see [BV73, APY91]) that can be used to compute the period d of C_r in linear time, and also, within the same time, to partition the nodes of C_r into sets $C_{r,0}, \ldots, C_{r,d-1}$, such that all edges from nodes in each partition $C_{r,j}$ lead to nodes in partition $C_{r,(j+1 \mod d)}$. (Again, we don't actually need to compute this partition in our algorithm, but it will serve us in the proof.)

Let C_r^d denote the directed graph on nodes of C_r such that there is an edge from node x_i to node $x_{i'}$ if and only if there is a path of length d from x_i to $x_{i'}$ in C_r. It then follows that the map f^d, i.e., the d'th iterate of f, has associated with it the dependence graph C_r^d. If d is the period of f, then C_r^d consists of d disjoint induced subgraphs $C_{r,j}^d$, $0 \leq j < d$, each induced by the partition $C_{r,j}$, respectively. Furthermore, each subgraph $C_{r,j}^d$ consists of one aperiodic SCC, because the lengths of cycles in $C_{r,j}^d$ are precisely the lengths of cycles in C_r divided by the gcd d, and hence they have gcd $= 1$.

It is not hard to show that if d is the period of C_r, then $B_r(1)^d$ is a matrix whose only non-zero entries are in positions (i, i') where x_i and $x_{i'}$ are in the same partition $C_{r,j}$ of C_r. We can thus use the partitions $C_{r,j}$ to easily compute a permutation matrix Q such that the matrix $QB_r(1)^dQ^{-1}$ is a block-diagoanl matrix with blocks $B_{C_{r,0}^d}(1), \ldots, B_{C_{r,d-1}^d}(1)$ as follows:

$$QB_r(1)^dQ^{-1} = \begin{bmatrix} B_{C_{r,0}^d}(1) & 0 & 0 & 0 \\ 0 & B_{C_{r,1}^d}(1) & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & B_{C_{r,d-1}^d}(1) \end{bmatrix}$$

Each submatrix $B_{C_{r,j}^d}(1)$, which we will refer to as $B_{r,j}(1)$ has an underlying aperiodic SCC $C_{r,j}^d$. Hence, since the matrix is non-negative, there is a power m_j such that $B_{r,j}^{m_j}(1)$ has all entries positive. In other words $B_{r,j}(1)$ is a *primitive* matrix and thus we will be able to apply the full power of the Perron-Frobenius theory ([LT85, HJ85]) to matrices $B_{r,j}(1)$.

To do so, we first set up the following branching process which has one type for each node (variable) x_i of the SCC C_r. If x_i is a *-node with equation $x_i = \alpha$ in the subsystem $x = f(x)$ of C_r then the process contains a rule $x_i \xrightarrow{1} \alpha$; note α is either the product of two variables or a single variable (the latter happens if one of the two variables in the original system $x = P(x)$ belongs to a lower SCC). If x_i is a +-node with equation $x_i = p_0 + \sum_{j \in J} p_j x_j$ in the subsystem $x = f(x)$ of C_r (the rhs may or may not contain

a constant term p_0), then we have one rule $x_i \stackrel{p_j}{\to} x_j$ for each $j \in J$, and a rule $x_i \stackrel{p_0}{\to} 0$ (no offsprings) if there is a constant term (i.e., $p_0 > 0$). Note that since all + nodes have outedge probabilities summing to 1, $\sum_j p_j = 1$, and this is a proper branching process.

Let $z^0 \in \mathbb{N}^n$ be a (row) vector where z_i^0 denotes the number of occurences of the type (i.e. node) x_i in the initial population. Let $z^0, z^1, z^2, ...$ be the random sequence of vectors generated by the branching process. The matrix $B_r(1)$ is the *first moment matrix* for this process. It is not hard to see that the vector $\tilde{z}^m = z^0(B_r(1))^m$ is the vector of expected numbers of each type at stage m of this process (see [Har63]). In other words, $\mathbb{E}(z_j^m) = \tilde{z}_j^m$.

Lemma 8.3 *1. Given an initial vector $z^0 \in \mathbb{N}^n$, for any $z' \in \mathbb{N}^n$, $z' \neq 0$, $Pr(\exists^\infty m \text{ s.t. } z^m = z') = 0$.*
2. Thus, also, $Pr(\exists z' \in \mathbb{N}^n,\ z' \neq 0,\ \exists^\infty m \text{ s.t. } z^m = z') = 0$

Proof. Harris (see [Har63], Thm 6.1) proves this for the case where $B_r(1)$ is a primitive matrix (*positively regular* in [Har63]'s terminology). We give a somewhat different proof for any irreducible matrix $B_r(1)$. We know that $q_u^* > 0$ for all $x_u \in C_r$, i.e. every corresponding vertex u has a terminating path in the RMC; pick one such path for each vertex u, let n be the maximum length among these paths and let $c > 0$ be the minimum probability among these paths. If the branching process starts with a single entity of any type $x_u \in C_r$, then with probability at least c after n steps it terminates.

Let $M = \sum_{j=1}^n z'_j$. If $z^i = z'$, by the above, with probability $\geq c^M > 0$, $z^{i+n} = 0$. If $z^{i+n} = 0$ then $z^{i+n+d} = 0$ for all $d \geq 0$. On the other hand, if we return to z' at $z^{i'}$ for some $i' \geq i + n$, then we repeat the experiment. Thus, the probability of infinitely often revisiting vector $z' \neq 0$ in the trajectory $z^0, z^1, z^2, \ldots$ is 0.

The second assertion follows since there is a countable number of finite vectors $z' \in \mathbb{N}^n$: $Pr(\exists z' \in \mathbb{N}^n,\ z' \neq 0,\ \exists^\infty m \text{ s.t. } z^m = z') \leq \sum_{z' \in \mathbb{N}^n, z' \neq 0} Pr(\exists^\infty m\ z^m = z') = 0$. ∎

Note that if we instead let $z^0 \in \mathbb{N}^{k_j}$ be a vector of length $k_j = |C_{r,j}|$, then $z^0(B_{r,j}(1))^m$ is the expected number of each type in partition $C_{r,j}$ after $d * m$ stages of the original process, starting with $(z^0)_i$ entities of type x_i in $C_{r,j}$, and no entities of other types.

Lemma 8.4 *For each partition $C_{r,j}$, if $\rho(B_{r,j}(1)) \leq 1$, then for every node $x_i \in C_{r,j}$, $q_i^* = 1$. Otherwise, i.e., if $\rho(B_j(1)) > 1$, then for every node $x_i \in C_{r,j}$, $q_i^* < 1$.*

Proof. The second assertion follows from Lemma 8.2, because in fact if $\rho(B_{r,j}(1)) > 1$, then $\rho(B_r(1)) > 1$, because $B_{r,j}(1)$ is a submatrix of $Q(B_r(1))^d Q^{-1}$. Thus if its maximum eigenvalue is > 1, then so is the maximum eigenvalue of $Q(B_r(1))^d Q^{-1}$. But the eigenvalues of $Q(B_r(1))^d Q^{-1}$ are d'th powers of the eigenvalues of $B_r(1)$. Thus $\rho(B_r(1)) > 1$, and therefore by Lemma 8.2, every node in C_r (including all of $C_{r,j}$) has probability $q_i^* < 1$.

For the first assertion, suppose $\rho(B_{r,j}(1)) \leq 1$. Let $M = B_{r,j}(1)$, and let $\rho = \rho(M)$. M is a *primitive* matrix, meaning its underlying graph is irreducible and aperiodic. By the Perron-Frobenius Theorem for primitive matrices (see, e.g., Thm 2.3 of appendix of [Kar66]; or see chapter 8, section 8.5 of [HJ85]) $M^n = \rho^n * M_1 + (M_2)^n$, for two matrices M_1 and M_2 where $\rho(M_2) < \rho$. Thus, since $\rho(M) \leq 1$, it follows, e.g., by taking suitable matrix norms, that there is a bound $t \in \mathbb{R}_{\geq 0}$, such that for all n, all entries

of M^n are bounded above by t. For completeness, we provide the argument here for the existence of a bound t. Consider the l_1 matrix norm, $\|A\|_1 = \sum_{i,j} |a_{i,j}|$. Note that $\|M^n\|_1 = \|\rho^n M_1 + M_2^n\|_1 \leq \rho^n \|M_1\|_1 + \|M_2^n\|_1$. But $\rho(M_2) < 1$, and hence $\|M_2^n\|_1 \to 0$. Thus, there must be a bound $C < \infty$ such that $\|M_2^n\|_1 < C$ for any n (otherwise $\|M_2^n\|_1$ could not converge to 0). Therefore $\|M^n\|_1 \leq \rho^n \|M_1\| + C$. But since $\rho \leq 1$ we have $\|M^n\|_1 \leq \|M_1\|_1 + C$. Let $t = \|M_1\|_1 + C$.

Consider the branching process with an initial vector z^0 of entities from $C_{r,j}$, and let z^m be the vector of the number of entities after dm stages. As remarked before the lemma, for all m, the expectation $\mathbb{E}(\sum_{j=1}^n z_j^m) = \sum_{j=1}^n z_j^m = \sum_{j=1}^n (z^0(B_{r,j}(1))^m)_j \leq nt'$ for some fixed t'. Thus, by the Markov inequality, for all m, and all $k > 0$, $Pr(\sum_{i=1}^n (z^m)_i \geq knt') \leq 1/k$. Thus, also, for all m and k, $Pr(\forall m' \geq m \sum_{i=1}^n (z^{m'})_i \geq knt') \leq 1/k$. Since this is true for any m, we have $Pr(\exists m \text{ s.t. } \forall m' \geq m \sum_{i=1}^n (z^{m'})_i \geq knt') = \lim_{m\to\infty} Pr(\forall m' \geq m \sum_{i=1}^n (z^{m'})_i \geq knt') \leq 1/k$. Since $k > 0$ is arbitrary, we have $Pr(\forall k > 0 \ \exists m \text{ s.t. } \forall m' \geq m, \ \sum_{i=1}^n (z^{m'})_i \geq knt') = \lim_{k\to\infty} Pr(\exists m \text{ s.t. } \forall m' \geq m \sum_{i=1}^n (z^{m'})_i \geq knt') = \lim_{k\to\infty} 1/k = 0$

We write this last probability, which is 0, as $Pr(\sum_i (z^m)_i \to \infty)$, i.e., this denotes the probability that the sum $\sum_i (z^m)_i$ diverges and is unbounded in the limit as $m \to \infty$. Likewise, let $Pr(\sum_i (z^m)_i \to 0)$ be the probability of the event that we will eventually reach the 0 vector.

By Lemma 8.3, the probability that some vector z' occurs infinitely often in the trajectory $\langle z^i \mid i = 0, 1, 2, \ldots \rangle$ is 0. Therefore, $Pr(\sum_i (z^m)_i \to 0) + Pr(\sum_i (z^m)_i \to \infty) = 1$, because either the process diverges, or it reaches 0, or otherwise it repeats some vector z' infinitely often. But we already established that $Pr(\sum_i (z^m)_i \to \infty) = 0$. Thus $Pr(\sum_i (z^m)_i \to 0) = 1$. Since the initial vector z^0 was arbitrary, it could be chosen to be the unit vector e_i, with 1 in position i and 0 everywhere else. We can thus conclude that for all $x_i \in C_{r,j}$, $q_i^* = 1$. ∎

As we have already observed, since all probabilities in C_r depend on all others, if the probability associated with some node in C_r is exactly 1 then the probability associated with all nodes in C_r is also exactly 1. Thus, if $\rho(B_{r,j}(1)) \leq 1$ for some $C_{r,j}$, then $\rho(B_{r,j}(1)) \leq 1$ for all $C_{r,j}$. Thus $QB_r(1)^d Q^{-1}$ also has maximum eigenvalue ≤ 1, because its eigenvalues are the union, over all j, of eigenvalues of each $B_{r,j}(1)$. But since the eigenvalues of $QB_r(1)^d Q^{-1}$ are d'th powers of the eigenvalues of $B_r(1)$, then ρ_r, the maximum eigenvalue $B_r(1)$, must also be ≤ 1. ∎

We next explain how one can check for a nonnegative matrix B whether $\rho(B) \leq 1$ in polynomial time using linear programming. This follows from well known variational characterizations of the spectral radius of a nonnegative matrix:

Lemma 8.5 *(see, e.g., [HJ85] Theorem (8.3.2)) For B an $n \times n$ nonnegative matrix, if there exists a vector $x \geq 0$, $x \neq 0$ such that $Bx \geq \alpha x$, for some $\alpha \in \mathbb{R}$, then $\rho(B) \geq \alpha$.*

Lemma 8.6 *Consider a square nonnegative matrix B with at most n rows and having rational entries with at most l bits each. If $\rho(B) > 1$ then $\rho(B) \geq 1 + 1/2^m$ where $m = poly(n, l)$ and $poly(n, l)$ is some polynomial in n and l.*

Proof. If $\rho(B) > 1$ then there is a nonzero vector $u \geq 0$ such that $Bu = ru$, with $r > 1$. Suppose I' is the set of indices i with $u_i > 0$ and let B' be the corresponding

square submatrix $B_{I',I'}$ of B. The LP $B'x \geq x + 1$, $x \geq 0$ has a solution (scale $u[I']$ appropriately). Therefore, it has a rational solution with at most $m = poly(n, l)$ bits, hence its entries are at most 2^m. This solution, together with $u_i = 0$ in the rest of the indices satisfies $Bu \geq (1 + 1/2^m)u$, $u \geq 0$, $u \neq 0$. It follows from Lemma 8.5 that $\rho(B) \geq (1 + 1/2^m)$. ■

The following proposition is now immediate, and shows that we can using Linear Programming to decide in P-time whether $\rho(B(1)) \leq 1$.

Proposition 8.7 *$\rho(B) \leq 1$ if and only if the following LP constraints are not feasible: $Bx \geq (1 + 1/2^m)x$, and $x \geq 0$, $\sum_i x_i = 1$.*

Proof. Follows from Lemmas 8.5 and 8.6. Note that since the constraint $Bx \geq (1+1/2^m)x$ is homogeneous in x, we can scale x appropriately and replace the constraint $x \neq 0$ with $\sum_i x_i = 1$. ■

There is an analogous algorithm to that of Figure 8 for classifying all the nonterminals of a SCFG or all the types of a multitype branching process with respect to their qualitative termination directly, without translating them first to a 1-exit RMC. We first find the useless nonterminals, i.e. those that cannot derive any terminal strings; these are exactly the nonterminals S_i with termination probability $p(S_i) = 0$. Then we form the system of equations $x = P(x)$ that has one variable for each other nonterminal of the SCFG, decompose it into strongly connected components and process it as in Figure 8, to determine which nonterminals have $p(S_i) = 1$ and which have $p(S_i) < 1$. The algorithm is completely analogous and is left to the reader.

We now argue that the correctness of the simpler algorithm in Figure 7 for determining the consistency of a given SCFG with a given start nonterminal follows from the correctness we have established for the 1-pass algorithm in Figure 8. To see this, consider the algorithm in Figure 7. It is clear that if there are any "useless" nonterminals reachable from S_1 (step 2), then the probability of termination from S_1 is < 1. Suppose this is not the case. Then every nonterminal reachable from S_1 has nonzero probability of termination. In the algorithm in Figure 8, this means the preprocessing step (step 2) does not remove any nonterminals (i.e., variables) reachable from S_1 (i.e., x_1). Note that step 3 of the consistency algorithm in effect will perform the last step (step 4(d)) of the 1-pass algorithm, but not on one SCC at a time bottom-up, but rather on all variables reachable from x_1. But it can be shown easily that $\rho(B(1)) \leq 1$ if and only if $\rho(B_r(1)) \leq 1$ for every SCC C_r of the dependency graph of $x = P(x)$ that is reachable from x_1, where $B_r(1)$ is the associated moment matrix of C_r. For the one direction, if $\rho(B_r(1)) > 1$ for some SCC C_r, then we can take an eigenvector of $B_r(1)$ with eigenvalue $\rho(B_r(1)) > 1$ and pad it with 0's to obtain an eigenvector of $B(1)$. For the other direction, if $\rho(B(1)) > 1$, take a corresponding eigenvector u of $B(1)$, and let C_r be a lowest SCC that contains a nonzero coordinate of u; then the restriction of u to C_r is an eigenvector of $B_r(1)$ with the same eigenvalue $\rho(B(1)) > 1$.

8.2 Bounded RMCs

In this section we show that we can compute the vector of termination probabilities for a *bounded RMC* in polynomial time. Recall that a bounded RMC is one that has a constant

number of components, each with a constant number of entries and exits; the components themselves can be arbitrarily large.

Theorem 8.8 *Given a bounded RMC, we can decide in polynomial time whether* $\mathrm{LFP}(P) = \mathbf{c}$*, or whether* $\mathbf{q}^*_k \, \Delta c_k$*, for any* $k = (u, ex)$*, and 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 to MT-BPs with a bounded number of types (provided the multisets on the right-hand sides of the rules are listed explicitly).*

Proof. We will reduce our problem to the existential theory of the reals with a bounded number of variables. This is especially easy to do for the special case of SCFGs with a bounded number of nonterminals and MT-BPs with a bounded number of types. Given a SCFG G with d nonterminals $S_1, \ldots, S_d$ (or a MT-BP with d types), use a variable x_i for each nonterminal S_i, $i = 1, \ldots, d$. The termination probabilities q^*_i for the different nonterminals S_i are the least nonnegative solution to a system that contains one equation $x_i = h_i(x)$ for each $i = 1, \ldots, d$, where h_i is a polynomial that contains a term for each rule of G with left-hand-side S_i; if the rule has probability p and the right-hand-side of the rule contains ℓ_j occurrences of nonterminal S_j, $j = 1, \ldots, d$, then the corresponding term in h_i is $p \prod_j x_j^{\ell_j}$. To determine whether $q^*_i \leq c$, for a given rational c, append to these equations the constraints $x_j \geq 0$, $j = 1, \ldots, d$, and $x_i \leq c$, and use the polynomial-time decision procedure for $\boldsymbol{ExTh}(\mathbb{R})$ with d variables.

For general bounded RMCs, it is not quite as simple to reduce the problem to a bounded number of variables. Note that even for single-entry single-exit RMCs with a bounded number of components, the transformation to SCFGs given in Section 2 cannot be used because it introduces one nonterminal for every vertex, hence the number of nonterminals is unbounded.

Let A be a bounded RMC. First, we determine those entry-exit pairs (en, ex) (of the same component) for which the corresponding probability $q^*_{(en,ex)}$ is 0. Recall (Theorem 2.2) that this can be done in polynomial time for general RMCs. Let D be the set of remaining entry-exit pairs (en, ex), with $q^*_{(en,ex)} > 0$. We use a variable $x_{(en,ex)}$ for each pair $(en, ex) \in D$; note that this is a bounded number d of variables. We will construct a system of equations in these variables by essentially eliminating all other variables from the fixpoint system $x = P(x)$ of the RMC given in Definition 1, in a manner that is efficient and does not blow up the size of the system exponentially.

For each component A_i, we replace each box b of A_i by a set of labelled edges from the call ports to the return ports of the box: if en is an entry of the component to which box b is mapped and ex is an exit of the component then there is an edge directed from the call port (b, en) of b to the return port (b, ex) iff $(en, ex) \in D$ and in this case the edge is labelled $x_{(en,ex)}$; if $(en, ex) \notin D$ then we do not include an edge from (b, en) to (b, ex). The resulting graph is an ordinary labelled graph $B_i = (Q_i, E_i)$ (no boxes) whose edges are labelled with explicit probabilities or variables. If we plugged in the actual probabilities for the variables, we would have an ordinary finite Markov chain. (For some of the call ports, the sum of the probabilities of the outgoing edges may be less than 1; if we wished we could add transitions to a new dead absorbing state with the remaining probability, but this is not necessary for the calculations).

As is well known, we can compute the reachability probabilities in a Markov chain by setting up and solving a linear system of equations, whose coefficients are the transition

probabilities. Regarding the variable-labels as parameters, distinct from the reachability variables, we can set up the linear systems and solve them symbolically. More specifically, for every exit ex of B_i we first determine the set Q_i^0 of vertices of B_i that cannot reach ex; note that these are precisely the vertices u for which $q^*_{(u,ex)} = 0$, while the remaining set $Q_i \setminus Q_i^0$ contains the vertices u for which $q^*_{(u,ex)} > 0$. For each vertex $u \in Q_i \setminus Q_i^0$ we have a variable $y_{(u,ex)}$. (We could have included variables also for the other vertices $u \in Q_i^0$ and set them to 0, but it is not necessary.) Regarding the x-variables as symbols, we form a linear system L_{ex} of equations in the variables $y_{(u,ex)}$ as follows. If $u = ex$ then $y_{(ex,ex)} = 1$. For every other vertex u of $Q_i \setminus Q_i^0$, the corresponding equation is $y_{(u,ex)} = \sum_v \ell_{uv} y_{(v,ex)}$, where the summation ranges over all $v \in Q_i \setminus Q_i^0$ such that there is an edge (u, v) in B_i and ℓ_{uv} is the label of the edge (u, v), i.e. ℓ_{uv} is either the transition probability p_{uv} or some $x_{(en',ex')}$.

Thus, we have a separate linear system L_{ex} for each exit ex of each component. Note that if we substitute in L_{ex} positive values for the symbols $x_{(en',ex')}$ such that the sum of the labels out of each call port are at most 1, then the resulting system has a unique solution in the variables $y_{(u,ex)}$. This follows from standard facts about computing reachability probabilities for finite Markov chains. If the values $x_{(en',ex')}$ that we substitute are equal to the termination probabilities $q^*_{(en',ex')}$, then the unique solution gives precisely the set of termination probabilities $q^*_{(u,ex)}$.

Consider each system L_{ex}. Regard the x's as symbolic transition probabilities and solve the system for the y variables. Every reachability probability $y_{(u,ex)}$ is expressed as a function $h_{(u,ex)}(x)$ of the variable-labels x. By Cramer's rule, each function h is the ratio of two determinants of matrices whose entries are the original probabilities on the transitions of the RMC and the variables $x_{(en',ex')}$. That is, every vertex-exit probability is expressed as a ratio $h(x) = f(x)/g(x)$, where $f(x)$ and $g(x)$ are determinants of matrices $F(x), G(x)$ involving the variable-labels x. Thus, f, g are polynomials of total degree at most n, the number of vertices of A_i. Since we have a bounded number d of entry-exit pairs, and hence variables, each polynomial f, g has a polynomial number of terms, less than $(n+1)^d$. Furthermore, it can be easily seen that the coefficients of the polynomials are rational numbers whose bit complexity (of numerator and denominator) is bounded by a polynomial in the bit complexity b of the input probabilities. To see this, note that by the definition of the determinant, the determinant $f(x) = det(F(x))$ is the sum of $n!$ terms, where each term is the product of n entries of the matrix, i.e. either given transition probabilities or variables x. Thus, the coefficient of each monomial of $f(x)$ is the sum of at most $n!$ terms, where each term is the product of at most n given rational numbers whose numerator and denominator have b bits. We can make all the terms have a common denominator, for example the product of all the denominators of the (non-variable) entries of the matrix $F(x)$. Then the denominator has at most bn^2 bits, and the total numerator is the sum of at most $n!$ terms, each with at most bn^2 bits, hence the numerator has at most $bn^3 \log n$ bits.

The polynomials f, g for each vertex-exit pair can be computed in polynomial time. Suppose that $f(x) = det(F(x))$, where $F(x)$ is a matrix whose entries are rationals and variables from x. We can compute $f(x)$ explicitly using interpolation as follows. Plug in $n+1$ different values for each of the variables, for example 0, 1, .., n. For each of the $(n+1)^d$ tuples t of variable values, compute $det(F(t))$, which is a rational number of polynomial bit complexity. Then set up a linear system whose unknowns are the coefficients of the multivariate polynomial $f(x)$ and solve it. The system has $O(n^d)$ variables

and equations, and all the coefficients are rationals with polynomial bit complexity, hence it can be solved in polynomial time. It can be easily seen that the system has a unique solution. We include briefly the argument. In the univariate case (d=1), the matrix of the system is a Vandermonde matrix and is invertible (this is the usual univariate interpolation property: no two distinct polynomials of degree n can agree on $n+1$ points, and there exists a polynomial interpolant of degree n for any $n+1$ points). The general d case can be shown by induction: if the variables are $x_1, \ldots, x_d$, write $f(x)$ as a polynomial in x_d whose coefficients are polynomials in $x_1, \ldots, x_{d-1}$: $f = f_0 x_d^n + f_1 x_d^{n-1} + \ldots + f_n$ where each f_i is a polynomial in $x_1, \ldots, x_{d-1}$ of total degree at most i. For any fixed tuple τ of the first $d-1$ variables, $f(x)$ becomes a univariate polynomial in x_d. Since we plug in $n+1$ different values of x_d together with τ, the resulting values of f determine uniquely the univariate polynomial $f(\tau, x_d)$ in x_d. Hence they determine uniquely each $f_i(\tau)$. By induction hypothesis, the values $f_i(\tau)$ for all tuples τ determine uniquely the polynomial $f_i(x_1, \ldots, x_{d-1})$. Therefore, the set of tuples on which we evaluate f determines f uniquely.

Thus, for every entry-exit pair $(en, ex) \in D$ of each component we can compute explicitly the corresponding rational function $h_{(en,ex)}(x) = f_{(en,ex)}(x)/g_{(en,ex)}(x)$ for the corresponding variable $y_{(en,ex)}$. Hence the entry-exit probabilities $q^*_{(en,ex)}$ satisfy the set of polynomial equations $f_{(en,ex)}(x) = g_{(en,ex)}(x) \cdot x_{(en,ex)}$. Let $C(A)$ be the set consisting of these constraints and the constraints $x_{(en,ex)} > 0$ for every entry-exit pair $(en, ex) \in D$ of each component, and $\sum_{ex \in Ex_i} x_{(en,ex)} \leq 1$ for each entry en of each component A_i. We claim that the system $C(A)$ of constraints has a unique minimal solution and this solution is precisely the vector of nonzero entry-exit probabilities $q^*_{(en,ex)}$.

First, it is clear that the probabilities $q^*_{(en,ex)}$ satisfy these constraints. On the other hand, suppose that r is any other solution of $C(A)$. Extend r to all vertex-exit pairs (u, ex) as follows. For pairs $(u, ex) \in Q_i^0$ we let $r_{(u,ex)} = 0$. For pairs $(u, ex) \in Q \setminus Q_i^0$, we substitute in the linear system L_{ex} the values of r for the edge labels and solve in the resulting linear system $L_{ex}(r)$ for the variables $y_{(u,ex)}$, i.e., we let $r_{(u,ex)} = f_{(u,ex)}(r)/g_{(u,ex)}(r)$; note that there is a unique solution to the system $L_{ex}(r)$, and these expressions are well-defined (the denominator is nonzero). The extended vector r is also a nonnegative fixpoint to the system $x = P(x)$ of the RMC A and therefore $r \geq q^*$. It follows that the system $C(A)$ has a unique minimal solution, which is the vector of entry-exit probabilities $q^*_{(en,ex)}$. We can then answer questions, comparing such a probability with a given rational number c by appending the constraint $x_{(en,ex)} \leq c$ or $x_{(en,ex)} < c$, and using a polynomial-time procedure for the existential theory of the reals with a bounded number, d, of variables.

If we want to determine whether another (nonzero) vertex-exit probability $q^*_{(u,ex)}$ is less than c then we add to $C(A)$ the constraints $f_{(u,ex)}(x) = g_{(u,ex)}(x) \cdot x_{(u,ex)}$ and $x_{(u,ex)} < c$. Note that by the above argument, if r is any solution to $C(A)$, then its extension to all vertex-exit pairs as above dominates q^*, i.e. $f_{(u,ex)}(r)/g_{(u,ex)}(r) \geq q^*_{(u,ex)}$. If we want to compare the termination probability starting from a vertex u of A_i with a rational c, then we add the constraints $f_{(u,ex)}(x) = g_{(u,ex)}(x) \cdot x_{(u,ex)}$ for all $ex \in Ex_i$ and the constraint $\sum_{ex \in Ex_i} x_{(u,ex)} \leq c$ or $\sum_{ex \in Ex_i} x_{(u,ex)} < c$. ∎

8.3 Linear Recursion, Hierarchical RMCs, and Piecewise Linear Recursion

We will analyze first linear RMCs, and then a more general class of 'piecewise linear RMCs' which includes hierarchical RMCs. Recall from the definitions in Section 2.1 that a *linear* RMC is one with the property that there is no positive probability path in any component from any return port to a call port in the same component (via transitions of that component only). This is a simple syntactic property that can easily be checked in linear time for an RMC. The class of linear RMCs inherits all the good properties of finite Markov chains, as we will show. That is, the reachability probabilities are rational, they have polynomial bit complexity and can be computed efficiently. Furthermore, qualitative questions do not depend on the precise values of the transition probabilities. The system of equations $x = P(x)$ for a linear RMC is still nonlinear. However, as we will see, it can be decomposed into a sequence of linear systems, and solved efficiently. We will show the following:

Theorem 8.9 *In a linear recursive Markov chain A, all the termination probabilities $q^*_{(u,ex)}$ are rational, with polynomial bit complexity and can be computed exactly in polynomial time. Furthermore, the qualitative problem of classifying which probabilities are 0, 1 or in-between does not depend on the values of the transition probabilities, but only on the structure of the RMC A and can be solved in time $O(|A|\xi\theta)$, where $\xi = \max_i\{|Ex_i|\}$ is the maximum number of exits of the components, and $\theta = \max_i \min\{|En_i|, |Ex_i|\}$.*

Proof. Let A be a linear RMC. First, we can determine all the vertex-exit pairs (u, ex) such that $q^*_{(u,ex)} = 0$ as in Theorem 2.2 and remove the corresponding variables from the system $x = P(x)$. Regard each component A_i as a directed graph on its set of vertices Q_i and let R_i be the subset of vertices that can reach a call port in this graph and $S_i = Q_i - R_i$ the remaining set of vertices. Let $R = \cup R_i$ and $S = \cup S_i$. Since A is a linear RMC, S includes all the exits of the components and all the return ports of all the boxes, but no call ports. Note that S is successor-closed: all successors of a vertex in S are also in S. The set S of vertices induces an ordinary finite Markov chain, where all the exits of the components are absorbing states. The corresponding subsystem of equations of the system $x = P(x)$ is a linear system. As is well known from the theory of Markov chains, after we set to 0 those variables $x^*_{(u,ex)}$ in this subsystem for which the reachability probability $q^*_{(u,ex)} = 0$, the remaining subsystem has a unique solution, which is precisely the set of reachability probabilities $q^*_{(u,ex)}$, $u \in S$.

Now take the remaining subsystem of the system $x = P(x)$ corresponding to the vertices in R and substitute in the right-hand side the values of all the variables $x^*_{(u,ex)}$, $u \in S$, and also set to 0 the variables $x^*_{(u,ex)}$, $u \in R$ for which $q^*_{(u,ex)} = 0$. Note that the resulting subsystem is a linear system because all return ports are in S and every nonlinear term of $x = P(x)$ includes the variable $x_{((b,ex'),ex)}$ of a return port (b, ex). That is, the resulting subsystem has the form $x' = Bx' + h$ where x' is the vector of (nonzero) variables of the form $x^*_{(u,ex)}$, $u \in R$ such that $q^*_{(u,ex)} > 0$, B is a nonnegative (constant) matrix and h is a nonnegative vector. However, this subsystem may itself not correspond to the linear system associated with a finite Markov chain. In particular, the matrix B may not be (sub)stochastic, and may have rows that sum to greater than 1. We can nevertheless show that this subsystem has a unique solution, which is then of course the vector q' of the corresponding nonzero probabilities $q^*_{(u,ex)}, u \in R$. One way to show this is as follows: by

Theorem 3.1, if we start with $x' = 0$ and repeatedly apply the transformation $x' = Bx' + h$, the resulting vector $x'^k = (I + B + B^2 + \ldots B^{k-1})h$ will converge to the true probability vector q' as $k \to \infty$; note that the statement of the theorem remains valid if we substitute the true probabilities for some of the variables and restrict the system to the remaining variables. On the other hand $q' = Bq' + h$ and iterating this equation k times, we have $q' = B^k q' + (I + B + B^2 + \ldots B^{k-1})h$. As $k \to \infty$ the second term on the right hand side tends to q', hence the first term $B^k q'$ must tend to the 0 vector. Since q' is a strictly positive vector (recall, we eliminated all the zero variables) and B is a nonnegative matrix, if follows that B^k converges to the all-zero matrix. This implies that there is a unique solution: If r is any solution, i.e. $r = Br + h$, then $r = B^k r + (I + B + B^2 + \ldots B^{k-1})h$. The right-hand side converges to $0 + q'$ as $k \to \infty$, therefore $r = q'$.

Thus, to summarize, we can compute the vector q^* of termination probabilities by first determining those coordinates that are 0, second, solving a linear system of equations for the coordinates corresponding to vertices $u \in S$, and third substituting the computed values and solving a second linear system for the vertices $u \in R$. Clearly this can be done in polynomial time, and all the computed probabilities are rationals with polynomial bit complexity. This establishes the quantitative part of the theorem.

For the qualitative problem, the transition probabilities of the RMC are not relevant and we do not need to solve any linear system. For the vertices $u \in S$, we can determine whether the termination probability $q^*_{(u,ex)}$ to each exit ex, or the total termination probability q^*_u, is 0, 1 or in-between as in the case of ordinary finite Markov chains: We decompose the subgraph induced by the set of vertices S into strongly connected components (scc's). Note that every exit is a bottom scc, i.e. an scc that has no outgoing edges. Let us call the vertices that have nonzero probability of not terminating, i.e. $q^*_u < 1$, *survivors*. For a vertex $u \in S$, its termination probability $q^*_u < 1$ iff u can reach a bottom strongly connected component that is not an exit. On the other hand $q^*_u = 0$ if u cannot reach any exit. Furthermore, for an exit ex in the component of u, $q^*_{(u,ex)} = 1$ iff the only bottom scc that u can reach is ex, and $q^*_{(u,ex)} = 0$ iff u cannot reach ex.

It remains to classify the probabilities q^*_u and $q^*_{(u,ex)}$ for vertices $u \in R$. We can determine the survivor vertices of R as follows. Construct the following ordinary directed graph H on the set of vertices Q of the RMC A. The graph H contains all the edges of A. In addition H contains for every call port (b, en) and return port (b, ex) of the same box b of A such that $q^*_{(en,ex)} > 0$ a "summary edge" from (b, en) to (b, ex), and also H contains for every call port (b, en) of every box b a "calling" edge from (b, en) to the entry en of the component to which b is mapped. Let D be the union of the set of vertices of S that are survivors and the set of call ports that have no outgoing summary edges.

Lemma 8.10 *A vertex u is a survivor, i.e. $q^*_u < 1$, if and only if u can reach in H a vertex in D.*

Proof. Suppose that u can reach in H a vertex v in D. Then the Markov chain M_A starting at state $\langle \epsilon, u \rangle$ will follow with nonzero probability a corresponding path that reaches a state of the form $\langle \beta, v \rangle$. If v is a survivor vertex in S then there is nonzero probability that the path will not be able to reach any exit of the component of v, and hence will not terminate. If v is a call port (b, en) with no outgoing summary edges, then clearly from that point on the RMC will surely not exit. Thus, in either case there is nonzero probability that the RMC will not terminate.

Conversely, suppose that u does not have any path in H to any node of D. Consider the bottom scc's of H. Note that every vertex of S can only reach in H vertices of S, thus if a bottom scc contains a vertex in S then all the vertices must be in S, and the scc is either an exit or a set of survivor vertices. If a bottom scc contains a vertex in R, then it must also contain a call port $v = (b, en)$ (because every vertex in R can reach a call port); since the bottom scc does not contain any vertex in S, it cannot contain any return port of the box b, hence the call port v has no outgoing summary edges and hence it is in D. Thus, we see that in either case every bottom scc of H that is not an exit contains a node in D.

Consider a trajectory of the RMC starting at vertex u of R, i.e. a path of M_A starting at $\langle \epsilon, u \rangle$. Since u cannot reach in H any node in D, with probability 1 the path will eventually reach either some exit of u's component or a call port of a box. Every time the path reaches the call port (b, en) of a box and initiates a new recursive call, since the call port has an outgoing summary edge, there is nonzero probability that the path will eventually reach a return port (b, ex) of the box; if this happens then the process from that point on will terminate with probability 1 for the following reason. Suppose that the state of the Markov chain M_A is $\langle b_1 \dots b_k, (b, ex) \rangle$. Note that if the path π of M_A entered these boxes through the call ports $(b_1, en_1), \dots (b_k, en_k), (b, en)$ respectively, then H contains a path from u to (b, en_1) to en_1 to (b, en_2) to ... to (b, en) to (b, ex). Then u can reach in the graph H the vertex (b, ex) which is in S and hence by assumption vertex (b, ex) is not a survivor. Thus a path from $\langle b_1 \dots b_k, (b, ex) \rangle$ will reach with probability 1 an exit $\langle b_1 \dots b_k, ex_k \rangle$ of its component, and then in one step it will reach a state $\langle b_1, \dots, b_{k-1}, (b_k, ex_k) \rangle$. Since H contains a path from u to (b_k, en_k) and there is a summary edge from (b_k, en_k) to the return port (b_k, ex_k), it follows again that since (b_k, ex_k) is in S it is not a survivor, and thus the path will go on with probability 1 to exit the box b_{k-1}, i.e., to reach the global state $\langle b_1, \dots, b_{k-2}, (b_{k-1}, ex_{k-1}) \rangle$, and so forth until it reaches a global exit $\langle \epsilon, ex' \rangle$ of the original component that contains the vertex u.

Every time the path enters a new box (initiates a new call) there is nonzero probability that it will exit the box and if this happens then it will go on to terminate almost surely. It follows that the probability that the path enters a new box infinitely often is 0, i.e., with probability 1 the path will only enter a finite number of boxes and hence it will terminate. ■

Once we have determined which vertices are survivors, we can determine easily whether a particular vertex-exit probability $q^*_{(u,ex)}$ is 1 or not: $q^*_{(u,ex)} = 1$ iff $q^*_u = 1$ and $q^*_{(u,ex')} = 0$ for all other exits ex' of the component of u.

Clearly we can compute in linear time the sets of vertices R and S, decompose the subgraph induced by S into strongly connected components and determine the vertices of S that are survivors. We compute then the entry-exit pairs en, ex such that $q^*_{(en,ex)} > 0$, we add the summary edges and construct the graph H. If every component of the RMC has at most ξ exits, then the graph H has size at most $|A|\xi$. We can then compute all the vertices in H that can reach a vertex of D in time $O(|H|) = O(|A|\xi)$, and thereby determine all the survivor vertices, and from them classify qualitatively as above all the termination probabilities $q^*_{(u,ex)}$. This establishes the second part of the theorem. ■

Piecewise Linear and Hierarchical RMCs

Linear RMCs can be extended to define the more general notion of *piecewise-linear RMCs*, which contains the class of hierarchical Markov chains, and for which a number of the results for RMCs carry over. The piecewise linear class is defined as follows. From the nonlinear system $\mathbf{x} = P(\mathbf{x})$ of an RMC, A, construct the *dependency graph* G_A as in Section 6: the nodes are those variables $x_{(u,ex)}$ such that $q^*_{(u,ex)} \neq 0$, plus an additional node labeled "1", and whose edges include edge (x_i, x_j) iff x_j appears in $P_i(\mathbf{x})$, and edge $(x_i, 1)$ iff $P_i(\mathbf{x}) \equiv 1$. Now, we can decompose the graph G_A into strongly connected components (SCCs), and we can also form the DAG, D_A, of SCCs of this decomposition. We call the original RMC A a *piecewise-linear RMC* if, for every SCC C, the subsystem of $\mathbf{x} = P(\mathbf{x})$ induced by the variables in C (treating variables in other SCCs as constant) is linear; that is, for every SCC, C, of G_A and every $Type_{call}$ variable $x_{((b,en),ex)}$ of C with corresponding polynomial $P_{((b,en),ex)}(\mathbf{x}) \equiv \sum_i x_{(en,ex'_i)} x_{((b,ex'_i),ex)}$, where the sum ranges over all exits ex'_i of the component that contains entry en, it must be the case that for each i, either $x_{(en,ex'_i)}$ is not in C or $x_{((b,ex'_i),ex)}$ is not in C (or both are not in C).

Clearly, all linear RMCs are also piecewise linear. It is easy to see also that all Hierarchical Markov Chains (HMCs) are piecewise-linear. Recall that a RMC A is *hierarchical* if the call graph of A is acyclic. The call graph $CG(A)$ has one node $i = 1, \ldots, k$ for each component A_i of A and has an edge (i, j) if a box of A_i is mapped to A_j. Every nonlinear equation of the equation system S_A for A has the form $x_{((b,en),ex)} = P_{((b,en),ex)}(\mathbf{x}) \equiv \sum_i x_{(en,ex'_i)} x_{((b,ex'_i),ex)}$ where b is a box of a component A_l. The box b is mapped to a lower component A_m that cannot reach A_l in $CG(A)$ since $CG(A)$ is acyclic. It follows that the variables $x_{(en,ex'_i)}$ (which correspond to entry-exit probabilities of A_m) cannot reach the variable $x_{((b,en),ex)}$ (a vertex-exit probability of A_l) in the dependency graph G_A, i.e. the variables $x_{(en,ex'_i)}$ belong to a lower SCC. It follows that each SCC induces a linear system, and therefore A is piecewise linear.

It should be clear to the reader that if we solve the subsystems induced by the SCCs "bottom up" according to the DAG D_A of the SCC's, then solving each SCC will only involve solving a purely linear system. The encountered linear systems can in fact all be solved uniquely by matrix inversion; this can be shown by a similar argument to the one given for the linear RMC case (Theorem 8.9). We remark also that the decomposed Newton's method will handle piecewise linear RMCs quite effectively: after we decompose the system into strongly connected components, the subsystem for each SCC after we substitute the values for the variables in lower SCCs is linear, hence Newton will solve the system and converge in one iteration.

Some of the results about linear RMCs readily carry over to piecewise-linear RMCs via simple modifications of the same proofs. In particular, all termination probabilities are again rational values. However, although the probabilities are rational, they are no longer necessarily concise and may require exponentially many bits to encode if the height of the DAG D_A is unbounded. An example is given by the HMCs depicted in figure 5, which have size $O(n)$ but where termination probabilities are as small as $\frac{1}{2^{2^n}}$. However, if the height of D_A is bounded by a fixed constant, then the rational termination probabilities will all have size polynomial in the size of the input RMC and can be computed efficiently via a bottom-up analysis of the SCCs of G_A.

Moreover, for the qualitative questions the height of D_A is immaterial (since there are no quantities to compute whose size might explode), and by a modification of the proofs for linear RMCs, we can answer qualitative questions for piecewise-linear RMCs with the same complexity, via a bottom-up analysis. We summarize these facts as follows:

Theorem 8.11 *In a piecewise-linear recursive (and in particular, for a hierarchical) Markov chain A, all the termination probabilities $q^*_{(u,ex)}$ are rational, and moreover if the DAG D_A has height bounded by a fixed constant, then these rationals have polynomial bit complexity and can be computed exactly in polynomial time. Furthermore, the qualitative problem of classifying which probabilities are 0, 1 or in-between does not depend on the values of the transition probabilities, but only on the structure of the RMC and can be solved for any piecewise-linear RMC A, even when the height of D_A is unbounded, in time $O(|A|\xi\theta)$.*

If we have a unit-cost RAM model with exact rational arithmetic (i.e. algebraic operations on arbitrary rationals can be done in unit time) [BCSS98], we do not have to worry about the size of the numbers. Computing the termination probabilities of a piecewise linear recursive Markov chain (and in particular a hierarchical Markov chain) involves the solution of a sequence of linear systems, one for each SCC of the dependency graph, thus it can be done in polynomial time in that model. That is, the exact computation of the termination probabilities is in the class of problems solvable in polynomial time in the real model of [BCSS98] with rational constants. On the other hand, we saw in Theorem 5.3 that the decision problem for HMCs is at least as hard as `PosSLP` (even for 1-exit HMCs), and `PosSLP` is as hard as any other (Boolean) decision problem solvable in polynomial time in the real model of [BCSS98] with rational constants. The notation $BP(P^0_{\mathbb{R}})$ is used in [ABKPM06] for this class (BP stands for Boolean part: decision problems where the inputs are finite binary strings rather than real numbers). Thus, we have:

Corollary 8.12 *The decision problem for hierarchical and piecewise linear recursive Markov chains is complete (under P-time Turing reductions) for the class $BP(P^0_{\mathbb{R}})$ of decision problems solvable in polynomial time with unit-cost exact rational arithmetic.*

9 Relation to other models

9.1 Probabilistic Pushdown Systems

We now observe that RMCs and Probabilistic Pushdown Systems (pPDSs), studied in [EKM04, BKS05, EKM05] (see also [AMP99]) are essentially equivalent, and can be translated in linear time to one another. This is entirely analogous to the correspondence between (non-probabilistic) RSMs and Pushdown Systems (PDSs) (for which, see [ABE+05] for a detailed treatment).

The difference between RMCs and pPDSs (and between RSMs and PDSs) is analogous to the difference between a program that is broken down into a set of procedures that may call each other in a potentially recursive manner, and a non-recursive single-procedure program that uses a stack to perform an equivalent computation. Both models of course have the same computational power, but recursion (as well as hierarchy) and modularity (decomposing a program or system into smaller components) are some of the most powerful and important structuring primitives in computer science, useful for specifying large programs and modeling large systems. The RMC (and RSM) model incorporates these structuring primitives explicitly in the model, to facilitate modeling the behavior of systems that have probabilistic and recursive features. Similarly, Hierarchical Markov Chains are a useful way to structure and represent succinctly classes of large Markov chains.

RMCs are defined to closely resemble recursive probabilistic procedural programs, and to reflect their structure and parameters: components of an RMC correspond to procedures, entries and exits correspond to parameter- and return values passed to and from procedures. The translation to a pPDS loses some of this correspondence: the maximum number of exits of the RMC corresponds precisely to the number of states of the pPDS (in both directions of our translations), but there are no natural analogous parameters in a pPDS corresponding to the number of components or the number of entries of an RMC. In particular, Theorem 8.8, our polynomial-time algorithm for bounded RMCs, has no natural analog in terms of general pPDSs. One can state the theorem in terms of pPDSs that have additional structure, and this follows from our translation below, but to do so one would have to add structure to pPDSs in such a way that, in effect, amounts to redefining RMCs.

We now formalize the relationship between these models. A *probabilistic Pushdown System* (pPDS) $\mathcal{P} = (Q_P, \Gamma, \Delta)$ consists of a set of control states Q_P, a stack alphabet Γ, and a probabilistic transition relation $\Delta \subseteq (Q_P \times \Gamma) \times [0,1] \times (Q_P \times \{swap(\Gamma), swap\text{-}and\text{-}push(\Gamma \times \Gamma), pop\})$. That is, a transition has the form $((s,\gamma), p_{(s,\gamma),(s',\mathcal{C})}, (s',\mathcal{C}))$, where based on the control state and the symbol on top of the stack symbol (s,γ), with probability p, the machine updates the control state to s', and performs action $\mathcal{C}$ on the stack. If $\mathcal{C} = swap(\gamma')$ then the action swaps the top-of-the-stack symbol γ with a new symbol γ'. If $\mathcal{C} = swap\text{-}and\text{-}push(\gamma', \gamma'')$, then the action both swaps γ with γ' and then pushes γ'' on top of the stack. Note that the standard push transition $((s,\gamma), p_{(s,\gamma),(s',push(\gamma'))}, (s', push(\gamma')))$ can be written as $((s,\gamma), p_{(s,\gamma),(s',swap\text{-}and\text{-}push(\gamma,\gamma'))}, (s', swap\text{-}and\text{-}push(\gamma,\gamma')))$. Lastly, if $\mathcal{C} = pop$, then the action pops the stack. Each such transition has an associated probability $p_{(s,\gamma),(s',\mathcal{C})}$, and we assume that for each pair (s,γ) of control state and top of stack symbol, $\sum_{(s',\mathcal{C})} p_{(s,\gamma),(s',\mathcal{C})} = 1$. We assume there is a special stack symbol $\perp$ that marks the bottom of the stack, and constitutes the initial content of the otherwise empty stack, and we assume that when $\perp$ is popped the pPDS terminates. A stack with letter γ at the top and remaining content $\omega \in \Gamma^*$ (running from bottom to top) will be written $\omega\gamma$.

A pPDS $\mathcal{P}$ defines a countable Markov chain $M(\mathcal{P})$ in the obvious way. The states are pairs (w,s) with $s \in Q_P$ and $w \in \Gamma^*$, and, e.g., a pPDS transition $((s,\gamma), p, (s', swap(\gamma')))$ yields a transition $((w\gamma, s), p, (w\gamma', s'))$ in $M(\mathcal{P})$.

First note that a pPDS with only one control state is essentially equivalent to an SCFG. To see this, note that the stack alphabet Γ can act as the set of nonterminals, and the three distinct kinds of actions *swap*, *swap-and-push*, and *pop* correspond to a grammar rule with 1, 2, and 0 nonterminals on the right hand side, respectively. We saw that this is sufficient to define an SCFG in generalized Chomsky Normal Form.

For an RMC A and pPDS $\mathcal{P}$, let $M(A)$ and $M(\mathcal{P})$ denote their associated countable Markov Chain. We now show that for every pPDS $\mathcal{P}$ there is a RMC A such that $M(A)$ and $M(\mathcal{P})$ are "essentially equivalent", and vice versa. The result is completely analogous to similar results in [ABE+05] for non-probabilistic RSMs and PDSs.

To make the notion of "essentially equivalent" concrete, we define the following: in a (countable) Markov chain $\mathcal{M}$ with states t and t', let $t \overset{p}{\rightsquigarrow} t'$ mean that either there is a direct transition (t,p,t') in $\mathcal{M}$, or there is another state t'' such that (t,p,t'') and $(t'',1,t')$ are transitions of $\mathcal{M}$.

Theorem 9.1

1. *For every pPDS $\mathcal{P}$, there is an easily (linear time) computable RMC $A = \langle A_1 \rangle$ with one component, and an easily computable one-to-one mapping f from states of $M(P)$ to states of $M(A)$ such that for every transition (t, p, t') of $M(P)$, we have $f(t) \overset{p}{\rightsquigarrow} f(t')$ in $M(A)$. Moreover, $|A| = O(|\mathcal{P}|)$, and $|Ex_1| = |Q_P|$.*

2. *For every RSM $A = \langle A_1, \ldots, A_k \rangle$, with $k' = \max_i |Ex_i|$, there is a easily (linear time) computable PDS $\mathcal{P}$ and one-to-one mapping f from the states of $M(A)$ to the states of $M(P)$, such that for every transition (t, p, t') of $M(A)$, we have $f(t) \overset{p}{\rightsquigarrow} f(t')$ in $M(P)$. Moreover, $|P| = O(|A|)$, and $|Q_P| \leq k'$.*

Proof. First, given a pPDS P, we build RMC A. A has only one component, A_1.[7] A_1 has an entry $en_{(s,\gamma)}$ for every pair (s, γ) with s in Q_P and γ in Γ, such that there is a transition leaving (s, γ) in $\mathcal{P}$. It has one exit ex_s for every $s \in Q_P$. It also has one box b_γ associated with each stack symbol $\gamma \in \Gamma$ that plays a role in some transition of $\mathcal{P}$. All boxes are, obviously, mapped to A_1. The transitions of A_1 are as follows:

1. For every transition $((s, \gamma), p, (s', \mathit{pop}))$ in Δ, there is a transition $(en_{(s,\gamma)}, p, ex_{s'})$ in δ_1.

2. For every transition $((s, \gamma), p, (s', \mathit{swap}(\gamma')))$ in Δ, there is a transition $(en_{(s,\gamma)}, p, en_{(s',\gamma')})$ in δ_1.

3. For every transition $((s, \gamma), p, (s', \mathit{swap\text{-}and\text{-}push}(\gamma_1, \gamma_2)))$ in Δ, there is a transition $(en_{(s,\gamma)}, p, (b_{\gamma_1}, en_{(s',\gamma_2)}))$ in δ_1.

4. For every box exit (b_γ, ex_s) of each box b_γ, there is a probability-1 transition $((b_\gamma, ex_s), 1, en_{(s,\gamma)})$ in δ_1.

The intuition should be clear: each entry node $en_{(s,\gamma)}$ of A_1 corresponds to the configuration of $\mathcal{P}$ with control state s and top-of-stack γ. The remainder of the content of the pushdown stack of $\mathcal{P}$ (with the top element excluded) is coded in the call stack of A, with box b_γ on the call stack acting like γ on the pushdown stack. The size $|A|$ is $O(|P|)$, and the number of exits of A_1 is $|Q_P|$. Consider the mapping f from global states of $M(\mathcal{P})$ to global states of $M(A)$, defined by $f(\langle \gamma_1 \ldots \gamma_n, s \rangle) = \langle b_{\gamma_1} \ldots b_{\gamma_{n-1}}, en_{(s,\gamma_n)} \rangle$ It is not hard to check that the mapping f has the "transition preservation" property specified in the statement of the theorem.

In the other direction, given a RMC A, we now describe how to build a corresponding pPDS $\mathcal{P}$.

The stack symbols Γ of $\mathcal{P}$ correspond to all possible "locations" (j, v) in the RMC: j gives the index of the component, and v is either a node of A_j or a box b of A_j. The control states Q_P of $\mathcal{P}$ are $\{1, \ldots, |ex|\}$, where $|ex|$ is the maximum number of exit nodes of any component. For every transition (v, p, v') between vertices v and v' of a component A_j where v is not of the form (b, ex) and v' is not of the form (b', en), there is a transition $((1, (j, v)), p, (1, \mathit{swap}(j, v')))$ in $M(A)$. For every transition $(v, p, (b, e))$ within a component A_j where v is not of the form (b', ex) and where

[7] For convenience, we define A in such a way that there may also be incoming transitions into entry nodes of a component. This richer definition of RMCs causes no problems in any of our results. It is also not essential here and can be eliminated if we wished, using a slightly modified notion of $\rightsquigarrow$ in the statement of the theorem.

b is a box of A_j mapped to component $A_{j'}$ and e is an entry node of $A_{j'}$, there is a transition $((1,(j,v)),p,(1,\textit{swap-and-push}((j,b),(j',e))))$ in $\mathcal{P}$. For every exit node x_i of component A_j (where x_i is the i^{th} exit of A_j), there is a probability-1 transition $((1,(j,x_i)),1,(i,\textit{pop}))$ in $\mathcal{P}$. For every box b of A_j mapped to component $A_{j'}$, for every exit x_i of $A_{j'}$ and every transition $((b,x_i),p,v')$ with v' not of the form (b',e), there is a transition $((i,(j,b)),p,(1,\textit{swap}(j,v')))$ in $\mathcal{P}$. Finally, for every box b of A_j mapped to component $A_{j'}$, for every exit x_i of $A_{j'}$ and every transition $((b,x_i),p,(b',e))$ with e an entry node of A_k, there is a transition $((i,(j,b)),p,(1,\textit{swap-and-push}((j,b'),(k,e))))$ in $\mathcal{P}$.

The size of $\mathcal{P}$ is linear in the size of the recursive Markov chain A. Note that in the case where A is single-exit, $\mathcal{P}$ has only one control state, and is thus context-free.

It is not hard to check that the "equivalence" property stated in the theorem holds for the following mapping: for every global state $g = \langle b_1 \dots b_n, v\rangle$ of $M(A)$ where v is not an exit node, g is mapped to the global state $f(g) = \langle \gamma_1 \dots \gamma_{n+1}, 1\rangle$ of $M(\mathcal{P})$ with, for all $i \leq n$, $\gamma_i = (j_i, b_i)$ (where b_i is a box of component A_{j_i}), and $\gamma_{n+1} = (j_{n+1}, v)$ with j_{n+1} being the index of the machine called by box b_n; and for every global state $g = \langle b_1 \dots b_n, x_i\rangle$ of A where x_i is the i^{th} exit node of of a component A_j called by box b_n, $f(g) = \langle \gamma_1 \dots \gamma_n, i\rangle$ with, for all $i \leq n$, $\gamma_i = (j_i, b_i)$ (where b_i is a box of component A_{j_i}). This concludes the proof of Theorem 9.1. ■

9.2 Random Walks with "Back Buttons"

Fagin, et. al. ([FKK$^+$00]) studied a probabilistic model of web surfing which they call a *Random walk with "Back Buttons"*, or a *backoff process*. The model extends a finite Markov chain with a "back-button" feature: in each state, besides moving forward probabilistically to a new state, with some probability we can press the back button and return to the previous state from which the current state was entered. In this section we will show that this model corresponds to a proper subclass of 1-exit RMCs and of SCFGs.

Backoff processes can be defined formally by a simple restriction of pPDSs. (This is essentially identical to the definition in ([FKK$^+$00]) , only differing slightly in notation.) A backoff process $\mathcal{P}$ is a pPDS which has just 1 control state, and only $\textit{push}(\gamma')$ and *pop* transitions. In other words, no $\textit{swap}(\gamma')$ and no $\textit{swap-and-push}(\gamma',\gamma'')$ actions are available in transitions.

Note that the associated countable Markov chain generated by a backoff process has the property that the global states are determined by the stack content alone, and that the stack either grows by 1 (a push) or shrinks by 1 (a pop) in each transition (i.e., the stack can not stay the same height during a transition).

Fagin et. al. showed, among a number of results, how to use semidefinite programming to compute quantities such as termination probabilities for backoff processes (which they call "revocation" probabilities), by using the system of nonlinear equations associated with a backoff process. The primary focus of their work was to study and compute (Cesaro) limit distributions for "local states" of the backoff processes, where a local state is just given by the top stack symbol.

Another way to view backoff processes is as a restricted form of SCFG.

Namely, if our stack alphabet is $\Gamma = \{\gamma_1, \dots, \gamma_k\}$, we can associate to each stack symbol γ_i a nonterminal S_i, and we have the following rules in our grammar:

- A rule $S_i \xrightarrow{p} S_j S_i$, for every "push" transition $((1,\gamma_i),p,(1,\textit{push}(\gamma_j))) \in \Delta$.

- A rule $S_i \xrightarrow{p} \epsilon$, for every "pop" transition $((1, \gamma_i), p, (1, pop)) \in \Delta$.

It is easy to see that a random walk (i.e., trajectory) of a backoff process $\mathcal{P}$ corresponds precisely to a *leftmost (probabilistic) derivation* of its corresponding SCFG, with rules applied according to their probabilities during the derivation. Thus, the probability of termination starting from a given stack symbol in $\mathcal{P}$ corresponds exactly to the termination probability starting from the corresponding nonterminal in the SCFG.

It is also easy to see that the system of nonlinear equations $x = P(x)$ associated with termination probabilities for a backoff process (what [FKK$^+$00] call *revocation* probabilities), has the following form. We have one variable x_i for each stack symbol $\gamma_i \in \Gamma$, and the equation associated with x_i is:

$$x_i = \sum_{(\gamma_i, p_{\gamma_i,\gamma_j}, push(\gamma_j))} p_{\gamma_i,\gamma_j} x_i x_j + b_i$$

Here b_i is the probability of popping directly when the top of stack symbol is γ_i, i.e., the probability of the transition $((1, \gamma_i), b_i, (1, pop))$. (Of course, $b_i = 0$ if this transition does not exists.) The LFP solution of these systems $x = P(x)$, again, defines the termination probabilities for backoff processes.

These restricted nonlinear systems have an important property: if we replace $x_i = P_i(x)$ by the constraint $P_i(x) - x_i \leq 0$, and add the constraint $x \geq 0$, then all our constraints are convex. We can then find the LFP solution to this system by minimizing the linear objective $\sum_i x_i$ subject to these convex constraints. This enables Fagin, et. al., to apply powerful convex optimization techniques, in particular semidefinite programming (see, e.g., [GLS93]), to approximate the termination probabilities for backoff processes to within any given number of bits of precision in polynomial-time.

It is a very interesting question whether convex optimization techniques such those employed by Fagin, et. al., can be extended to RMCs, or even to 1-exit RMCs. Unfortunately, the richer systems $x = P(x)$ defined by RMCs and 1-exit RMCs are not convex in general.

10 Conclusions

We introduced Recursive Markov Chains, and studied basic algorithmic problems for their analysis, namely termination and reachability. We observed that the key to these problems is computation of the least fixed point solution of certain monotone polynomial systems of equations $x = P(x)$. A wide variety of techniques came into play, from the existential theory of the reals, theory of branching processes, numerical computing, combinatorial algorithms, etc. We showed that the qualitative and quantitative problems for the general class of RMCs are in PSPACE. We presented a more practical Newton-based method for numerical estimation, which we showed is guaranteed to converge monotonically to the desired probabilities, and we presented more efficient polynomial time algorithms for important special subclasses of RMCs. We also presented lower bounds, showing that both the qualitative (almost sure) termination and the approximation problems for RMCs are at least as hard as all of P-time in the powerful unit-cost RAM model with exact rational arithmetic (no integer division), and for hierarchical RMCs, the decision problem is complete for this class even in the 1-exit case.

We have built on the theory and algorithms developed in this paper, to extend our methods to algorithms for the verification of linear time properties of RMC's ([EY05a, YE05], see also [BKS05]). As shown there, the computation of termination probabilities lies at the heart of the qualitative and quantitative analysis of general properties of RMCs. Computation of other quantities for pPDS (equivalently, RMCs) is investigated in [BEK05, EKM05]. We have studied extensions of RMCs to Recursive Markov Decision Processes and Recursive Stochastic Games, where transitions are not purely probabilistic but can also be controlled by players that are trying to optimize their objectives [EY05c, EY06a, EY06b].

A number of questions remain open for the problems studied in this paper. The most important of course is, can the problems be solved in polynomial time? In view of our lower bounds, namely hardness results with respect to `SQRT-SUM` and `PosSLP`, a positive answer would imply that, as far as solving decision problems is concerned (and thus also, for problems with polynomial length output), unit-cost exact rational arithmetic can be simulated with polynomial overhead in the standard Turing machine model (equivalently, logarithmic cost RAM model). For the class of hierarchical Markov chains, this is necessary and sufficient for polynomial time solvability of the decision problem. Does this hold for the general class of RMCs, i.e. is the problem solvable in polynomial time with unit-cost exact rational arithmetic? Is there any other evidence of hardness, such as NP-hardness? Even if someone believes that, e.g., `PosSLP` is solvable in polynomial time (which looks unlikely at present), it is worth noting that even the much easier problem `EquSLP`, which asks to decide whether the output of an arithmetic circuit with integer inputs and gates $\{+, *, -\}$ is exactly equal to 0, and which is known to be decidable in probabilistic polynomial time (i.e., is in BPP), is P-time equivalent to the well known open problem of polynomial identity testing (see [ABKPM06]), and therefore (by results of [KI03]) a P-time algorithm for it (in the standard Turing model) would imply non-trivial circuit lower bounds.

We proposed a decomposed Newton method for approximating the reachability and termination probabilities. The method has been implemented and appears to perform well in practice [NS06, WE07]. Following an early version of this paper, there has been significant progress in investigating the rate of convergence of the decomposed Newton's method for RMCs and for monotone systems of polynomial equations [KLE07, EKL08]. These papers show that for a strongly connected system $x = F(x)$, after some initial number k_F of iterations, Newton's method gains at least one bit of precision per iteration. The upper bound shown on the initial number k_F of iterations is exponential in the input size; a polynomial bound is shown for a subclass of systems (that includes strongly connected systems arising from the back-button process [FKK$^+$00]). For non-strongly connected systems and RMCs, an example family of RMCs is provided in [KLE07] for which an exponential number of iterations in the size of the RMCs is needed to gain i bits of precision, but no general upper bound is known, in terms of the size of the RMC, for the number of Newton iterations needed for non-strongly connected systems. Note that an upper bound on the number of Newton iterations implies a similar bound in the unit-cost model, but not in the standard Turing machine (or logarithmic cost RAM) model of computation because the number of bits of the generated numbers may become exponential after a linear number of iterations.

For the important models of branching processes, stochastic context-free grammars, and 1-exit RMCs, we gave a polynomial time algorithm for the qualitative termination

problem. We also showed a lower bound for the decision problem. Can we approximate the termination probabilities in polynomial time? For the subclass of back-button processes, a polynomial time approximation algorithm using semidefinite programming was given in [FKK+00]. This does not seem to extend in any immediate way to the more general class of 1-exit RMCs (and MT-BPs, SCFGs), but perhaps there is a way to tackle the problem with more general convex optimization methods.

Finally, in a recent paper [EWY08] we have established a close relationship between another subclass of RMCs and a stochastic model studied for decades in queuing theory and performance evaluation, called Quasi-Birth-Death processes (QBDs) ([Neu81, LR99]). We have shown that discrete-time QBDs are equivalent in a precise sense to a subclass of RMCs (namely, RMCs with only one box, or equivalently, probabilistic 1-counter automata), and that a recently studied extension of QBDs, called (discrete-time) tree-like QBDs, is equivalent to RMCs. Furthermore, we showed in [EWY08] that for QBDs the decomposed Newton's method described in this paper converges in a polynomial number of iterations (in the size of the QBD and in the desired number of bits of precision) to the termination probabilities (also known as the QBD's G matrix). Thus for QBDs these quantities can be approximated to within a desired error in polynomial time in the unit-cost model of computation, but our results leave open whether such an approximation can be carried out in polynomial time in the standard Turing model of computation. On the other hand, the quantitative decision problem for termination probabilities of QBDs (is it $\geq p$) is `SQRT-SUM`-hard. See [EWY08] for details of these results and for a more extensive bibliography of the relevant literature on QBDs.

Acknowledgement: Research partially supported by NSF Grants CCF-04-30946 and CCF-07-28736.

References

[ABE+05] R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T. Reps, and M. Yannakakis. Analysis of recursive state machines. *ACM Trans. Program. Lang. Syst.*, 27(4):786–818, 2005.

[ABKPM06] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the complexity of numerical analysis. In *Proc. 21st IEEE Conference on Computational Complexity*, pages 331–339, 2006.

[AEY01] R. Alur, K. Etessami, and M. Yannakakis. Analysis of recursive state machines. In *Proc. of 13th Int. Conf. on Computer-Aided Verification*, pages 304–313, 2001.

[AMP99] S. Abney, D. McAllester, and F. Pereira. Relating probabilistic grammars and automata. In *Proc. 37th Ann. Meeting of Ass. for Comp. Linguistics*, pages 542–549. Morgan Kaufman, 1999.

[AN72] K. B. Athreya and P. E. Ney. *Branching processes*. Springer-Verlag, 1972.

[APY91] E. M. Arkin, C. H. Papadimitriou, and M. Yannakakis. Modularity of cycles. *Journal of the ACM*, 38:255–274, 1991.

[AV01] K. B. Athreya and A. N. Vidyashankar. Branching processes. In *Stochastic processes: theory and methods*, volume 19 of *Handbook of Statist.*, pages 35–53. North-Holland, 2001.

[AY01] R. Alur and M. Yannakakis. Model checking of hierarchical state machines. *ACM Trans. Program. Lang. Syst.*, 23(3):273–303, 2001.

[BCSS98] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, 1998.

[BEK05] T. Brázdil, J. Esparza, and A. Kucera. Analysis and prediction of the long-run behavior of probabilistic sequential programs with recursion. In *Proc. 46th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 521–530, 2005.

[BEM97] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Applications to model checking. In *Proc. 8th Int. Conf. on Concurrency Theory (CONCUR)*, pages 135–150, 1997.

[BGR01] M. Benedikt, P. Godefroid, and T. Reps. Model checking of unrestricted hierarchical state machines. In *Proc. 28th Int. Coll. Automata, Languages and Programming (ICALP)*, pages 652–666, 2001.

[BKS05] T. Brázdil, A. Kučera, and O. Stražovský. Decidability of temporal properties of probabilistic pushdown automata. In *Proc. 22nd Symp. on Theoretical Aspects of Comp. Sci. (STACS)*, 2005.

[BMS81] A. Bertoni, G. Mauri, and N. Sabadini. A characterization of the class of functions computable in polynomial time on random access machines. In *Proc. 13th ACM Symp. on Theory of Computing (STOC)*, pages 168–176, 1981.

[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.

[BPR03] S. Basu, F. Pollack, and M. F. Roy. *Algorithms in Real Algebraic Geometry*. Springer, 2003.

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

[BV73] Y. Balcer and A. F. Veinott. Computing a graph's period quadratically by node condensation. *Discrete Mathematics*, 4:295–303, 1973.

[Can88] J. Canny. Some algebraic and geometric computations in PSPACE. In *Proc. 20th ACM Symp. on Theory of Computing (STOC)*, pages 460–467, 1988.

[CG98] Z. Chi and S. Geman. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):298–305, 1998.

[DEKM99] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic models of Proteins and Nucleic Acids*. Cambridge U. Press, 1999.

[EHRS00] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proc. 12th Int. Conference on Computer Aided Verification (CAV)*, pages 232–247, 2000.

[EKL08] J. Esparza, S. Kiefer, and M. Luttenberger. Convergence thresholds of Newton's method for monotone polynomial equations. In *Proc. 25th Int. Symp. on Theoretical Aspects of Comp. Science (STACS)*, pages 289–300, 2008.

[EKM04] J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. In *Proc. 19th IEEE Symp. on Logic in Computer Science (LICS)*, pages 12–21, 2004.

[EKM05] J. Esparza, A. Kučera, and R. Mayr. Quantitative analysis of probabilistic pushdown automata: expectations and variances. In *Proc. of 20th IEEE Symp. on Logic in Computer Science (LICS)*, 2005.

[EU48] C. J. Everett and S. Ulam. Multiplicative systems, part i., ii, and iii. Technical Report 683,690,707, Los Alamos Scientific Laboratory, 1948.

[EWY08] K. Etessami, D. Wojtczak, and M. Yannakakis. Quasi-birth-death processes, tree-like qbds, probabilistic 1-counter automata, and pushdown systems. In *Proc. 5th Int. Symp. on Quantitative Evaluation of Systems (QEST)*, pages 243–253, 2008.

[EY05a] K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic state machines. In *Proc. 11th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 253–270, 2005.

[EY05b] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. In *Proc. of 22nd Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 340–352, 2005.

[EY05c] K. Etessami and M. Yannakakis. Recursive Markov decision processes and recursive stochastic games. In *Proc. 32nd Int. Coll. on Automata, Languages, and Programming (ICALP)*, pages 891–903, 2005.

[EY06a] K. Etessami and M. Yannakakis. Efficient analysis of classes of recursive Markov decision processes and stochastic games. In *Proc. 23rd Symp. on Theoretical Aspects of Comp. Sci. (STACS)*, pages 634–645, 2006.

[EY06b] K. Etessami and M. Yannakakis. Recursive concurrent stochastic games. In *Proc. 33rd Int. Colloquium on Automata, Languages, and Programming (ICALP)*, pages 324–335, 2006.

[EY07] K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points. In *Proc. 48th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 113–123, 2007.

[FKK+00] R. Fagin, A. Karlin, J. Kleinberg, P. Raghavan, S. Rajagopalan, R. Rubinfeld, M. Sudan, and A. Tomkins. Random walks with "back buttons" (extended abstract). In *Proc. ACM Symp. on Theory of Computing (STOC)*, pages 484–493, 2000.

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

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 2nd edition, 1993.

[Gre76] U. Grenander. *Lectures on Pattern Theory, Vol. 1*. Springer-Verlag, 1976.

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

[HJ85] R. J. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge U. Press, 1985.

[HJV05] P. Haccou, P. Jagers, and V. A. Vatutin. *Branching Processes: Variation, Growth, and Extinction of Populations*. Cambridge U. Press, 2005.

[HMU00] J. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addisson-Wesley, 2000.

[Jag75] P. Jagers. *Branching Processes with Biological Applications*. Wiley, 1975.

[KA02] M. Kimmel and D. E. Axelrod. *Branching processes in biology*. Springer, 2002.

[Kar66] S. Karlin. *A First Course in Stochastic Processes*. Academic Press, 1966.

[KI03] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proc. 35th ACM Symp. on Theory of Computing (STOC)*, pages 355–364, 2003.

[KLE07] S. Kiefer, M. Luttenberger, and J. Esparza. On the convergence of newton's method for monotone systems of polynomial equations. In *Proc. 39th Symp. on Theory of Computing (STOC)*, pages 217–226, 2007.

[KS47] A. N. Kolmogorov and B. A. Sevastyanov. The calculation of final probabilities for branching random processes. *Dokaldy*, 56:783–786, 1947. (Russian).

[LR99] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA-SIAM series on statistics and applied probability, 1999.

[LT85] P. Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, 2nd edition, 1985.

[Mod71] C. J. Mode. *Multitype Branching Processes. Theory and Applications*. Modern Analytic and Computational Methods in Science and Mathematics, No. 34. American Elsevier, 1971.

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

[Neu81] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models:an algorithmic approach*. Johns Hopkins U. Press, 1981.

[NS06] M. J. Nederhof and G. Satta. Using Newton's method to compute the partition function of a PCFG, 2006. *unpublished manuscript*.

[OR70] J. M. Ortega and W.C. Rheinbolt. *Iterative solution of nonlinear equations in several variables*. Academic Press, 1970.

[Ren92] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals, parts I-III. *J. Symbolic Computation*, 13(3):255–352, 1992.

[Rep98] T. Reps. Program analysis via graph reachability. *Information and Software Technology*, 40(11-12):701–726, 1998.

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

[SBH+94] 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.

[Sch79] A. Schönhage. On the power of random access machines. In *Proc. 6th Int. Coll. Aut. Lang. Prog. (ICALP)*, pages 520–529, 1979.

[Sev51] B. A. Sevastyanov. The theory of branching processes. *Uspehi Mathemat. Nauk*, 6:47–99, 1951. (Russian).

[Ste89] I. Stewart. *Galois Theory*. Chapman & Hall, 2nd edition, 1989.

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

[WE07] D. Wojtczak and K. Etessami. Premo: an analyzer for probabilistic recursive models. In *Proc. 13th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 66–71, 2007.

[YE05] M. Yannakakis and K. Etessami. Checking LTL properties of Recursive Markov Chains. In *Proceedings 2nd Int. Symp. on Quantitative Evaluation of Systems (QEST)*, pages 155–165, 2005.