Quasi-Birth-Death Processes, Tree-Like QBDs, Probabilistic 1-Counter Automata, and Pushdown Systems

Kousha Etessami
U. of Edinburgh
kousha@inf.ed.ac.uk

Dominik Wojtczak
U. of Edinburgh
D.K.Wojtczak@sms.ed.ac.uk

Mihalis Yannakakis
Columbia U.
mihalis@cs.columbia.edu

Abstract

We begin by observing that (discrete-time) Quasi-Birth-Death Processes (QBDs) are equivalent, in a precise sense, to (discrete-time) probabilistic 1-Counter Automata (p1CAs), and both Tree-Like QBDs (TL-QBDs) and Tree-Structured QBDs (TS-QBDs) are equivalent to both probabilistic Pushdown Systems (pPDSs) and Recursive Markov Chains (RMCs).

We then proceed to exploit these connections to obtain a number of new algorithmic upper and lower bounds for central computational problems about these models. Our main result is this: for an arbitrary QBD (even a null-recurrent one), we can approximate its termination probabilities (i.e., its G matrix) to within i bits of precision (i.e., within additive error $1/2^i$), in time polynomial in both the encoding size of the QBD and in i, in the unit-cost rational arithmetic RAM model of computation. Specifically, we show that a decomposed Newton's method can be used to achieve this.

We emphasize that this bound is very different from the well-known "linear/quadratic convergence" of numerical analysis, known for QBDs and TL-QBDs, which typically gives no constructive bound in terms of the encoding size of the system being solved. In fact, we observe (based on recent results for pPDSs) that for the more general TL-QBDs this bound fails badly. Specifically, in the worst case Newton's method "converges linearly" to the termination probabilities for TL-QBDs, but requires exponentially many iterations in the encoding size of the TL-QBD to approximate these probabilities within any non-trivial constant error $c < 1$.

Our upper bound proof for QBDs combines several ingredients: a detailed analysis of the structure of 1-counter automata, an iterative application of a classic condition number bound for errors in linear systems, and a very recent constructive bound on the performance of Newton's method for monotone systems of polynomial equations.

1 Introduction

A variety of important stochastic models are finitely presentable but describe an infinite-state underlying stochastic process. Among the many examples are (multi-type) branching processes, (quasi-)birth-death processes, stochastic petri nets, and stochastic context-free grammars. Computation of basic quantities associated with such stochastic models (both transient analyses and steady-state analyses) are fundamental to many applications. Yet the complexity of computing many such quantities is not adequately understood.

This paper begins by observing that there is a close correspondence between different denumerable-state probabilistic models studied, on the one hand, in the queueing theory and structured Markov chain community, and, on the other hand, more recently, in the literature on analysis and model checking of recursive probabilistic procedural programs. Specifically, we observe that discrete-time Quasi-Birth-Death processes (QBDs) are equivalent, in a precise sense, to probabilistic 1-Counter Automata (p1CAs), which are in turn a strict subclass of probabilistic Pushdown Systems (pPDSs), namely they are pPDSs restricted to a 1-letter stack alphabet. Furthermore, we show that Tree-Structured and Tree-Like QBDs (TL-QBDs), which are extensions of QBDs, are indeed equivalent to pPDSs and Recursive Markov Chains (RMCs).

These results are not at all surprising once one gets over the differences in notation and language used by the two communities. Both types of models are infinite-state structured Markov chains that are finitely presented; in the case of QBDs and their tree extensions the notation and methodology is more algebraic, matrix-based, while in the case of pPDSs it is more automata-theoretic and combinatorial.

We exploit these equivalences to obtain several new algorithmic results about these models. A number of results follow immediately from the equivalences and existing results about the various models. For in-

stance, it follows from results on RMCs that quantitative model-checking of linear-time (ω-regular) temporal properties for QBDs and TL-QBDs can be decided in PSPACE in the size of the model ([13]). On the other hand, obtaining any non-trivial approximation of the "termination probabilities" for TL-QBDs (the analog of the G matrix of QBDs), even to within any constant additive factor $c < 1/2$, is at least as hard as long standing open problems in *exact* numerical computation, such as the *square-root sum* problem, whose complexity (in the standard Turing model of computation) is not even known to be in NP [15].

Our main result is a new upper bound on numerical approximation of central quantities associated with QBDs. Specifically, we show that, given a QBD (even a null-recurrent one), the basic G matrix of "termination probabilities" for the QBD (and various other quantities of interest that can be derived from it) can be approximated to within i bits of precision in time polynomial in *both* the encoding size of the QBD and in i, in the unit-cost rational arithmetic RAM (i.e., rational Blum-Shub-Smale) model of computation. More precisely, in the stated time complexity in the unit-cost model, one can compute a matrix $\tilde{G} \geq 0$ such that $\|G - \tilde{G}\|_\infty \leq 1/2^i$. Specifically, we show that the decomposed Newton's method (studied for RMCs and for arbitrary monotone systems of polynomial equations in [12]) can be used to achieve this bound.

We emphasize that this analysis is very different from the well-known "linear/quadratic convergence" analyses traditional to numerical analysis, which is known to hold (in null-recurrent/non-null-recurrent cases, respectively) on the equations that arise for QBDs and TL-QBDs, using Newton's method and several other methods (such as logarithmic reduction and cyclic reduction). "Linear/quadratic convergence" results only bound the number of iterations required as a function of the desired error $\epsilon > 0$ (i.e., the desired number of bits i of precision). They completely ignore how large the number of iterations may need to be as a function of the encoding size of the input QBD.

In fact, we observe using recent results for pPDSs ([18]) that this polynomial upper bound for QBDs fails badly for TL-QBDs. Specifically, there are worst-case examples of TL-QBDs which require exponentially many iterations of Newton's method, as a function of the size of the TL-QBD, in order to approximate termination probabilities (the analog of the G matrix for TL-QBDs) to within any non-trivial constant additive error, thus even to within 1 bit of precision. This is the case even though Newton's method is "linearly convergent" on these examples. Our results thus reveal a vast difference in the worst case behavior of Newton's method on QBDs and TL-QBDs, not apparent from the usual "linear/quadratic" convergence analysis.

Our proof of the new upper bound for QBDs relies on several ingredients. We first perform a detailed analysis of the structure of 1-counter automata, establishing key properties. Firstly, there is a fixed polynomial, $q(n)$, such that for any QBD whose encoding size is n,[1] the termination probabilities (i.e., entries of the G matrix), which may of course be irrational, are each either 0 or $\geq 1/2^{q(n)}$. This bound fails badly for TL-QBDs, as there are simple examples (already noted for RMCs [12]) of size $O(n)$ for which positive termination probabilities are $1/2^{2^n}$. As a second crucial property, we show that the dependencies among variables in the non-linear (matrix) equation $X = A_{-1} + A_0 X + A_1 X^2$, associated with a QBD (whose least non-negative solution is the G matrix) have a very special structure when decomposed into strongly connected components (SCCs). Roughly speaking, the SCCs can have non-linear internal structure, but distinct nonlinear SCCs can not "depend" on each other. This special structure does not hold for the equations associated with termination probabilities of TL-QBDs.

These two structural results allow us to bring in other key ingredients in order to establish the polynomial upper bound for QBDs. Specifically, we use an important constructive upper bound recently established in [9] on the performance of Newton's method for (strongly connected) monotone systems of polynomial equations, combined with our result that QBD termination probabilities can be "polynomially" bounded away from zero, in order to establish that for the non-linear SCCs in the equations for G, a polynomial number of iterations of Newton's method (as a function of the encoding size and number of bits of precision), starting from the 0 vector, suffice to obtain a desired number of bits of precision for the variables in a non-linear SCC. Finally, to approximate the entire matrix G, we deal with a possibly nested series of linear SCCs "above" nonlinear ones in the DAG of SCCs, by using an iterative application of a classic, but rather delicate, condition number bound for errors in the solution of linear systems resulting from coefficient errors.

On the other hand, as a "lower bound" for QBDs, we show that deciding whether $G_{i,j} \geq p$, for a given rational p, is at least as hard as the square-root sum problem. Thus, resolving *exact* quantitative decision problems for QBDs in polynomial time or even in NP, in the traditional Turing model, is not possible with-

[1] In other words, n is the number of bits needed to describe the QBD, by describing all the rational coefficients (given by numerator and denominator in binary) in all the $m \times m$ matrices that define the QBD.

out a major breakthrough in exact numerical analysis. By contrast, for the more general TL-QBDs, we observe that our recent result in [14] for RMCs implies that even the problem of obtaining any non-trivial *approximation* of termination probabilities for TL-QBDs is square-root-sum hard.

These results lead us to suspect that a similar difference should exist in the worst-case behavior on QBDs and TL-QBDs for other numerical solution methods such as the logarithmic or cyclic reduction type algorithms (see, e.g, [2]). We have however not analysed these other algorithms. Indeed, the equivalences we point out open the door for the extensive methods and algorithms developed in the structured Markov chain community (which after all has a much longer history) to be applied to the analysis of the more recently studied models like pPDSs and RMCs, for analysis and model checking of recursive probabilistic procedural program. In the other direction, we feel that the "automata-theoretic" viewpoint, offered by the work on RMCs and pPDSs, and related literature, can be further exploited in research on QBDs, TL-QBDs, and related models. In any case, we believe a cross-fertilization between these two communities will be a fruitful source of research in the near future. A tool called PReMo [32] which implements optimized versions of the decomposed Newton's method and other methods for the analysis of Recursive Markov Chains (and their controlled and game extensions) has been augmented with an input format for QBDs.

We have conducted a very preliminary comparison of PReMo's performance on QBDs with that of an existing tool for QBDs: SMCSolver [3]. SMCSolver's implementation of algorithms like (shifted) cyclic reduction handily beats PReMo (by an order of magnitude or more) on large "dense" QBDs where the input A_i matrices are dense. This is explained by the following facts: firstly, such dense systems are typically not decomposable; moreover, SMCSolver exploits concise matrix representations of the nonlinear equations associated with QBDs, which require $O(n^2)$ encoding size (where n is the number of control states, and assuming bounded size coefficients), whereas PReMo employs an explicit algebraic formula representation of these equations (which allows handling arbitrary monotone systems of nonlinear equations) which for dense input A_i's requires $O(n^3)$ encoding size. Algorithms (like cyclic reduction) employed in SMCSolver operate directly on these matrix equations, and thus have far lower cost per iteration. However, unlike PReMo, SMCSolver does not exploit the potential for decomposing these equations (indeed, decomposition can destroy their simple matrix equation form). Thus on very decomposable systems, PReMo can do better. In particular, we constructed a family of highly decomposable QBDs, where even the G matrix itself is sparse. On these examples, with 500 control states, SMCSolver's fastest algorithm (shifted cyclic reduction) took over 40 seconds to achieve 10^{-10} error bounds, and on 5000 control states it crashed. PReMo ran in under 1 second even for 5000 control states, to achieve 10^{-10} error. However, these are very contrived examples explicitly designed to be highly decomposable. On more dense examples SMCSolver is far superior to PReMo. An interesting line of future research would be to combine the benefits of the concise matrix representations employed in, e.g., SMCSolver, and the decomposition methods employed in PReMo. One challenge in this regard is this: Newton's method can also be carried out directly over $O(n^2)$ sized matrix equations for QBDs, with low cost per iteration ($O(n^3)$ operations), using known efficient methods for solving the concise linear matrix equations that arise in each iteration of Newton's method over QBDs (certain generalized Sylvester matrix equations, see [2]). However, while TL-QBDs and RMCs also have nonlinear equations with $O(n^2)$ matrix representations, no such efficient solution method is known for the more general linear matrix equations that arise in iterations of Newton's method on them. Finding such a method would make Newton's method more practical on large "dense" TL-QBDs, RMCs, and pPDSs. But even with such an efficient method, it remains a challenge to combine it well with decomposition, because in general decomposition destroys the matrix form of the equations.

All proofs are placed in the appendix.

Related work: Quasi-Birth-Death Processes (QBDs) and more generally M/G/1-type and G/M/1-type Markov chains have been studied for decades in queueing theory, performance evaluation, and related areas, both in discrete and continuous time, and so have numerical solution methods for them (see, e.g., the books [24, 25, 22, 2]). In particular, Latouche in [21], studied the behavior of Newton's method on these models, and showed (building on [26]) that under certain assumptions (namely when $A = \sum_i A_i$ is irreducible and parameter $\rho \neq 1$) the Newton iterates are well defined and converge monotonically and "quadratically" to the matrix G. Several other "quadratically convergent" methods have also been developed, e.g., logarithmic reduction [23], and cyclic reduction (see [2]). Remke et. al. in [28] have studied numerical algorithms for model checking of continuous-time QBDs against properties expressed in the continuous-time temporal logic CSL. Several other models, in particular, (discrete-time) stochastic Petri Nets restricted to markings where just one place can be unbounded, are

already known to be equivalent to QBDs (see, e.g, [27]).

Tree-Structured QBDs (TS-QBDs) are a generalization of QBDs, first studied in [35, 29, 34]. Tree-Like QBDs (TL-QBDs) are a restriction of TS-QBDs, studied in, e.g., [22, 4, 31]. It was already observed in [30] that TL-QBDs and TS-QBDs are equivalent, under a tight notion of equivalence which amounts to an instance of what we use to show equivalence also to pPDSs and RMCs. Bini et. al. [4] studied the performance of several numerical algorithms for TL-QBDs, including Newton's method. Building on [21], they show that under a similar set of assumptions, Newton's iterations are defined and converge monotonically and quadratically for various quantities such as the termination probabilities (the analog of the G matrix).

Pushdown automata are of course classic models that date back to the origins of automata theory (see, e.g., [16]). They have many applications, e.g., in parsing of languages. Pushdown systems (the transition graphs of pushdown automata), and equivalent models such as Recursive State Machines, have been studied extensively in the past decade for the analysis and model checking of procedural programs (see, e.g., [8, 1]). In more recent years, researchers have extended these models with probabilistic behavior, i.e., to probabilistic Pushdown Systems (pPDSs) ([10, 6, 11, 5]) and Recursive Markov Chains ([12, 13, 33]), and developed model checking algorithms for them. In particular, results in [13] yield that linear-time ω-regular quantitative model checking of RMCs and pPDSs can be decided in PSPACE (we note that this is an upper bound for an exact decision procedure, not numerical estimation). A key role was played in all these analyses by the computation of termination probabilities (the analog of the G matrix) for RMCs. A number of "lower bounds" were established in [12, 14], showing that these upper bounds could not be substantially improved without major breakthroughs on long standing open problems in exact numerical computation. In [12], a decomposed Newton's method was studied for approximation of termination probabilities, and it was shown that, after decomposition, Newton's method converges monotonically, starting from 0, for arbitrary monotone polynomial systems (MSPs) that do have a non-negative solution. Subsequently, [18, 9] studied in much greater detail the performance of (decomposed) Newton's method on such monotone systems of polynomial equations. They not only established worst-case linear convergence results (even when the Jacobian at the least fixed point (LFP) is singular), but importantly for our results in this paper, they also provided a strong constructive upper bound on the number of iterations required for Newton's method as a function of the encoding size of the polynomial system. For pPDSs and RMCs their upper bounds require exponentially many iterations in the model's size in order to converge to within a constant factor of termination probabilities, but we will exploit their results, and other things, to show polynomial bounds for QBDs.

1-Counter Automata, which amount to Pushdown Systems with only one stack symbol, are a standard automata-theoretic model, and their relationship to other infinite-state models in automata theory has been well studied (see, e.g., [20, 19]). Probabilistic 1-Counter Automata have not yet been extensively studied in the literature on model checking and verification. Recently however, Brožek and Kučera have informed us [7], that they have obtained a (as yet unpublished) polynomial-time algorithm for deciding whether termination occurs almost surely (with probability 1) starting from a given control state, and counter value 1, for a probabilistic 1-counter automaton.

The rest of this paper is organized as follows. Section 2 gives basic definitions. In Section 3 we show the equivalence between QBDs and p1CA, and between Tree-structured and Tree-like QBDs and pPDS, state some consequences, and show that the square-root sum problem reduces to the decision problem for QBDs. In Section 4 we prove important structural properties of p1CAs, and in Section 5 we use them to analyze the decomposed Newton method for QBDs and prove a polynomial bound on the number of iterations.

2 Definitions

Efficient embeddings and equivalences. We show various probabilistic models are "essentially equivalent". To make the notion of "essentially equivalent" precise, we use the following definitions.

Definition 1 *For a (countable-state, discrete-time) Markov chain $\mathcal{M}$ with states t and t', we write $t \overset{\bar{t},p}{\rightsquigarrow} t'$ to denote that there is a sequence of states $\bar{t} = t_0, \ldots, t_k$, where $t_0 = t$ and $t_k = t'$, and probabilistic transitions (t_0, p, t_1) and $(t_i, 1, t_{i+1})$ for $1 \leq i < k$. (Note that if $k = 1$, this just says that (t, p, t') is a transition of $\mathcal{M}$.)*

We shall say that one (countable state) Markov chain $\mathcal{M}$ embeds efficiently *in another Markov chain $\mathcal{M}'$, if there exist two polynomial-time computable mappings, f, g, where f is a one-to-one mapping from states of $\mathcal{M}$ to states of $\mathcal{M}'$, and g is a one-to-one mapping that maps a transition (t, p, t') of $\mathcal{M}$ to a sequence, $\bar{t} = t_0 \ldots t_k$ of states in $\mathcal{M}'$, with $t_0 = f(t)$ and $t_k = f(t')$, and such that $f(t) \overset{\bar{t},p}{\rightsquigarrow} f(t')$ holds in $\mathcal{M}'$, and furthermore such that none of the auxiliary states $t_1, \ldots, t_{k-1}$ are in the range of the mapping f.*

Intuitively, this is essentially a monomorphic embedding of one Markov chain inside another, except that a transition (t, p, t') can be "stretched" into a sequence of transitions, using intermediate auxiliary states, and with probability 1 transitions out of these auxiliary states leading to the target, $f(t')$. All models we consider, even countable-state ones, have a finite description. So, for a family $\mathcal{F}$ of finite presentations of Markov chains, each $\mathcal{A} \in \mathcal{F}$, describes a potentially infinite-state underlying chain $M(\mathcal{A})$.

Definition 2 *If $\mathcal{F}$ and $\mathcal{F}'$ are two classes of finitely-presented Markov chains, we say that $\mathcal{F}$ is* efficiently subsumed by *$\mathcal{F}'$ iff: there is a polynomial-time computable mapping $h : \mathcal{F} \mapsto \mathcal{F}'$, which maps a model $\mathcal{A} \in \mathcal{F}$ to a $h(\mathcal{A}) \in \mathcal{F}'$, and such that there exists a pair of functions $f_{\mathcal{A}}$ and $g_{\mathcal{A}}$, which can themselves be efficiently computed (as Turing machines) from $\mathcal{A}$, and such that $f_{\mathcal{A}}$ and $g_{\mathcal{A}}$ constitute an efficient embedding of $M(\mathcal{A})$ into $M(h(\mathcal{A}))$. Finally, we say two classes $\mathcal{F}$ and $\mathcal{F}'$ of finitely-presented chains are* M-equivalent *if both of them are efficiently subsumed by the other.*

It is not hard to see that if one family $\mathcal{F}$ of finitely presented Markov chains is efficiently subsumed by another family $\mathcal{F}'$, via a mapping h, then a variety of computational problems for $M(A)$, where $A \in \mathcal{F}$, efficiently reduce to basically the same analyses of $M(h(A))$ where $h(A) \in \mathcal{F}'$. These include both transient analyses (such as reachability or hitting probability) as well as limit distributions.

In all the probabilistic models we define, we assume that all probability coefficients in the models are rational (for computational purposes), and that they are encoded in the standard way, by providing numerator and denominator in binary.

Probabilistic Pushdown Systems. There are a number of equivalent variations on the definition of (probabilistic) Pushdown Systems. We use a standard definition which is convenient for analysis. 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 $\Delta \subseteq (Q_P \times \Gamma) \times [0,1] \times Q_P \times \{swap(\Gamma), swap\&push(\Gamma \times \Gamma), pop\}$. That is, a transition has the form $((s,\gamma), p_{(s,\gamma),(s',\mathcal{C})}, (s',\mathcal{C}))$, where based on the control state s and the symbol on top of the stack, γ, with probability $p_{(s,\gamma),(s',\mathcal{C})}$, the transition 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-stack symbol, γ, with symbol γ'. If $\mathcal{C} = swap\&push(\gamma', \gamma'')$, then the action both swaps γ with γ' and then pushes γ'' on top of the stack. Lastly, if $\mathcal{C} = pop$, then the action pops the top-stack-symbol γ off 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 $\perp \in \Gamma$ that marks the bottom of the stack. Accordingly, $\perp$ is never overwritten with a different stack symbol, nor popped off the stack, and is never pushed onto the stack or overwrites a different stack symbol. A stack with letter γ at the top and remaining content $\omega \in \Gamma^*$ will be written $\omega\gamma$ (note that the leftmost symbol in $\omega\gamma$ is $\perp$). A pPDS $\mathcal{P}$ defines a countable-state Markov chain $M(\mathcal{P}) = (V', \Delta')$ in an obvious way. Namely, the states of $M(\mathcal{P})$ are $V' = \{(w,s) \mid s \in Q_P,\ w \in \perp\Gamma^*\}$, and the probabilistic transitions of $M(\mathcal{P})$ are $\Delta' = \{((w,s),p,(w',s')) \mid ((s,\gamma),p,(s',\mathcal{C})) \in \Delta$ & applying action $\mathcal{C}$ to w yields $w'\}$. It was shown in [12] that pPDSs are M-equivalent to *Recursive Markov Chains* (RMCs). Since we do not explicitly use RMCs, we will not recall their formal definition.

Probabilistic 1-Counter Automata. A *probabilistic 1-counter automaton* (p1CA), A, is just a pPDS with only one stack symbol γ (other than the special bottom symbol $\perp$). In other words, it is a pPDS with $\Gamma = \{\perp, \gamma\}$. This is not the usual definition: they are typically defined as having a finite number of control states and an additional non-negative counter which can be incremented or decremented during transitions, and such that transitions can be enabled/disabled depending on whether the counter is equal to 0 or not. However, this can easily be seen to be equivalent to a pPDS with one stack symbol, γ. The stack acts as precisely a (unary) counter, and the counter is equal to 0 precisely when the top stack symbol is $\perp$.

Formally, a p1CA is usually defined in the following form, which we will find convenient. A p1CA, A, is 3-tuple $A = (S, \delta, \delta_0)$ where S is a finite set of *control states* and $\delta \subseteq S \times \mathbb{R}_{>0} \times \{-1, 0, 1\} \times S$ and $\delta_0 \subseteq S \times \mathbb{R}_{>0} \times \{0, 1\} \times S$ are *transition relations*. The transition relation δ is enabled when the counter is non-zero, and the transition relation δ_0 is enabled when it is zero. We use $p^{(c)}_{u,v}$ to denote the unique probability such that there is a transition $(u, p^{(c)}_{u,v}, c, v) \in \delta$, and likewise we use $q^{(c)}_{u,v}$ to denote the unique probability such that there is a transition $(u, q^{(c)}_{u,v}, c, v) \in \delta_0$. If such a transition exists, it is unique, and thus $p^{(c)}_{u,v} > 0$ (or $q^{(c)}_{u,v} > 0$) is uniquely determined. If such a transition doesn't exist, we may sometimes assume for convenience that $p^{(c)}_{u,v} = 0$ (or $q^{(c)}_{u,v} = 0$), even though there are no explicit 0-probability transitions provided in the input which describes A. The transition probabilities out of each control state u define a probability distribution, i.e., $\sum_{c=-1}^{1} \sum_v p^{(c)}_{u,v} = 1$, and $\sum_{c=0}^{1} \sum_v q^{(c)}_{u,v} = 1$. A

p1CA, A, generates a denumerable-state Markov chain $M(A) = (V', \Delta')$ with state set $V' = \{(s,d) \mid s \in S,\ d \in \mathbb{N}\}$, and probabilistic transition relation $\Delta' = \{((s,0),p,(s',j)) \mid (s,p,j,s') \in \delta_0\} \cup \{((s,i),p,(s',j)) \mid i > 0,\ \&\ (s,p,c,s') \in \delta,\ \&\ j = i + c\}$. Obviously, pPDSs with only one stack symbol γ (other than $\perp$) and p1CAs (with unary counter) are M-equivalent.

A *1-counter automaton* (1CA) is just a p1CA without probabilities, i.e., the transition relation is nondeterministic. To each p1CA, $A = (S, \delta, \delta_0)$, we can associate an *underlying* 1CA, $A' = (S, \delta', \delta_0')$, which ignores probabilities of transitions and treats them nondeterministically. Specifically, a transition $(u,c,v) \in \delta'$ ($\in \delta_0'$) iff $p_{u,v}^{(c)} > 0$, ($q_{u,v}^{(c)} > 0$, respectively). For a 1CA, $\mathcal{A} = (S, \delta, \delta_0)$, a *path* starting at state (s_1, n_1) is a sequence of states $(s_1,n_1),(s_2,n_2),.....,(s_r,n_r)$, such that, for all $i \in \{1,\ldots,r-1\}$, either $n_i > 0$ and $(s_i, n_{i+1} - n_i, s_{i+1}) \in \delta$ or $n_i = 0$ and $(s_i, n_{i+1} - n_i, s_{i+1}) \in \delta_0$. It is called a *non-zero path* if $n_i > 0$ for all $i \in \{1,\ldots,r-1\}$. (Note that we allow $n_r = 0$ in non-zero paths.) Such a (non-zero) path is called a *(non-zero) terminating path* if $n_r = 0$, and if so it is said to terminate in state $(s_r, 0)$. For p1CAs, A, we define paths, non-zero paths, etc., as simply the paths, non-zero paths, etc., in the underlying 1CA. Note that for a p1CA, the probability that a particular non-zero path $(s_1,n_1),(s_2,n_2),.....,(s_r,n_r)$ occurs, in a random walk starting at state (s_1,n_1) of the Markov chain $M(A)$ is precisely $\prod_{1 \le i < r} p_{s_i s_{i+1}}^{(n_{i+1}-n_i)}$.

Quasi-Birth-Death Processes (QBDs). We consider discrete-time QBDs only. Of course, many analyses for continuous-time QBDs boil down to analyses of their respective embedded discrete-time chains.

A *Quasi-Birth-Death process* (QBD) is a countable state Markov chain whose transition matrix has the following block structure:[2]

$$\begin{bmatrix} B_0 & B_1 & 0 & 0 & 0 & \ldots \\ A_{-1} & A_0 & A_1 & 0 & 0 & \ldots \\ 0 & A_{-1} & A_0 & A_1 & 0 & \ldots \\ 0 & 0 & A_{-1} & A_0 & A_1 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix}$$

where $B_0, B_1, A_{-1}, A_0, A_1 \in \mathbb{R}_{\ge 0}^{m \times m}$. Thus, the finite input which describes a QBD consists of the five $m \times m$ matrices: B_0, B_1, A_{-1}, A_0, and A_1. We can represent each state of a QBD by a pair (i,j), where $1 \le i \le m$ is the index of the state within its block and $j \in \mathbb{N}$ is the index of the block. Central to many analyses for QBDs is the computation of the associated G matrix, which we will call the termination probability matrix. This is a $m \times m$ matrix, whose (i,i') entry $G_{i,i'}$ denotes the probability that, starting in state $(i,1)$, the Markov chain will eventually visit a state in block 0, and such that the first such state it visits is $(i',0)$. As is well known (e.g., [24]), G is the least non-negative solution to the matrix equation $X = A_{-1} + A_0 X + A_1 X^2$, i.e., for any non-negative solution matrix G', we have $G \le G'$ (entry-wise inequality). Other key matrices, which are also central to computations for QBDs, can be derived from the matrix G. Specifically, the R matrix, has $R_{i,i'}$ equal to the expected number of visits to state $(i', n+1)$, starting from state (i,n), before returning to a state in a block $\le n$. The matrix U (the "taboo probability" matrix) has $U_{i,i'}$ equal to the probability that starting from state $(i,1)$ the chain does not visit a state in block 0 until it eventually revisits a state in block 1, and it does so in state $(i',1)$. The matrices U and R can be obtained from G: $U = A_0 + A_1 G$, and $R = A_{-1}(I - U)^{-1}$. (Of course, the approximate solution of G will introduce errors in the solutions for U and R.) If the QBD is positive recurrent, these matrices can be used to compute steady state probabilities for being in any given state (i,j) (see, e.g., [22]).

Tree-Like and Tree-Structured QBDs. Several slight variants of TL-QBDs (and TS-QBDs) have appeared in the literature. We used the most restrictive definition of TL-QBDs in order to have the strongest results about the equivalence of all these models. Consider the infinite rooted d-ary tree T_d, label every edge with a symbol in $\Gamma = \{1,\ldots,d\}$, and label every node with the string $w \in \Gamma^*$ corresponding to the path from the root; the root is labelled with the empty string ϵ. The states of TS-QBDs and TL-QBDs consist of pairs (w,i), where $w \in \Gamma^*$ is (the label of) a node of the tree T_d and $i \in \{1,\ldots,m\}$ acts as a "control state". The transitions of a TS-QBD are as follows. From a state (ϵ, i), $i \in \{1,\ldots,m\}$, there is a transition to state:

1. (ϵ, j) with probability $f^{i,j}$, where $j \in \{1,\ldots,m\}$.
2. (s,j) with probability $u_s^{i,j}$, where $s \in \Gamma$, and $j \in \{1,\ldots,m\}$.

From any state (wk,i), where $w \in \Gamma^*$ and $k \in \Gamma$, and $i \in \{1,\ldots,m\}$, there is a transition to state:

3. (w,j) with probability $d_k^{i,j}$.
4. (ws,j), where $s \in \Gamma$, with probability $a_{k,s}^{i,j}$.
5. (wks,j), where $s \in \Gamma$, with probability $u_s^{i,j}$.

A TS-QBD can thus be described by a finite collection of $m \times m$ matrices (specifically, d^2+2d+1 such matrices)

[2] In fact, various slightly different definitions of QBDs are given in the literature, typically differing slightly on the structure of transition probabilities in the boundary cases, i.e., for the first few blocks. These differences are immaterial and these variants can be efficiently embedded in the transition structure described here, as many authors have already observed.

with rational entries, namely the matrices D_k, $A_{k,s}$, U_s, and F, where $k, s \in \Gamma$, and where their (i,j) entry is $d_k^{i,j}$, $a_{k,s}^{i,j}$, $u_s^{i,j}$, and $f^{i,j}$, respectively.

TL-QBDs are defined by restricting TS-QBDs: TL-QBDs are TS-QBDs with the additional requirement that if $k \neq s$, then $A_{k,s} = 0$ (i.e., the zero matrix), and secondly that $A_{k,k} = A_{s,s}$ for all $k, s \in \Gamma$.

3 Equivalences and basic consequences

Proposition 1 *QBDs and p1CAs are M-equivalent.*

The proof is very simple and is in the appendix. Briefly, the block index of a state of a QBD corresponds to the counter value of a state of the p1CA, the B matrices of a QBD correspond to the probabilities q of transitions on 0 counter and the A matrices correspond to the probabilities p of transitions on nonzero counter.

Theorem 2 *pPDSs, RMCs, TL-QBDs, and TS-QBDs are all M-equivalent.*

Obviously TL-QBDs are a special case of TS-QBDs. Furthermore, TS-QBDs are themselves a special case of pPDSs (equivalently, RMCs [12]), where transitions are constrained as follows: (1) If $((s,\gamma), p_{(s,\gamma),(s',\mathcal{C})}, (s',\mathcal{C})) \in \Delta$, where $\mathcal{C} = \mathit{swap\&push}(\gamma', \gamma'')$, we must have $\gamma = \gamma'$, i.e., every "swap and push" operation must be just a "push", and (2) for all $\gamma, \gamma' \in \Gamma$, we must have $p_{(s,\gamma),(s',\mathit{swap\&push}(\gamma,\gamma''))} = p_{(s,\gamma'),(s',\mathit{swap\&push}(\gamma',\gamma''))}$, i.e., the probability of the "push" does not depend on the top stack symbol. The argument that pPDS can be efficiently embedded in TL-QBDs is a bit more involved and is given in the appendix. Briefly, we can use auxiliary control states in TL-QBDs in order to mimic, e.g., a pPDS's *swap&push* operation via a sequence of transitions in a TL-QBD.

Thus all the known results for pPDSs and RMCs apply to TL-QBDs, and vice versa. The following corollary highlights a few results for TL-QBDs (and TS-QBDs) that follow from work on pPDSs and RMCs. The *square-root sum problem* asks, given natural numbers $(d_1, \ldots, d_n) \in \mathbb{N}^n$ and $k \in \mathbb{N}$, whether $(\sum_i \sqrt{d_i}) \geq k$. This decision problem is contained in PSPACE, but its containment even in NP is a longstanding open problem first posed in the 1970s ([15]), with many applications. See ([12]) for more background.

Corollary 3
1. *([13]) The quantitative model checking problem for QBDs and TL-QBDs, against linear-time ω-regular temporal properties, is decidable in PSPACE.*
2. *([12, 14]) The square-root sum problem is polynomial time reducible to the problem of approximating the termination probability (the analog of the G matrix) for TL-QBDs, even to within just one bit of precision (or even to within any constant additive factor $c < 1/2$). Furthermore, even deciding whether a termination probability for a TL-QBD is 1 is sqrt-sum-hard.*
3. *([18]) There are TL-QBDs for which at least exponentially many iterations of the (decomposed) Newton's method ([12]), applied to the non-linear equations for termination probabilities are needed as a function of the TL-QBD's encoding size, to even converge to within just one bit of precision of a termination probability.*

The following is not a corollary of earlier results. It is proved in the appendix.

Theorem 4 *The square-root sum problem is polynomial time reducible to the following problem: given a p1CA (QBD) with control states u and v, and given a rational value p decide whether $G_{u,v} \leq p$.*

4 Structural properties of (p)1CAs

This section develops crucial structural properties of (probabilistic) 1-Counter Automata, used in section 5 to establish strong results on the performance of (decomposed) Newton's method for QBDs. Let $\mathtt{mp}(s,s')$ ($\mathtt{mp}_{\mathtt{n\text{-}z}}(s,s')$) denote the length of the shortest (non-zero, respectively) terminating path starting at state $(s,1)$ and terminating at state $(s',0)$. If there is no such (non-zero) terminating path, then by definition $\mathtt{mp}(s,s') = \infty$ ($\mathtt{mp}_{\mathtt{n\text{-}z}}(s,s') = \infty$, respectively). By convention, a path with a single state has length 0. The next lemma (proved in the appendix) shows that in 1CAs whenever a terminating path exists, a "short" (polynomial length) such path also exists.

Lemma 5 *Suppose $\mathcal{A} = (S, \delta, \delta_0)$ is a 1CA where $|S| = k$. For any pair of control states $s, s' \in S$, either $\mathtt{mp}_{n\text{-}z}(s,s') = \infty$ or else $\mathtt{mp}_{n\text{-}z}(s,s') \leq k^3$. Likewise, either $\mathtt{mp}(s,s') = \infty$, or else $\mathtt{mp}(s,s') \leq k^4$.*

Corollary 6 *Let $A = (S, \delta, \delta_0)$ be a p1CA where $|S| = k$, and let $p_{min} > 0$ be the smallest positive probability on any transition of A. For any pair of states $s, s' \in S$, either $G_{s,s'} = 0$ or $G_{s,s'} \geq p_{min}^{k^3}$.*

Indeed, $G_{s,s'} > 0$ iff there is a non-zero terminating path starting at $(s,1)$ and terminating at $(s',0)$. By Lemma 5, the length of the shortest such path is $\leq k^3$. Therefore its probability is at least $p_{min}^{k^3}$.

The termination probabilities $G_{u,v}, u, v \in S$ are the smallest nonnegative solution to the following fixed point equation system

$$x_{uv} = p_{uv}^{(-1)} + (\sum_{w \in S} p_{uw}^{(0)} x_{wv}) + \sum_{y \in S} p_{uy}^{(1)} \sum_{z \in S} x_{yz} x_{zv} \quad (1)$$

We can clean up this system of equations for G by removing the variables x_{uv} for which $G_{u,v} = 0$. This can be done in polynomial-time, even for more general fixed point equations associated with pPDSs and RMCs (see [12]). (After clean-up, the equations may no longer have the simple matrix form.) Henceforth, we consider only cleaned-up equation systems, where only non-zero variables remain.

Based on this equation system we can build a dependency graph, $D = (\tilde{X}, E)$, whose nodes are all non-zero variables $\tilde{X} = \{x_{uv} : u, v \in S \text{ and } G_{u,v} \neq 0\}$ and there is an edge $(x_{uv}, x_{st}) \in E$ iff x_{st} occurs on the rhs of the equation $x_{uv} = \alpha$ corresponding to x_{uv}. We decompose this graph into strongly connected components (SCCs) and sort them topologically. As a result we obtain a sequence of SCCs $X_1, X_2, \ldots, X_m$ such that there can exist a path in graph D from variable $x \in X_i$ to variable $x' \in X_j$ only if $i \geq j$. We will write $x_{st} \equiv x_{uv}$ iff $s = u$ and $t = v$. We say a variable x_{uv} *depends* on the value of a variable x_{st} iff either $x_{st} \equiv x_{uv}$, or there is a path from x_{uv} to x_{st} in the graph D. Of course this relation is transitive. We say that an equation $x_{uv} = \alpha$ is *non-linear in a set X' of variables* if, by removing all variables that are not in X' from monomials in α, we are left with an expression α' that is non-linear. We say that SCC X_i is nonlinear if the equation $x_{uv} = \alpha$ of some variable $x_{uv} \in X_i$ is nonlinear in X_i.

Theorem 7 *If the clean equation $x_{uv} = \alpha$, for a variable $x_{uv} \in X_i$ is non-linear in X_i, and if the clean equation for a variable $x_{st} \in X_j$ is non-linear in X_j, and there is a path from x_{uv} to x_{st} in dependency graph D, then there is a path from x_{st} to x_{uv} in D.*

The proof is long and is given in the appendix. With the dependency graph D, we associate in the standard way a directed acyclic graph (DAG), H, whose nodes are SCCs of D, and which has an edge from SCC X_i to X_j iff there is an edge in D from some variable in X_i to some variable in X_j. The following is an easy corollary.

Corollary 8 *In the DAG, H, along any directed path of SCCs there is at most one nonlinear SCC.*

5 New upper bounds on Newton's method for QBDs

We will now exploit the structural results about p1CAs established in section 4, to establish strong new upper bounds on the performance of (decomposed) Newton's method on QBDs. In our analysis in this section, we assume a unit-cost exact rational arithmetic RAM model of computation. In other words, individual arithmetic operations on rationals have unit cost, regardless of the size of the rationals.

Recall that in (multi-variate) Newton's method, we are given a suitably differentiable map $F : \mathbb{R}^n \mapsto \mathbb{R}^n$, and we wish to find a solution to the system of equations $F(\mathbf{x}) = \mathbf{0}$. 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 $F'(\mathbf{c})$ is the *Jacobian matrix* of partial derivatives, whose (i, j) entry is $\frac{\partial F_i}{\partial x_j}$ evaluated at $\mathbf{c}$.

In the setting of p1CAs, we have a system of n equations in n variables, $x_i = P_i(x)$, which we can denote by $\mathbf{x} = P(\mathbf{x})$. Thus, we wish to find a solution to $F(\mathbf{x}) \doteq P(\mathbf{x}) - \mathbf{x} = 0$. Note that these are polynomial functions, and thus certainly differentiable.

We shall solve this system of equations using the *decomposed Newton's method* of [12], which applies more generally not just to systems $\mathbf{x} = P(\mathbf{x})$ arising for p1CAs, but to any monotone system $\mathbf{x} = P(\mathbf{x})$ of polynomial equations (i.e., where the coefficients in $P(\mathbf{x})$ are non-negative) which has a non-negative solution. Specifically, for any such system $\mathbf{x} = P(\mathbf{x})$ which has been *cleaned up* (i.e., variables which are necessarily zero in any least solution have been removed, something which can be done easily in polynomial time [12]) we form the dependency graph D for the non-zero variables in the corresponding cleaned system of equations, we decompose D into SCCs, and form the DAG of SCCs, H. We then "solve" for the values of variables in each SCC of H, "bottom up" by applying Newton's method starting at the vector 0 to the equations for each SCC, beginning with bottom SCCs. Once one SCC is "solved" the values computed for the variables in that SCC are plugged into equations in higher SCCs that depend on those values.

Of course, since values may in general be irrational and are only converged to in the limit, we have to specify more carefully what we mean by "solve" an SCC. This is where we make crucial use of the special structure of SCCs in the case p1CAs and QBDs. By Corollary 8, for any non-linear SCC, X_i, it must be the case that any other SCC, X_j, for which there is a path in H from X_i to X_j, is linear, i.e., any variable $x_{uv} \in X_j$ has a corresponding clean equation $x_{uv} = \alpha$ which is linear in the variables of X_j, assuming variables in even lower SCCs have been assigned fixed values. It was shown in [12] (in the more general setting of monotone systems arising from RMCs and pPDSs) that for such linear SCCs, X_j, Newton's method converges in just one iteration, starting at the vector 0, to the exact rational least fixed point (LFP) solution we are after (i.e., to the values $G_{u,v}$ for these variables in $x_{uv} \in X_j$). Thus, in a bottom up fashion we can compute the exact solutions

$G_{u,v}$ for those variables x_{uv} which are in linear SCCs below any nonlinear SCC. After computing these values we plug them into equations for variables in higher SCCs that depend on them, and we eliminate the linear SCC which was already solved. We do this until there are no bottom linear SCCs remaining.

We next have to apply Newton's method to nonlinear SCCs, which can have irrational solutions which are only converged to in the limit. How many iterations are "enough"? For this, we will use the following recent result by Esparza et. al. (Theorem 3.2 of [9]) on the behavior of Newton's method on precisely such monotone nonlinear systems. Let $P(X)$ be a cleaned monotone system of polynomials (i.e., $P(X)$ consists of n multi-variate polynomials, P_i, $i = 1, \ldots, n$, in the variables $X = x_1, \ldots, x_n$), such that $X = P(X)$ has a non-negative solution, and since it is cleaned, only positive solutions, and therefore a least fixed point (LFP) solution, $q^* > 0$. A vector q' is said to have *i valid bits* of q^* if $|q^*_j - q'_j|/q^*_j \leq 2^{-i}$ for every $1 \leq j \leq n$.

Theorem 9 *([9]) Let $P(X)$ be a cleaned strongly connected monotone system of quadratic polynomials (i.e., $P(X)$ consists of n quadratic multi-variate polynomials in n variables). Let c_{min} be the smallest nonzero coefficient of any monomial in $P(X)$, and let μ_{min} and μ_{max} be the minimal and maximal components of the LFP vector $q^* > 0$, respectively. Let $k_f = n \cdot \log(\frac{\mu_{max}}{c_{min} \cdot \mu_{min} \cdot \min\{\mu_{min}, 1\}})$. Let $\mathbf{x}^j$ denote the vector of values obtained after j iterations of Newton's method on the system $F(X) = P(X) - X$, starting with the initial all 0 vector, $\mathbf{x}^0 = \mathbf{0}$. Then for every $i \geq 0$, $\mathbf{x}^{(\lceil k_f \rceil + i)}$ has i valid bits of q^*.*

For a given p1CA, we hereafter use m to denote the maximum number of bits required to encode the integer numerators and denominators of transition probabilities of the p1CA. Thus, in particular, the smallest non-zero transition probability is $p_{min} \geq 1/2^m$. Using Theorem 9, the following can be shown (see the appendix).

Theorem 10 *Let $P(X)$ be the cleaned strongly connected monotone system of quadratic polynomials associated with a nonlinear SCC, X_i, of the decomposed system of equations associated with a p1CA, and where the exact rational values $G_{u,v}$ associated with variables x_{uv} in already solved "lower" linear SCCs have been substituted for x_{uv} on the right hand side of equations for variables in X_i. Suppose that the p1CA has n control states, and thus $|X_i| \leq n^2$, and let $G|_{X_i}$ denote those entries $G_{u,v}$ of the matrix G, such that $x_{uv} \in X_i$. Then, starting with $\mathbf{x}^0 := \mathbf{0}$, for every $i \geq 0$, the Newton iteration $\mathbf{x}^{(4mn^5 + mn^2 + i)}$ has i valid bits of $G|_{X_i}$.*

Theorem 10 implies that we can compute i bits of the values $G_{u,v}$ for variables x_{uv} in non-linear SCCs of the system $X = P(X)$ associated with a p1CA (QBD), using only a number of iterations of Newton's method which is polynomially bounded in the size of the p1CA, and linearly bounded in i.

We now have to confront a major difficulty: there may be other, linear, SCCs, X_r, which are "above" such non-linear SCCs in H. Specifically, there may be a linear SCC X_r, from which there is a path in H to a non-linear SCC, X_i. In order to be able to (approximately!) compute $G_{u,v}$ for variables $x_{uv} \in X_r$, we have to first approximately compute the (possibly irrational) values $G_{u',v'}$, for $x_{u'v'} \in X_i$, and substitute this value in occurrences of $x_{u'v'}$ in equations for higher linear SCCs. The question arises: how many bits of precision i, do we need to compute $G_{u',v'}$ to in order to compute $G_{u,v}$ to within i bits of precision? To answer this, we employ a classic bound, based on condition numbers, on errors in the solution of a linear systems.

Theorem 11 *(see, e.g., [17], Chap 2.1.2, Thm 3.[3]) Consider a system of linear equations, $Bx = b$, where $B \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$. Suppose B is non-singular, and $b \neq 0$. Let $x^* = B^{-1}b$ be the unique solution to this linear system, and suppose $x^* \neq 0$. Let $\|\cdot\|$ denote any vector norm and associated matrix norm (when applied to vectors and matrices, respectively). Let $\mathrm{cond}(B) = \|B\| \cdot \|B^{-1}\|$ denote the condition number of B. Let $\varepsilon, \varepsilon' > 0$, be values such that $\varepsilon' < 1$, and $\varepsilon \cdot \mathrm{cond}(B) \leq \varepsilon'/4$. Let $\mathcal{E} \in \mathbb{R}^{n \times n}$ and $\zeta \in \mathbb{R}^n$, be such that $\frac{\|\mathcal{E}\|}{\|B\|} \leq \varepsilon$, $\frac{\|\zeta\|}{\|b\|} \leq \varepsilon$, and $\|\mathcal{E}\| < 1/\|B^{-1}\|$. Then the system of linear equations $(B + \mathcal{E})x = b + \zeta$ has a unique solution x^*_ε such that:*

$$\frac{\|x^*_\varepsilon - x^*\|}{\|x^*\|} \leq \varepsilon'$$

We will apply this theorem using the l_∞ vector norm and induced matrix norm (*maximum absolute row sum*): $\|x\|_\infty = \max_i |x_i|$ and $\|A\|_\infty = \max_i \sum_j |a_{ij}|$.

Suppose that the fixed point equation system for a linear SCC of a p1CA, which lives "above" some nonlinear SCCs in the DAG H, looks like this: $x = Ax + b$. We know that $A \geq 0$ is an irreducible matrix (precisely because the variables being solved for are in the same SCC), $b \geq 0$, and $b \neq 0$ since otherwise the unique solution for this system would be $q^* = 0$, and zero variables were already eliminated. We can of course rewrite this linear equation as $(I - A)x = b$. It follows from a more general result in [12] about the decomposed systems of equations arising for RMCs (pPDSs) (specifically, see Lemma 17 and Theorem 14 of [12]),

[3] Our statement is weaker, but derivable from that theorem.

that $\rho(A) < 1$, where $\rho(A)$ denotes the spectral radius of A, and that therefore $(I - A)$ is non-singular, and furthermore $(I - A)^{-1} = (\sum_{i=0}^{\infty} A^i)$. Thus the LFP of this equation system is $q^* = (I-A)^{-1}b = (\sum_{k=0}^{\infty} A^k)b$. To prove bounds on errors in "higher" linear SCCs, when values in nonlinear SCCs are approximated, we will need the following two lemmas (proof in the appendix):

Lemma 12 *Let $A \in \mathbb{R}_{\geq 0}^{n \times n}$ and $b \in \mathbb{R}_{\geq 0}^{n}$, such that: $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$, and $(\sum_{k=0}^{\infty} A^k)b \leq \mathbf{1}$, and A is an irreducible non-negative matrix whose smallest non-zero entry is $c > 0$, and $b \neq 0$ and $p > 0$ is the largest entry of b. Then: $\| \sum_{k=0}^{\infty} A^k \|_\infty \leq \frac{n}{pc^n}$.*

Lemma 13 *Let X_r be a linear SCC of the cleaned equation system for a p1CA, whose corresponding linear equation system is $x = Ax+b$, after variables x_{uv} in lower SCCs have been substituted by their exact (possibly irrational) values $G_{u,v}$. Let p_{min} denote the smallest positive probability on any transition of the p1CA, and let n be its number of control states. Then the following bounds hold:*

1. $\frac{1}{2^{2mn^3+m}} \leq p_{min}^{2n^3+1} \leq \|(I - A)\|_\infty \leq n + 1$
2. $\|(I - A)^{-1}\|_\infty \leq \frac{n^2}{p_{min}^{5n^5}} \leq n^2 \cdot 2^{5mn^5}$
3. $\text{cond}(I - A) \leq \frac{2n^3}{p_{min}^{5n^5}} \leq 2n^3 \cdot 2^{5mn^5}$
4. $\|b\|_\infty \geq p_{min}^{2n^3+1} \geq \frac{1}{2^{2mn^3+m}}$

For a "higher" linear SCC, X_r, i.e., one which can reach some non-linear SCC in H, let us define its *height*, $h_r < \infty$, to be the maximum finite distance in H between X_r and some lower non-linear SCC that it can reach. Let $h_{max} = \max_r h_r$, where the maximum is taken over all linear SCCs that can reach a non-linear SCC. Note that, as a very loose upper bound, certainly $h_{max} \leq n^2$, where $n = |S|$ is the number of control states of the p1CA, because there are at most n^2 variables in the entire system. Now consider the decomposed Newton's method applied to the fixed point equations for a p1CA, with the following specification for the number of iterations to be applied to each SCC:

1. Use, one iteration of Newton's method (starting at vector $x^0 = 0$), or any linear system solving method, to solve a remaining bottom linear SCC exactly. Remove the linear SCC, and plug the corresponding values of variables into equations for higher SCCs. Do this until only non-linear bottom SCCs remain, or all SCCs are solved.
2. For each remaining non-linear SCC, apply Newton's method (starting with vector $x^0 = 0$) to the non-linear equations for these SCCs, using the following number of iterations:
$$4mn^5 + mn^2 + h_{max}(9mn^5 + 4) + i$$
Afterwards, plug the resulting (approximate) values for variables in each such non-linear SCC into the equations for higher (linear) SCCs.
3. For each remaining linear SCC, use one iteration of Newton's method (or any other linear system solution method) to solve for the exact (unique) solution of the corresponding linear system (note that the coefficients of these equations will have errors because of the approximations below, but we still seek their exact solution), then remove the linear SCC, and plug these values into higher (linear) SCCs that remain, until no SCCs remain.

Theorem 14 *Given a p1CA (or, equivalently, a QBD), the above algorithm, based on (a decomposed) Newton's method, approximates every entry of the matrix G of termination probabilities for the p1CA (QBD) to within i bits of precision (i.e., to within additive error $1/2^i$). In the unit-cost arithmetic RAM model of computation (i.e., rational Blum-Shub-Smale model), the algorithm has a running time which is polynomial in both the encoding size of the p1CA (QBD) and in i.*

The proof is in the appendix. We briefly sketch the argument. It proceeds by induction on the height, h, of a given "higher" linear SCC, X_r, above non-linear SCCs, to show that for every variable $x_{uv} \in X_r$ we compute $G_{u,v}$ to within $W_h = (h_{max} - h)(9mn^5 + 4) + i$ bits of precision. The base case $h = 0$ follows from the fact that, applying Theorem 10, all non-linear SCCs are approximated to within $W_0 = h_{max}(9mn^5 + 4) + i$ bits of precision, and all "lower" linear SCCs are computed exactly. The inductive step shows, by using Theorem 11 and Lemma 13, that at each height $h \geq 1$, solution of the linear SCCs at that height can "lose for us" at most another $9mn^5+4$ bits of precision. We emphasize that these (impractical) upper bounds for the number of iterations are very coarse, and are only intended to facilitate our proof that polynomially many iterations of Newton's method suffice. A more detailed analysis would likely yield polynomial bounds with much smaller exponents as the required number of iterations.

An important remaining open question is this: can polynomial upper bounds for approximating the G matrix for QBDs be established in the standard Turing model of computation, rather than in the unit-cost rational arithmetic RAM model as we have done? A possible approach for doing this is via a detailed analysis of the effect of round-off errors on iterations of Newton's method over the nonlinear equations for QBDs.

References

[1] 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.

[2] D. Bini, G. Latouche, and B. Meini. *Numerical methods for Structured Makov Chains.* Oxford Press, 2005.

[3] D. Bini, B. Meini, S. Steffe, and B. Van Houdt. Structured markov chains solver: algorithms/software tools. In *Proc. of SMCtools'06*, 2006.

[4] D. A. Bini, G. Latouche, and B. Meini. Solving nonlinear matrix equations arising in tree-like stochastic processes. *Linear Algebra Appl.*, 366:39–64, 2003.

[5] T. Brázdil, J. Esparza, and A. Kucera. Analysis and prediction of the long-run behavior of probabilistic sequential programs with recursion. In *Proc. of FOCS*, pages 521–530, 2005.

[6] T. Brázdil, A. Kučera, and O. Stražovský. Decidability of temporal properties of probabilistic pushdown automata. In *Proc. of STACS'05*, 2005.

[7] V. Brožek & A. Kučera. personal communication,2008.

[8] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proc. 12th CAV*, pages 232–247, 2000.

[9] J. Esparza, S. Kiefer, and M. Luttenberger. Convergence thresholds of Newton's method for monotone polynomial equations. In *Proc. 25th STACS*, pages 289–300, 2008.

[10] J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. In *Proc. of 19th LICS*, pages 12–21, 2004.

[11] J. Esparza, A. Kučera, and R. Mayr. Quantitative analysis of probabilistic pushdown automata: expectations and variances. In *Proc. of 20th LICS*, 2005.

[12] K. Etessami and M. Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of non-linear equations. In *Proc. of 22nd STACS'05*. Springer, 2005. (See full version at: `http://homepages.inf.ed.ac.uk/kousha/`).

[13] K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic state machines. In *Proc. 11th TACAS*, 2005.

[14] K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points. In *Proc. of 48th IEEE FOCS*, 2007.

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

[16] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Formal Languages and Computation.* Addison-Wesley, 1979.

[17] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods.* J. Wiley & Sons, 1966.

[18] S. Kiefer, M. Luttenberger, and J. Esparza. On the convergence of newton's method for monotone systems of polynomial equations. In *Proc. 39th ACM STOC*, pages 217–226, 2007.

[19] A. Kucera. The complexity of bisimilarity checking for one-counter processes. *TCS*, 304:157–183, 2003.

[20] A. Kucera and P. Jancar. Equivalence-checking with infinite-state systems: Techniques and results. In *SOFSEM*, pages 41–73, 2002.

[21] G. Latouche. Newton's iteration for non-linear equations in Markov chains. *IMA J. Numer. Anal.*, 14(4):583–598, 1994.

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

[23] G. Latouche and V. Ramaswami. A logarithmic reduction algorithm for quasi-birth-death processes. *J. of Applied Prob.*, 30(3):650–674, 2003.

[24] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models:an algorithmic approach.* Dover, 1981.

[25] M. F. Neuts. *Stuctured Stochastic Matrices of M/G/1 Type and their applications.* Marcel Dekker, 1989.

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

[27] A. Ost. *Performance of Communication Systems. A Model-Based Approach with Matrix-Geometric Methods.* PhD thesis, RWTH Aachen, 2001.

[28] A. Remke, B. R. Haverkort, and L. Cloth. CSL model checking algorithms for QBDs. *Theor. Comput. Sci.*, 382(1):24–41, 2007.

[29] T. Takine, B. Sengupta, and R. W. Yeung. A generalization of the matrix $M/G/1$ paradigm for Markov chains with a tree structure. *Comm. Statist. Stochastic Models*, 11(3):411–421, 1995.

[30] B. van Houdt and C. Blondia. Tree structured QBD markov chains and tree-like QBD processes. *Stochastic Models*, 19(4):467–482, 2003.

[31] J. Van Velthoven, B. Van Houdt, and C. Blondia. Transient analysis of tree-like processes and its application to random access systems. In *SIGMETRICS/Performance*, pages 181–190, 2006.

[32] D. Wojtczak and K. Etessami. Premo: an analyzer for probabilistic recursive models. In *Proc. of TACAS*, 2007. Tool web page: `http://groups.inf.ed.ac.uk/premo/`.

[33] M. Yannakakis and K. Etessami. Checking LTL properties of Recursive Markov Chains. In *Proc. 2nd QEST*, 2005.

[34] R. W. Yeung and A. S Alfa. The quasi-birth-death type Markov chain with a tree structure. *Comm. Statist. Stochastic Models*, 15(4):639–659, 1999.

[35] R. W. Yeung and B. Sengupta. Matrix product-form solutions for Markov chains with a tree structure. *Adv. in Appl. Probab.*, 26(4):965–987, 1994.

A Appendix

A.1 Proof of Proposition 1

Proposition 1 *QBDs and p1CAs are M-equivalent.*
Proof.
1. Given a QBD, $\mathcal{A}$, with underlying $k \times k$ matrices $B_0, B_1, A_{-1}, A_0, A_1$, the states of the corresponding PDS, $h(\mathcal{A})$, shall have the structure $\mathcal{P} = (Q_P, \Gamma, \Delta)$, where $\Gamma = \{\bot, \gamma\}$, and $Q_P = \{1, \ldots k\}$. The transition relation Δ is defined to contain precisely the following transitions: for $1 \leq i, j \leq k$:

- $((i, \bot), (B_0)_{i,j}, (j, \mathit{swap}(\bot))) \in \Delta$.
- $((i, \bot), (B_1)_{i,j}, (j, \mathit{swap\&push}(\bot, \gamma))) \in \Delta$.
- $((i, \gamma), (A_{-1})_{i,j}, (j, \mathit{pop})) \in \Delta$.
- $((i, \gamma), (A_0)_{i,j}, (j, \mathit{swap}(\gamma))) \in \Delta$.
- $((i, \gamma), (A_1)_{i,j}, (j, \mathit{swap\&push}(\gamma, \gamma))) \in \Delta$.

Clearly, $\mathcal{P}$ defines a pPDS with the property that it has one stack symbol γ other than $\bot$, and the stack is always of the form $\bot\gamma^r$, for some $r \geq 0$. It is not hard to see that this translation yields an efficient embedding.
2. Any pPDS with only one stack symbol can be viewed as a QBD. Indeed, this is fairly easy to see. Given such a pPDS, the *swap* transitions out of pairs of the form $(q, \bot)$, where, recall, we must swap $(q, \bot)$ with $(q', \bot)$ in order to maintain $\bot$ at the bottom of the stack, can be viewed as giving the matrix B_0, and any $\mathit{swap\&push}(\bot, \gamma)$ transitions out of $(q, \bot)$ can be viewed as giving the matrix B_1. Furthermore, for the transitions out of pairs of the form (q, γ), we can view the *pop*, $\mathit{swap}(\gamma)$ and $\mathit{swap\&push}(\gamma, \gamma)$ transitions as giving the matrices A_{-1}, A_0, and A_1, respectively. ∎

A.2 Proof of Theorem 2

Theorem 2 *pPDSs, RMCs, TL-QBDs, and TS-QBDs are all M-equivalent.*
Proof. It is easy to see from the definitions that pPDSs are the most general model and TL-QBDs the least general, To prove all equivalences, we show that the *swap&push* operation of a pPDS can be encoded using a sequence of 3 transitions of a TL-QBD, using new auxiliary states. Note that the *pop* operation of a pPDS effectively already exists in TL-QBDs, and the *swap* operation of a pPDS can then also be encoded once we have *swap&push*: we can simply add a new symbol, ζ, to Γ and instead of a transition from state $(w\gamma, i)$ to state $(w\gamma', j)$ with probability p, which do a transition from state $(w\gamma, i)$ to $(w\gamma'\zeta, j)$ with probability p, and furthermore for any state $(w'\zeta, j)$ we have a probability 1 transition to to (w', j). Note that the two transitions together take us from state $(w\gamma, i)$ to state $(w\gamma', j)$ with probability p. Note that we do have available, in a TL-QBD, the ability to do a "pop" with probability 1, as in the second transition described here, which can depend on the top stack symbol, in this case ζ, and we need not change the control state.

Now we describe how to implement *swap&push*. If the original control states of the pPDS are $\{1, \ldots, n\}$, then the new control states of the TL-QBD will be of the form $\{1, \ldots, n\} \times \Gamma^{\leq 2} \times \{1, 2, 3\}$. The swap operations of the pPDS shall be mimiced by swap operations (as described above) on control states of the form $(q, \emptyset, 1)$. The only place control states of the form $(q, \gamma, 2)$ and $(q, \gamma, 3)$ shall be used is as follows: a transition of the form: $((q, \gamma), p_{(q,\gamma),(q',\mathcal{C})}, (q', \mathcal{C})$ of the pPDS, where $\mathcal{C} = \mathit{swap\&push}(\gamma', \gamma'')$, shall be mimiced by using the following three transitions of the TL-QBD:

Starting at state $(w\gamma, (q, \emptyset, 1))$ of the TL-QBD, there is a transition with probability $p_{(q,\gamma),(q',\mathcal{C})}$ $(= d_\gamma^{(q,\emptyset,1),(q',\gamma'\gamma'',2)})$ to state $(w, (q', \gamma'\gamma'', 2))$, followed by a probability 1 $(= u_{\gamma'}^{(q',\gamma'\gamma'',2),(q',\gamma'',3)})$ transition from state $(w, (q', \gamma'\gamma'', 2))$ to state $(w\gamma', (q', \gamma'', 3))$, and then finally a probability 1 $(= u_{\gamma''}^{((q',\gamma'',3),(q',\emptyset,1)})$ transition from $(w\gamma', (q', \gamma'', 3))$ to $(w\gamma'\gamma'', (q', \emptyset, 1))$.

The given transformation constitutes an efficient embedding of the Markov chain $M(\mathcal{P})$, for the given pPDS, $\mathcal{P}$, into the Markov chain $M(\mathcal{A}_\mathcal{P})$ for a corresponding TL-QBD, $\mathcal{A}_\mathcal{P}$. In particular, the number of control states of $\mathcal{A}_\mathcal{P}$ is at most $3|Q_\mathcal{P}| \cdot |\Gamma_\mathcal{P}|^2$, and the size of the stack alphabet for $\mathcal{A}_\mathcal{P}$ is the same as that of $\mathcal{P}$. This mapping thus defines an efficient embedding, and establishes the equivalence. ∎

A.3 Proof of Theorem 4

Theorem 4 *The square-root sum problem is polynomial time reducible to the following problem: given a p1CA (QBD) with control states u and v, and given a rational value p decide whether $G_{u,v} \leq p$.*
Proof. This proof is very similar to the proof in [12] that 1-exit RMCs are sqrt-sum-hard.

Given numbers $(d_1, \ldots, d_n)$ and k, we will construct a p1CA as follows. The p1CA has control state u and n other control states, t_i, corresponding to the given numbers, d_i, $i = 1, \ldots, n$. It also one other control state, v. Let $m = \max_i d_i$. Let $c_i = (1/2)(1 - (d_i/m^2))$, for $i = 1, \ldots, n$. The transitions of the p1CA are as follows, for $i = 1, \ldots, n$: $(u, 1/n, 0, t_i) \in \delta$

$(t_i, 1/2, +1, t_i) \in \delta$ and $(t_i, c_i, -1, t_i) \in \delta$ and $(t_i, 1/2 - c_i, 0, v) \in \delta$, also $(v, 1, -1, v) \in \delta$.

We claim that $G_{u,v} = (1/(nm)) \cdot \sum_{i=1}^{n} \sqrt{d_i}$, and thus that $G_{u,v} \leq (k/(nm))$ if and only if $\sum_{i=1}^{n} \sqrt{d_i} \leq k$. To see the claim, note that for each i, we have G_{t_i,t_i} is the least non-negative solution to equation $x = (1/2)x^2 + c_i$, and thus that $G_{t_i,t_i} = (1 - \sqrt{(1-2c_i)}) = (1 - \sqrt{d_i}/m)$. Next note that the probability of terminating (in any state) starting from each t_i is 1, because it satisfies the equation $x = (1/2)x^2 + (1/2)$. Thus, $G_{t_i,t_i} + G_{t_i,v} = 1$ and therefore $G_{t_i,v} = \sqrt{d_i}/m$. Thus, $G_{u,v} = \sum_i (1/n)\sqrt{d_i}/m = 1/(nm)\sum_i \sqrt{d_i}$. ∎

A.4 Proof of Lemma 5

Lemma 5 *Suppose $\mathcal{A} = (S, \delta, \delta_0)$ is a 1CA where $|S| = k$. For any pair of control states $s, s' \in S$, either $\mathtt{mp}_{n\text{-}z}(s, s') = \infty$ or else $\mathtt{mp}_{n\text{-}z}(s, s') \leq k^3$. Likewise, either $\mathtt{mp}(s, s') = \infty$, or else $\mathtt{mp}(s, s') \leq k^4$.*

Proof. We first prove the k^3 upper bound for the length of non-zero terminating paths, and we then show why a k^4 upper bound follows for the length of arbitrary terminating paths. Let $(s_1, n_1), (s_2, n_2), (s_3, n_3), ..., (s_r, n_r)$ be the shortest non-zero terminating path starting from $(s, 1)$ and terminating in $(s', 0)$. (In particular, $(s_1, n_1) = (s, 1)$ and $(s_r, n_r) = (s', 0)$.)

Let $c_{max} = \max_{i=1}^{r} n_r$ be the maximum value of the counter along this path. There exists some state (s_j, c_{max}) on this path that achieves the highest counter value. (c_{max} may occur more than once, but let's just pick one, say the earliest occurrence.)

For every counter value $c = 1, \ldots, c_{max}$, we define the pairs (s_{i_c}, c) and $(s_{i'_c}, c)$ as follows: i_c is the largest index $i \leq j$ in the path such that the i'th state is (s_i, c), and such that for all $i \leq j' \leq j$, the j''th state on the path is $(s_{j'}, c')$ where $c' \geq c$. (In other words, in the segment from (s_i, c) to (s_j, c_{max}) the count doesn't go below c.) Likewise i'_c is the smallest index $i \geq j$ such (s_i, c) is on the path and such that on the subpath from (s_j, c_{max}) to (s_i, c) the counter doesn't go below c. Note that $i_{c_{max}} = i'_{c_{max}} = j$.

Clearly such pairs of indices i_c and i'_c are uniquely defined for each $c = 1, \ldots, c_{max}$, and we have $i_1 < i_2 < .... < i_{c_{max}} = i'_{c_{max}} < ... < i'_2 < i'_1$.

Now the key observation: if $c_{max} > k^2$ then by the pigeon-hole principle there must exist a pair of control states s^a and s^b such that for two distinct values $1 \leq c' < c'' \leq c_{max}$ of the counter, we have $s^a = s_{i_{c'}} = s_{i_{c''}}$ and $s^b = s_{i'_{c'}} = s_{i'_{c''}}$. Therefore, since we must have $i_{c'} < i_{c''} \leq i'_{c''} < i'_{c'}$, we can remove the following two, positive length, segments from the above shortest path and still get a valid non-zero terminating path from $(s_1, 1)$ to $(s_r, 0)$, which would be a contradiction. Namely, we can remove segments: $(s_{i_{c'}}, n_{i_{c'}}) \ldots (s_{i_{c''}-1}, n_{i_{c''}-1})$ and $(s_{i'_{c''}+1}, n_{i'_{c''}+1}) \ldots (s_{i'_{c'}}, n_{i'_{c'}})$. The resulting path is guaranteed by its construction to be a shorter non-zero terminating path, starting at $(s, 1)$ and terminating at $(s'.0)$, contradicting the fact that the original path was the shortest such path. Therefore, by contradiction, it must be the case that $c_{max} \leq k^2$.

Therefore, the path $(s_1, 1)....(s_{r-1}, n_{r-1})$ can contain at most $k(k^2) = k^3$ distinct states (not counting repetitions). However, note that in fact no state can repeat along this shortest non-zero terminating because otherwise it wouldn't be a shortest non-zero terminating path. Therefore the length of the shortest non-zero terminating path from $(s, 1)$ to $(s', 0)$ is $\mathtt{mp}_{\text{n-z}}(s, s') \leq k^3$.

Next we show why it follows that unless $\mathtt{mp}(s, s') = \infty$, then $\mathtt{mp}(s, s') \leq k^4$. Consider a shortest terminating path $\pi = (s, 1) \ldots (s', 0)$, which may include intermediate states with 0 counter values. Note that such a shortest path can only hit the counter value 0 at most k times, because otherwise a 0-counter state would be repeated, and this would then not constitute a shortest path. By the established k^3 upper bound on the length of shortest non-zero terminating paths, we know that the subpath between every pair of 0-counter states in the shortest path π can have at most length k^3. Since there are at most k 0-counter states along the path, the total length of the path is $|\pi| \leq k^4$. ∎

A.5 Background for the proof of Theorem 7

We introduce some additional notation. For a 1CA, $A = (S, \delta, \delta_0)$, we write $u \xrightarrow{+} v$ iff $(u, 1, v) \in \delta$; we write $u \rightarrow v$ iff $(u, 0, v) \in \delta$, and $u \xrightarrow{-} v$ iff $(u, -1, v) \in \delta$. We use the same notation for p1CAs, to denote positive probability transitions, i.e., such transitions existing in the underlying 1CA. For a (p)1CA, and for $k < 0$, we write $s \xrightarrow{k} t$ iff there exists a non-zero terminating path starting at $(s, |k|)$ and terminating at $(t, 0)$. For $k \geq 0$ we write $s \xrightarrow{k} t$ iff there exists a non-zero path starting at $(s, 1)$ and ending at $(t, k+1)$. Note that all states along this path have counter value ≥ 1. In the special case $k = 0$ we have $u \xrightarrow{0} u$ for all $u \in S$, since we allow paths to have length 0. Also note that $s \xrightarrow{+} t$ implies $s \xrightarrow{1} t$, and $s \rightarrow t$ implies $s \xrightarrow{0} t$ and finally $s \xrightarrow{-} t$ implies $s \xrightarrow{-1} t$.

Suppose that for some k, $s \xrightarrow{k} t$ holds, and that $(s, n_1) \dots (t, n_l)$ is a non-zero path that witnesses this. Then note that, for any $d > 0$, $(s, n_1 + d) \dots (t, n_l + d)$ is also a non-zero path in the same (p)1CA. We will exploit this fact repeatedly.

Proposition 15 *If $u \xrightarrow{k_1} v \xrightarrow{k_2} w$ for some $u, v, w \in S$, and either $k_1 \geq 0$ or $k_1, k_2 \leq 0$, then $u \xrightarrow{k_1+k_2} w$.*

Proof. We join the two paths: from u to v satisfying $\xrightarrow{k_1}$ and from v to w satisfying $\xrightarrow{k_2}$. The resulting path will fulfil the $\xrightarrow{k_1+k_2}$ requirements. For instance if $k_1 \geq 0$ and $k_1 + k_2 \geq 0$ then the first part of the joined path from u to v starting at $(u, 1)$ will reach $(v, k_1 + 1)$ without encountering a 0-counter state, since it fulfils $\xrightarrow{k_1}$. The second part from v to w will have the counter shifted up by k_1, thus it starts at (v, k_1+1) and finishes at (w, k_1+k_2+1), but does not hit counter 0 in between, since it fulfils $\xrightarrow{k_2}$. ∎

Note that it might be the case that $u \xrightarrow{k_1} v \xrightarrow{k_2} w$, but $u \xrightarrow{k_1+k_2} w$ does not hold. This can only happen if $k_1 < 0$ and $k_2 \geq 0$. For instance, if $\delta = \{(u, 1.0, -1, v), (v, 1.0, 1, w)\}$, we have $u \xrightarrow{-1} v \xrightarrow{1} w$, but not $u \xrightarrow{0} w$.

Proposition 16 *If $u \xrightarrow{k} v$ for some $u, v \in S$, then:*

- *if $k < -1$, $u \xrightarrow{-1} w \xrightarrow{k+1} v$, for some $w \in S$*
- *if $k > 1$, $u \xrightarrow{k-1} w \xrightarrow{1} v$, for some $w \in S$,*
- *if $k = 1$, $u \xrightarrow{0} w \xrightarrow{+} z \xrightarrow{0} v$, for some $w, z \in S$,*

(in the last case z might be equal to v and u might be equal to w).

Proof. For $k \leq -1$ pick as w the first control state on the $u \xrightarrow{k} v$ path from $(u, |k|)$ to $(v, 0)$ that has counter value $|k| - 1$. For $k \geq 1$ pick as w the last state on the $u \xrightarrow{k} v$ path from $(u, 1)$ to $(v, k + 1)$ that has counter value k. For $k = 1$, the transition after such a state $(w, 1)$ has to increase the counter since otherwise it would not be the last state on the non-zero path with counter value 1. So let the next state be $(z, 2)$. From there the non-zero path must reach the end state $(s, 2)$ without encountering a state with counter value 1. ∎

Remark 1 *After cleanup, if a variable x_{st} is on the rhs of a clean equation $x_{uv} = \alpha$, there are 3 (not mutually exclusive) possibilities for how x_{st} occurs in α:*

1. *as $p^{(0)}_{us} x_{st}$, so $u \to s \xrightarrow{-1} t = v$*
2. *as $p^{(1)}_{us} x_{st} x_{tv}$, so $u \xrightarrow{+} s \xrightarrow{-1} t \xrightarrow{-1} v$*
3. *as $p^{(1)}_{uw} x_{ws} x_{st}$, so $u \xrightarrow{+} w \xrightarrow{-1} s \xrightarrow{-1} t = v$*

Note that in cases (1.) and (3.) we have $u \xrightarrow{0} s \xrightarrow{-1} t = v$ and in case (2.) we have $u \xrightarrow{1} s \xrightarrow{-1} t \xrightarrow{-1} v$.

A.6 Proof of Theorem 7

Theorem 7 *If the clean equation $x_{uv} = \alpha$, for a variable $x_{uv} \in X_i$ is non-linear in the variables belonging to X_i, and if the clean equation for a variable $x_{st} \in X_j$ is non-linear in the variables belonging to X_j, and there is a path from x_{uv} to x_{st} in dependency graph D, then there is a path from x_{st} to x_{uv} in D.*

Proof. We will first prove a few lemmas. For control states $u, v \in S$, let δ_{uv} denote the usual Kronecker δ: $\delta_{uv} = 1$ if $u = v$ and $\delta_{uv} = 0$ if $u \neq v$.

Lemma 17 *In dependency graph D, if the shortest path from x_{uv} to x_{st} has a length $k < \infty$ then for some k', $1 - \delta_{vt} \leq k' \leq k$, we have $u \xrightarrow{k'} s \xrightarrow{-1} t \xrightarrow{-k'} v$.*

Proof. Proof by induction on k. The case $k = 1$ follows from Remark 1 and the fact that if $t = v$ (in other words $\delta_{vt} = 1$) then $t \xrightarrow{0} v$ holds by default. Assume the statement is true for k and consider some shortest path of length $k + 1$ between two variables x_{uv} and x_{st}. Let us consider the variable that is just before x_{st} on this shortest path and assume it is x_{wz} for some $w, z \in S$. Obviously the shortest path in D from x_{uv} to x_{wz} has a length k. We know from the induction assumption that for some $1 - \delta_{vz} \leq k' \leq k$ we have $u \xrightarrow{k'} w \xrightarrow{-1} z \xrightarrow{-k'} v$. On the other hand we know that from x_{wz} we can reach x_{st} in one step, thus from Remark 1 we get that $w \xrightarrow{1} s \xrightarrow{-1} t \xrightarrow{-1} z$ or $w \xrightarrow{0} s \xrightarrow{-1} t = z$ (both of these form a $w \xrightarrow{-1} z$ path). Considering these two facts together we get that either $u \xrightarrow{k'} w \xrightarrow{1} s \xrightarrow{-1} t \xrightarrow{-1} z \xrightarrow{-k'} v$ or $u \xrightarrow{k'} w \xrightarrow{0} s \xrightarrow{-1} t = z \xrightarrow{-k'} v$. Now using Proposition 15 we get that either $u \xrightarrow{k'+1} s \xrightarrow{-1} t \xrightarrow{-(k'+1)} v$ or $u \xrightarrow{k'} s \xrightarrow{-1} t \xrightarrow{-k'} v$. Hence the statement for $k + 1$ is true as well. ∎

Lemma 18 *If x_{wv} is a non-zero variable and $u \xrightarrow{0} w$ then x_{uv} is also non-zero and depends on x_{wv}.*

Proof. First of all notice that if $u = w$ then the statement is trivial. Secondly the variable x_{uv} is non-zero since a path $u \xrightarrow{0} w \xrightarrow{-1} v$ forms a $u \xrightarrow{-1} v$ path.

Now if $u \neq w$ then take a path from $(u,1)$ to $(w,1)$ that fulfils $u \xrightarrow{0} w$. Take all the states along that path that have the the counter equal to 1: $(s_0,1),(s_1,1),\ldots,(s_n,1)$ where $s_0 = u$ and $s_n = w$ (we know that $n \geq 1$ since $u \neq w$). Notice that for all $i \leq n$ the variables $x_{s_i v}$ are non-zero because path $s_i \xrightarrow{-1} v$ exists (just take a subpath of the $u \xrightarrow{0} w \xrightarrow{-1} v$ path). Now consider the state $(s_{n-1},1)$. From this state the path can not take transition reducing the counter to 0 since then the path would finish before reaching $(w,1)$. If the path takes a transition that leaves the counter unchanged then the next state on this path has to be $(s_n,1)$. It is because $(s_n,1)$ was supposed to be the next state after $(s_{n-1},1)$ to have the counter equal to 1. This means that on the rhs of the equation for the variable $x_{s_{n-1}v}$ there is an expression $p^{(0)}_{s_{n-1}s_n} x_{s_n v}$ and as a result variable $x_{s_{n-1}v}$ depends on $x_{s_n v}$. Finally, if the path from $(s_{n-1},1)$ takes a transition $s_{n-1} \xrightarrow{+} z$ then on the rhs of the equation for the variable $x_{s_{n-1}v}$ there is an expression $p^{(1)}_{s_{n-1}z} x_{z s_n} x_{s_n v}$. This is because s_n is the first state after $(z,2)$ that has the value of the counter equal to 1 and so the the path $z \xrightarrow{-1} s_n$ exists. Therefore $x_{z s_n} \neq 0$ and similarly $x_{s_n v} \neq 0$ thus after the cleaning step this expression will remain on the rhs of the equation for $x_{s_{n-1}v}$. Hence again $x_{s_{n-1}v}$ depends on $x_{s_n v}$. By an easy induction we can prove that for all $0 \leq i < n$ the variable $x_{s_i v}$ depends on $x_{s_{i+1} v}$. Now finally, from the transitivity of this relation we can deduce that variable $x_{s_0 v} (\equiv x_{uv})$ depends on $x_{s_n v} (\equiv x_{wv})$. ∎

Lemma 19 *A non-zero variable x_{uv} depends on the value of a non-zero variable x_{st} iff for some $k \geq 1 - \delta_{vt}$ we have $u \xrightarrow{k} s \xrightarrow{-1} t \xrightarrow{-k} v$.*

Proof. ($\Rightarrow$) Note that if $x_{uv} \equiv x_{st}$ then $u = s$ and $v = t$, so $1 - \delta_{vt} = 0$ and $s \xrightarrow{-1} t$ (since $x_{st} > 0$) thus we have $u \xrightarrow{0} u = s \xrightarrow{-1} t \xrightarrow{0} t = v$.

If $x_{uv} \not\equiv x_{st}$ then there is a path in D from x_{uv} to x_{st} and so there is also the shortest one. Let us denote its length by k'. From Lemma 17 for some $1 - \delta_{vt} \leq k \leq k'$ we have $u \xrightarrow{k} s \xrightarrow{-1} t \xrightarrow{-k} v$.

($\Leftarrow$) Of course x_{uv} and x_{st} are both non-zero since from $u \xrightarrow{k} s \xrightarrow{-1} t \xrightarrow{-k} v$ we know that $s \xrightarrow{-1} t$ and $u \xrightarrow{-1} v$ holds.

If it happens that $k = 0$ then necessarily $v = t$. In other words we know that $u \xrightarrow{0} s \xrightarrow{-1} t = v$ which means that $u \xrightarrow{0} s$ and $x_{st} > 0$. Now from Lemma 18 we get that x_{ut} $(\equiv x_{uv})$ is non-zero and depends on x_{st}.

The rest of the proof is by induction on k. If $k = 1$ then $u \xrightarrow{1} s \xrightarrow{-1} t \xrightarrow{-1} v$. Of course we instantly have that x_{uv}, x_{st}, x_{tv} are non-zero. From Proposition 16 we know that we can decompose the $u \xrightarrow{1} s$ part into $u \xrightarrow{0} w \xrightarrow{+} z \xrightarrow{0} s$ for some $w, z \in S$ and the whole path would like this: $u \xrightarrow{0} w \xrightarrow{+} z \xrightarrow{0} s \xrightarrow{-1} t \xrightarrow{-1} v$. Furthermore, $z \xrightarrow{-1} t$, $w \xrightarrow{-1} v$ and so x_{zt} and x_{wv} are non-zero. From this we can deduce that on the rhs of the equation for x_{wv} we will have an expression $p^{(1)}_{wz} x_{zt} x_{tv}$. This means that x_{wv} depends on variable x_{zt}. In addition from the facts $u \xrightarrow{0} w$, $z \xrightarrow{0} s$ and Lemma 18 we get that x_{uv} depends on x_{wv} and x_{zt} depends on x_{st}. Finally, from the transitivity of this relation we obtain that x_{uv} depends on x_{st}.

Now assume that the statement is true for some k' and let us consider a $u \xrightarrow{k'+1} s \xrightarrow{-1} t \xrightarrow{-(k'+1)} v$ path. From Proposition 16 we know that for some $w, z \in S$ we can decompose this path into a $u \xrightarrow{k'} w \xrightarrow{1} s \xrightarrow{-1} t \xrightarrow{-1} z \xrightarrow{-k'} v$ path. It follows that $w \xrightarrow{1} s \xrightarrow{-1} t \xrightarrow{-1} z$ and $u \xrightarrow{k'} w \xrightarrow{-1} z \xrightarrow{-k'} v$. Now from the induction assumption for $k = 1$ we get that x_{wz} is non-zero and depends on x_{st} and from the induction assumption for $k = k'$ we get that x_{uv} is non-zero and depends on x_{wz}. This means that x_{uv} also depends on x_{st}. ∎

Example 1 *It might be the case that $u \xrightarrow{0} s \xrightarrow{-1} t \xrightarrow{0} v$ where $t \neq v$, but x_{uv} does not depend on x_{st} like in the following example: $\delta = \{(u, 1.0, 0, s), (s, 1.0, -1, t), (t, 1.0, 0, v)\}$.*

Lemma 20 *If the clean equation for a variable $x_{uv} \in X_i$ is non-linear in the variables belonging to X_i then for some $k_0 \geq 1, k_1 \geq 0$ and some $w \in S$ we have $u \xrightarrow{k_0} u \xrightarrow{-1} v \xrightarrow{1-k_0} w \xrightarrow{k_1} u \xrightarrow{-1} v \xrightarrow{-k_1} v$.*

Proof. Since x_{uv} is non-linear in the variables belonging to X_i then from Remark 1 we can deduce that for some $s, t \in S$ we have $x_{st}, x_{tv} \in X_i$ and the clean equation for x_{uv} has on the rhs an expression $p^{(1)}_{us} x_{st} x_{tv}$. It follows that $u \xrightarrow{+} s \xrightarrow{-1} t \xrightarrow{-1} v$. Since x_{st} is in the same SCC as x_{uv} then there has to be a path from x_{st} to x_{uv} in the graph D and using Lemma 17 we get that for some $k \geq 1 - \delta_{vt}$ we have $s \xrightarrow{k} u \xrightarrow{-1} v \xrightarrow{-k} t$. From the same argument we get that for some $k' \geq 0$ we have $t \xrightarrow{k'} u \xrightarrow{-1} v \xrightarrow{-k'} v$. Now joining these paths together we get $u \xrightarrow{+} s \xrightarrow{k} u \xrightarrow{-1} v \xrightarrow{-k} t \xrightarrow{k'} u \xrightarrow{-1} v \xrightarrow{-k'} v$. Finally, using Proposition 15 we have $u \xrightarrow{k+1} u \xrightarrow{-1} v \xrightarrow{-k} t \xrightarrow{k'} u \xrightarrow{-1} v \xrightarrow{-k'} v$. ∎

We can now finish the proof of Theorem 7. Using Lemma 20 we get that for some $k_0, l_0 \geq 1$, $k_1, l_1 \geq 0$ and $w, z \in S$ we have $u \xrightarrow{k_0} u \xrightarrow{-1} v \xrightarrow{1-k_0} w \xrightarrow{k_1} u \xrightarrow{-1} v \xrightarrow{-k_1} v$ and $s \xrightarrow{l_0} s \xrightarrow{-1} t \xrightarrow{1-l_0} z \xrightarrow{l_1} s \xrightarrow{-1} t \xrightarrow{-l_1} t$. We can simplify the later to $s \xrightarrow{l_0} s \xrightarrow{-1} t \xrightarrow{-l_0} t$ for some $l_0 \geq 1$ using Proposition 15.

Since there is a path from x_{uv} to x_{st} then from Lemma 17 we have $u \xrightarrow{k} s \xrightarrow{-1} t \xrightarrow{-k} v$ for some $k \geq 1 - \delta_{vt}$. Now we will show that $s \xrightarrow{k'} u \xrightarrow{-1} v \xrightarrow{-k'} t$ holds for some $k' \geq 1$ and using Lemma 19 we will get that the variable x_{st} depends on the variable x_{uv}. We start the $s \xrightarrow{k'} u \xrightarrow{-1} v \xrightarrow{-k'} t$ path by iterating the $s \xrightarrow{l_0} s$ path n times for sufficiently large n obtaining a $s \xrightarrow{n \cdot l_0} s$ path: $s \underbrace{\xrightarrow{l_0} s \xrightarrow{l_0} s \xrightarrow{l_0} \ldots \xrightarrow{l_0} s}_{n \text{ times}}$. We will see how big n should be later. Now from the last s we do: $s \xrightarrow{-1} t \xrightarrow{-k} v \xrightarrow{1-k_0} w \xrightarrow{k_1} u \xrightarrow{k_0} u \xrightarrow{-1} v \xrightarrow{-k_1} v \xrightarrow{-k_1} v \xrightarrow{1-k_0} w \xrightarrow{k_1} u \xrightarrow{k_0} u \xrightarrow{k} s \xrightarrow{-1} t$ and now in the end we iterate n times the $t \xrightarrow{-l_0} t$ path. Along the whole path the value of the counter is changed by: $nl_0 - 1 - k + 1 - k_0 + k_1 + k_0 - 1 - k_1 - k_1 + 1 - k_0 + k_1 + k_0 + k - 1 - nl_0 = -1$. Now if $nl_0 > k + k_0 + k_1$ (it can be done since $l_0 \geq 1$) then using Proposition 15 we can rewrite it as $s \xrightarrow{nl_0 - k + k_1} u \xrightarrow{-1} v \xrightarrow{-nl_0 + k - k_1} t$. Basically, we can get the value of the counter sufficiently high at the beginning of the path in order to prevent it from reaching zero later along the path (the $(w, nl_0 - k - k_0 - k_1)$ state being the state with the lowest counter value along the first part of the path) until it reaches the final t state. Now finally, since $nl_0 - k + k_1 \geq 1$ it follows from Lemma 19 that x_{st} depends on x_{uv}. ∎

A.7 Proof of Corollary 8

Corollary 8 *In the DAG of SCCs, H, along any directed path of SCCs there can be at most one nonlinear SCC.*

Proof. Let X_i and X_j $(i < j)$ be two SCCs on such a path. If inside these two SCCs there are variables $x \in X_i$ and $y \in X_j$ whose equations are non-linear in the variables belonging to X_i and X_j, respectively, then since there is a path from x to y in D (in other words x depends on y) we know from Theorem 7 that there is also a path from y to x. But that implies x and y are in the same SCC, a contradiction. ∎

A.8 Proof of Theorem 10

Theorem 10 *Let $P(X)$ be the cleaned strongly connected monotone system of quadratic polynomials associated with a nonlinear SCC, X_i, of the decomposed system of equations associated with a p1CA, and where the exact rational values $G_{u,v}$ associated with variables x_{uv} in already solved "lower" linear SCCs have been substituted for x_{uv} on the right hand side of equations for variables in X_i. Suppose that the p1CA has n control states, and thus $|X_i| \leq n^2$, and let $G|_{X_i}$ denote those entries $G_{u,v}$ of the matrix G, such that $x_{uv} \in X_i$. Then, starting with $\mathbf{x}^0 := \mathbf{0}$, for every $i \geq 0$, the Newton iteration $\mathbf{x}^{(4mn^5+mn^2+i)}$ has i valid bits of $G|_{X_i}$.*

Proof. For the cleaned system $X = P(X)$ associated with a p1CA, A, by Corollary 6, $p_{min}^{n^3} \leq q^* \leq 1$ (coordinate-wise inequality), where $p_{min} > 0$ is the smallest positive probability on any transition of A. Note, in particular, that $\mu_{max} \leq 1$, and $\mu_{min} \geq p_{min}^{n^3} \geq \frac{1}{2^{mn^3}}$. Furthermore, note that because the entire system of non-linear equations for a p1CA is quadratic, the smallest coefficient c_{min} of any monomial in the system $X = P(X)$ for this non-linear SCC, can only arise as the product of p_{min} times at most 2 previously computed values $G_{u',v'}$ and $G_{u'',v''}$ for variables $x_{u'v'}$ and $x_{u''v''}$ which appeared in lower (linear) SCCs. Again, by Corollary 6, we know that $G_{u',v'}, G_{u'',v''} \geq p_{min}^{n^3}$, and thus $c_{min} \geq p_{min}^{2n^3+1} \geq 1/2^{m(2n^3+1)}$. Thus, noting that the cleaned system $X = P(X)$ for a p1CA with n control states has at most n^2 variables, the expression for k_f in Theorem 9 can be seen to be $k_f \leq n^2 \cdot \log(2^{2mn^3+m} 2^{mn^3} 2^{mn^3}) = 4mn^5 + mn^2$. ∎

A.9 Proof of Lemma 12

Lemma 12 *Let $A \in \mathbb{R}_{\geq 0}^{n \times n}$ and $b \in \mathbb{R}_{\geq 0}^{n}$, such that: $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$, and $(\sum_{k=0}^{\infty} A^k) b \leq \mathbf{1}$, and A is an irreducible non-negative matrix whose smallest non-zero entry is $c > 0$, and $b \neq 0$ and $p > 0$ is the largest entry of b. Then:*

$$\| \textstyle\sum_{k=0}^{\infty} A^k \|_{\infty} \leq \frac{n}{pc^n}$$

.

Proof. Let a_{ij}^d and a_{ij}^* denote the (i, j) entry of matrix A^d and $A^* = \sum_{k=0}^{\infty} A^k$ respectively. Since A is irreducible, for every pair of indices i, j, there exists a power $1 \leq d \leq n$ such that $a_{ij}^d > 0$. Furthermore, since the smallest non-zero entry of A is c, we have $a_{ij}^d \geq c^d$.

We know that $A^* b \leq \mathbf{1}$. Wlog we can assume that the first entry of b is $b_1 = p$, by basically permuting rows/columns of A and b. Now the i-th entry of $A^* b$ is $(A^* b)_i = \sum_{j=1}^{n} a_{ij}^* b_j \leq 1$ and thus

obviously $(A^*b)_i \geq a^*_{i1}b_1 = a^*_{i1}p$. It follows that $a^*_{i1} \leq \frac{1}{p}$, for all i. At the same time, for all $d \geq 0$, $A^*A^d = (\sum_{k=0}^{\infty} A^k)A^d = \sum_{k=d}^{\infty} A^k \leq \sum_{k=0}^{\infty} A^k = A^*$. Thus $(A^*A^d)_{(i,1)} = \sum_{j=1}^{n} a^*_{ij}a^d_{j1} \leq a^*_{i1}$. Let $a'_{i1} = (\sum_{d=1}^{n} A^*A^d)_{(i,1)}$. Thus, $a'_{i1} \leq na^*_{i1} \leq n/p$. On the other hand:

$$a'_{i1} = \sum_{d=1}^{n}\sum_{j=1}^{n} a^*_{ij}a^d_{j1} = \sum_{j=1}^{n} a^*_{ij}\left(\sum_{d=1}^{n} a^d_{j1}\right) \geq c^n \sum_{j=1}^{n} a^*_{ij}$$

The last inequality holds because, for every j, for some $1 \leq d \leq n$ we have $a^d_{j1} \geq c^d \geq c^n$. Therefore for all i we have $\sum_{j=1}^{n} a^*_{ij} \leq \frac{n}{pc^n}$ and thus $\|A^*\|_\infty \leq \frac{n}{pc^n}$. ∎

A.10 Proof of Lemma 13

Lemma 13 *Let X_r be a linear SCC of the cleaned equation system of a p1CA, whose corresponding linear equation system is $x = Ax + b$, after variables x_{uv} in lower SCCs have been substituted by their exact (possibly irrational) values $G_{u,v}$. Let p_{min} denote the smallest positive probability on any transition of the p1CA, and let n be its number of control states. Then the following bounds hold:*

1. $\frac{1}{2^{2mn^3+m}} \leq p_{min}^{2n^3+1} \leq \|(I-A)\|_\infty \leq n+1$
2. $\|(I-A)^{-1}\|_\infty \leq \frac{n^2}{p_{min}^{5n^5}} \leq n^2 \cdot 2^{5mn^5}$
3. $\text{cond}(I-A) \leq \frac{2n^3}{p_{min}^{5n^5}} \leq 2n^3 \cdot 2^{5mn^5}$
4. $\|b\|_\infty \geq p_{min}^{2n^3+1} \geq \frac{1}{2^{2mn^3+m}}$

Proof. We first show that $\|A\|_\infty \leq n$, and therefore $\|I-A\|_\infty \leq n+1$ (because A is non-negative). To see this, note that because this is a linear SCC, this means that the equations (1) for every variable x_{uv} of a linear SCC, X_r, must take the form: $x_{uv} = b_{uv} + (\sum_w p^{(0)}_{uw}x_{wv}) + \sum_y p^{(1)}_{uy}\sum_z x'_{yz}x'_{zv}$, but such that for each z, either x'_{yz} has been assigned a fixed constant (≤ 1) or x'_{zv} is a fixed constant (≤ 1). This is because, one such variable in each quadratic term must belong to a lower SCC and was thus substituted by a constant. Thus, summing the coefficients for all variables on the right hand side, we see that since $\sum_{c=-1}^{1}\sum_w p^{(c)}_{uw} \leq 1$, the full sum $\sum_j a_{ij}$ of all entries in row i of A corresponding to the variable x_{uv}, can not be more than n, the number of control states.

Before showing the lower bound on $\|I-A\|_\infty$, next we show the bound $\|b\|_\infty \geq p_{min}^{2n^3+1}$. Observe that since the equation system has been cleaned, the least fixed point solution for all variables, including in linear SCCs, is non-zero, and therefore there must exist at least one equation $x_{uv} = \alpha$ in the linear SCC with a non-negative constant term in α. The only ways such a constant term can arise is as a sum of terms of the form p, or px', or $px'x''$, where p is a transition probability of the p1CA and x' and x'' are variables in lower SCCs which have been assigned fixed constants. By Corollary 6, we have that $\|b\|_\infty \geq p_{min}^{2n^3+1}$.

Next, in order to estimate $\|(I-A)^{-1}\|_\infty$ note that, using Corollary 6, all non-zero entries of A are $\geq p_{min} \cdot (p_{min})^{n^3} = (p_{min})^{n^3+1}$. This is because all coefficients are either equal to some $p^{(c)}_{uv}$ or to $p^{(c)}_{uv} \cdot x_{wz}$ where x_{wz} is a variable from a lower SCC that has been substituted by a constant. We now use Lemma 12. Note that the dimensions of our matrix A here can in fact be as large as $n^2 \times n^2$ (because n is the number of control states, and the dimensions of A are based on the number of variables in the SCC). We thus get from Lemma 12, using the bound $\|b\|_\infty \geq p_{min}^{2n^3+1}$, and the fact that all non-zero entries of A are $\geq (p_{min})^{n^3+1}$, that $\|(I-A)^{-1}\|_\infty = \|\sum_{k=0}^{\infty} A^k\|_\infty \leq \frac{n^2}{p_{min}^{n^5+n^2+2n^3+1}} \leq \frac{n^2}{p_{min}^{5n^5}}$. It follows that $\text{cond}(I-A) = \|I-A\|_\infty \cdot \|(I-A)^{-1}\|_\infty \leq \frac{2n^3}{p_{min}^{5n^5}}$.

Finally, to see that $p_{min}^{2n^3+1} \leq \|I-A\|_\infty$, we will show that for every variable x_{uv}, the diagonal entry $(I-A)_{uv,uv} \geq p_{min}^{2n^3+1}$. To see this it suffices to note that in the original cleaned equation $x_{uv} = \alpha$ for a variable $x_{uv} \in X_r$, it can not be the case that α consists of just one linear term cx_{uv}, because otherwise the LFP of $x_{uv} = cx_{uv}$ is 0, and we have already eliminated 0 variables. Hence, it must be the case that α contains either another linear term $c'x_{st}$ or a constant term c'', or both. In either case, if we plug in the actual LFP values for all other variables besides x_{uv} into α, we will have left an equation of the form $x_{uv} = cx_{uv} + c'$, where, by the arguments of the previous two paragraphs, it must be the case that $c' \geq (p_{min})^{2n^3+1}$. Thus, solving for the (unique) solution for x_{uv}, we have $x_{uv} = c'/(1-c) \leq 1$. Therefore, $c' \leq (1-c)$, and thus $(1-c) \geq (p_{min})^{2n^3+1}$. But note that $(1-c)$ is precisely the diagonal entry $(I-A)_{uv,uv}$. Therefore $p_{min}^{2n^3+1} \leq \|I-A\|_\infty$. ∎

A.11 Proof of Theorem 14

Theorem 14 *Given a p1CA (or, equivalently, a QBD), the above algorithm, based on (a decomposed) Newton's method, approximates every entry of the matrix G of termination probabilities for the p1CA (QBD) to within i bits of precision (i.e., to within additive error $1/2^i$). Moreover, in the unit-cost arithmetic RAM model of computation (i.e., rational Blum-Shub-Smale model),*

the algorithm has a running time which is polynomial in both the encoding size of the p1CA (QBD) and in i.

Proof. First, note that up until the non-linear SCCs, all values for lower linear SCCs are computed exactly. Next note that, given the number of iterations of Newton's method that are applied in step (2.) of the algorithm for non-linear SCCs, by Theorem 10, the values $G_{u,v}$ for variables x_{uv} in non-linear SCCs are computed to within $W_0 = h_{max}(9mn^5 + 4) + i$ valid bits of precision. In other words, for each such x_{uv}, a value $G'_{u,v}$ is computed such that $|G_{u,v} - G'_{u,v}|/G_{u,v} \leq \frac{1}{2^{W_0}}$. Moreover, since $0 < G_{u,v} \leq 1$, we can conclude that $|G_{u,v} - G'_{u,v}| \leq \frac{1}{2^{W_0}}$.

Thus, since $W_0 = h_{max}(9mn^5+4)+i \geq i$, for all non-linear SCCs and all linear SCCs which are below them, we certainly do compute $G'_{u,v}$ which approximates the value $G_{u,v}$ for the variables x_{uv} in these SCC, to within at least i bits of precision (i.e., such that $|G_{u,v}-G'_{u,v}| \leq 2^i$).

The rest of the proof proceeds by induction on the height, h, of a given higher linear SCC, X_r, above the non-linear SCCs, to show that for every variable $x_{uv} \in X_r$ we compute $G_{u,v}$ to within $W_h = (h_{max} - h)(9mn^5 + 4) + i$ bits of precision.

For the base case, $h = 0$, this follows from the fact that all non-linear SCCs are computed to within $W_0 = h_{max}(9mn^5 + 4) + i$ bits of precision, and all "lower" linear SCCs are computed exactly.

For the inductive case, let X_r be an "upper" linear SCC in H at height $h > 0$ above non-linear SCCs, and and suppose that the values of all SCCs below it have been computed to within at least $W_{h-1} = (h_{max} - h + 1)(9mn^5 + 4) + i$ bits of precision, and plugged into the equations for X_r. We will show that after the linear system associated with X_r has been solved exactly, the solution gives, for each $x_{uv} \in X_r$, a value $G'_{u,v}$ such that $|G_{u,v}-G'_{u,v}| \leq \frac{1}{2^{W_h}}$, i.e., such that $G'_{u,v}$ approximates $G_{u,v}$ to within i bits of precision.

To do this, we employ Theorem 11, which gives us bounds on the errors in solutions of linear systems in terms of condition numbers and other quantities associated with the linear system, and Lemma 13, which gives us bounds on these quantities for the specific linear systems that arise for one linear SCC of a p1CA.

Suppose that, if the values of lower SCCs had been computed "exactly" (even though they can be irrational), then the resulting linear system for X_r, which may have irrational coefficients, would be $(I-A)x = b$.

Note that if the values of lower SCCs are approximated to within W_{h-1} bits of precision, then the resulting system can be written as $((I - A) + \mathcal{E})x = (b + \zeta)$. We will now bound the absolute values of entries of $\mathcal{E}$ and ζ.

Note that each entry of the matrix A is the coefficient $a_{uv,st}$ of $x_{st} \in X_r$ in the linear expression α for the equation $x_{uv} = \alpha$ of some variable $x_{uv} \in X_r$. Now, the question is, how much can $a_{uv,st}$ change when the values of lower SCCs are approximated to W_{h-1} bits of precision?

The answer is that, since the original quadratic equation $x_{uv} = \alpha'$ can have another variable $x_{s't'}$ in monomial terms that contain x_{st}, and since all such monomial terms in α' have a coefficient ≤ 1, it can be the case that by under-approximating $G_{s',t'}$ to within W_{h-1} bits of precision, we have under-approximated the resulting coefficient that arises from that monomial by at most $1/2^{W_{h-1}}$. Next we note that the coefficient $a_{uv,st}$ of x_{st} may actually arise as the sum of at most n such monomial terms (actually $n+2$, but this is immaterial and in fact it can easily be shown that they can sum to at most 2). Thus it must be the case that $\mathcal{E}_{uv,st} \leq n/2^{W_{h-1}}$.

We can ask a similar question about b. Since a constant term may arise because both variables in a quadtratic monomial of α' belonged to the lower SCCs, we now have that the resulting error $1/2^{W_{h-1}}$ could have arisen for both variables that were fixed in a monomial. It is not hard to see that the resulting error for the entire monomial is at most $2/2^{W_{h-1}}$, basically because such monomials in α' have a coefficient ≤ 1, and because for values $x, x' > 0$, we have $(x-\epsilon)(x'-\epsilon) \geq xx' - 2\epsilon$. Thus $\zeta_{uv} \leq 2n/2^{W_{h-1}}$. Since the pairs uv and st were arbitrary, and $\mathcal{E}$ is at most an $n^2 \times n^2$ matrix, we have $\|\mathcal{E}\|_\infty \leq n^3/2^{W_{h-1}}$, and $\|\zeta\|_\infty \leq 2n/2^{W_{h-1}}$.

Therefore, using Lemma 13, part (1.), we can conclude that $\frac{\|\mathcal{E}\|_\infty}{\|(I-A)\|_\infty} \leq \frac{n^3 2^{2mn^3+m}}{2^{W_{h-1}}}$, and also, using Lemma 13, part (4.), we can conclude that $\frac{\|\zeta\|_\infty}{\|b\|_\infty} \leq \frac{2n2^{2mn^3+m}}{2^{W_{h-1}}}$. Next, by Lemma 13, part (2.), we have $1/\|(I-A)^{-1}\|_\infty \geq 1/(n^2 \cdot 2^{5mn^5})$, and since $\|\mathcal{E}\|_\infty \leq n^3/2^{W_{h-1}}$, it is easy to check that $\|\mathcal{E}\|_\infty \leq 1/\|(I-A)^{-1}\|_\infty$. Finally, by Lemma 13, part (3.), $\text{cond}(I-A) \leq 2n^3 \cdot 2^{5mn^5}$.

Now we use these bounds and apply Theorem 11. Let $\varepsilon = \frac{2n^3 2^{2mn^3+m}}{2^{W_{h-1}}}$, and let $\varepsilon' = 8\varepsilon n^3 \cdot 2^{5mn^5} = \frac{16n^6 2^{2mn^3+m} 2^{5mn^5}}{2^{W_{h-1}}}$. It can be checked that, by construction, the matrix equation $(I-A)x = b$ and its approximate version $(I - A + \mathcal{E})x = (b + \zeta)$, as well as $\|\mathcal{E}\|_\infty$, $\|\zeta\|_\infty$, ε, and ε', all satisfy the conditions of Theorem 11.

Recall that the unique solution x^* to the original system is $G|_{X_r}$: it consists of those values $G_{u,v}$ where $x_{uv} \in X_r$. Thus in particular $0 < \|x^*\|_\infty \leq 1$. Thus,

by the conclusion of Theorem 11, there is a unique solution vector x^*_ε to the approximate system, such that $\|x^*_\varepsilon - x^*\|_\infty \leq \epsilon' = \frac{16n^6 2^{2mn^3+m} 2^{5mn^5}}{2^{W_{h-1}}}$.

The proof of the inductive claim will now be completed by simply checking that $16n^6 2^{2mn^3+m} 2^{5mn^5} \leq 2^{2mn^3+m+5mn^5+n^5+4} \leq 2^{9mn^5+4}$, and thus since $W_h = (h_{max} - h)(9mn^5 + 4) + i$, that $\|x^*_\varepsilon - x^*\|_\infty \leq \frac{1}{2^{W_h}}$.

The fact that the algorithm has polynomial running time in the unit-cost RAM model follows immediately from the fact that there are only polynomially many iterations of Newton's method, and each iteration essentially involves solving a linear system (or matrix inversion), which can of course be done with polynomially many arithmetic operations (e.g., using Gaussian elimination). ■