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

Kousha Etessami[a], Dominik Wojtczak[a,b], Mihalis Yannakakis[c]

[a] *School of Informatics, University of Edinburgh*
[b] *CWI, Amsterdam*
[c] *Computer Science Department, Columbia University*

Abstract

We begin by observing that (discrete-time) Quasi-Birth-Death Processes (QBDs) are equivalent, in a precise sense, to 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, 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) that for the more general TL-QBDs such a polynomial upper bound on Newton's method fails badly. 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 strongly connected monotone systems of polynomial equations.

We show that the quantitative termination decision problem for QBDs (namely, "is $G_{u,v} \geq 1/2$?") is at least as hard as long standing open problems in the complexity of exact numerical computation, specifically the square-root sum problem. On the other hand, it follows from our earlier results for RMCs that any non-trivial approximation of termination probabilities for TL-QBDs is sqrt-root-sum-hard.

Key words: Quasi-Birth-Death Processes, Tree-Like QBDs, Probabilistic 1-Counter Automata, Pushdown Systems, Newton's method

☆A preliminary version of this paper ([12]) appeared in the Proceedings of the 5th International Conference on the Quantitative Evaluation of SysTems (QEST'08).

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. Likewise, QBDs are equivalent to a strict subclass of Recursive Markov Chains (RMCs), namely 1-box RMCs. 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. Indeed both QBDs and 1-counter automata have as states pairs of the form (i, j), where $i \in \{1, \ldots, m\}$ ranges over a finite number of "control states", and $j \in \mathbb{N}$ denotes the value of a non-negative counter. Probabilistic transitions can change the control state, and increment, decrement, or keep the counter unchanged. Special transitions apply when the counter value is 0 (or, in some presentations of QBDs, less than a fixed bound). Similarly, TL-QBDs, TS-QBDs, pPDSs, and RMCs, all have states of the form (i, w) where $i \in \{1, \ldots, m\}$ ranges over a finite control, and $w \in \Sigma^*$ ranges over strings (sequences) over a finite alphabet Σ, which acts as nodes of a LIFO stack (or equivalently, nodes of a $|\Sigma|$-ary tree) such that the transitions can change the control state and add/remove/swap a symbol on the head of the stack (equivalently, move to a parent/child/sibling in the tree). These models differ in, e.g., whether the top stack symbol can influence whether or not a transition is enabled, but we show that they are all nevertheless equivalent in a precise sense.

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 instance, 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 [17].

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., discrete 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 ([14]) 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 ([20]) 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, including the structure of dependencies among variables in the non-linear equation associated with a QBD whose least non-negative solution is the G matrix and 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 [14]) 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 $X = A_{-1} +$

[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.

$A_0X+A_1X^2$, 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 nonlinear internal structure, but distinct nonlinear SCCs cannot "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 by Esparza, Kiefer, and Luttenberger 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 Directed Acyclic Graph (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 without a major breakthrough in exact numerical analysis. By contrast, for the more general TL-QBDs, we observe that our recent result in [16] 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 than Newton iteration numerical solution methods such as the logarithmic or cyclic reduction type algorithms (see, e.g, [2]). We have however not analyzed 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 [36] 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 preliminary comparison of PReMo's performance on QBDs with that of an existing tool for QBDs: SMCSolver [3] (see Section 6). SMCSolver's implementations of algorithms like (shifted) cyclic reduction handily beat 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. Finally, PReMo uses an algebraic formula encoding of equations for RMCs (and QBDs) which allows handling, much more generally, arbitrary monotone systems of nonlinear equations which are not necessarily in sparse monomial form. This encoding adds some additional cost to each computation over the explicit equations arising for RMCs and QBDs (but adding a special encoding for, e.g., QBD equations is not hard to do and would address this last issue). 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. See Section 6 for discussion of some experimental results and issues raised by them.

Related work: Quasi-Birth-Death Processes (QBDs) and more generally M/G/1-type and G/M/1-type Markov chains[2], 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 [26, 27, 24, 2]). In particular, Latouche in [23], studied the behavior of Newton's method on these models, and showed (building on [28]) 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 [25], and cyclic reduction (see [2]). Remke et. al. in [30] 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, [29]).

Tree-Structured QBDs (TS-QBDs) are a generalization of QBDs, first studied in [39, 32, 38]. Tree-Like QBDs (TL-QBDs) are a restriction of TS-QBDs, studied in, e.g., [24, 4, 35]. It was already observed in [34] 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 [23], they show that under a similar set of assumptions, Newton's itera-

[2]These chains also have the underlying transition structure of a 1-counter automaton, but one which can increase, or decrease, respectively, the counter by more than 1 in a single transition. These models need not in general be finitely presented, because they do not a priori bound how many transitions (with distinct counter value changes) can exist from a given state. But of course typical instances are finitely presented.

tions 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., [18]). 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, 7, 11, 6]) and Recursive Markov Chains ([14, 13, 37]), 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 [14, 16], showing that these upper bounds could not be substantially improved without major breakthroughs on long standing open problems in exact numerical computation. In [14], 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 that do have a non-negative solution. These results built on the classic text [28]. We were at that point unaware of the earlier related work on Newton's method for (TL-)QBDs [23, 4], but these earlier works did not derive convergence results for arbitrary (TL-)QBDs, but only for those that satisfy certain conditions. In particular, like [28], they did not handle the "critical" case of monotone polynomial systems where the Jacobian can be singular at the least fixed point (least non-negative solution), whereas in [14] we showed that monotone convergence of the decomposed Newton's method holds for arbitrary monotone polynomial systems, including in the more difficult critical case. Subsequently, Esparza, Kiefer, and Luttenberger, [20, 9] studied in much greater detail the performance of (decomposed) Newton's method on such monotone systems of polynomial equations. Firstly, they established worst-case linear convergence results even in the critical case when the Jacobian at the least fixed point (LFP) is singular (in the non-critical cases quadratic convergence can be established based on results in [28]). Importantly for our results in this paper, in [9] they also established in the case of *strongly connected* system of polynomial equations a constructive upper bound on the number of iterations required for Newton's method as a function of the encoding size of the polynomial system (see Theorem 15 in this paper). We will make crucial use of this result. For general (not strongly connected) monotone polynomial equation systems no such constructive upper bound is known. In particular, for QBDs whose corresponding polynomial equation system are not strongly connected, no constructive upper bound (as a function of the encoding size of the QBD) for Newton's method follows from any results prior to this paper. Nevertheless, we are able to use the results of [9] for the strongly connected case, combined with detailed analysis of the structure of 1-counter automata, including the special structure of the equation systems for QBDs once they are decomposed into SCCs, to show that polynomially many iterations of Newton's method suffice, as a function of *both* the size of the input *and* i, in order to

converge to within i bits of the termination probabilities.

1-Counter Automata, which amount to Pushdown Systems with only one stack symbol, are also a classic automata-theoretic model (see, e.g., [33]), and their relationship to other infinite-state models in automata theory has been well studied (see, e.g., [22, 21]). Probabilistic 1-Counter Automata have not yet been extensively studied in the literature on model checking and verification.

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 quantitative termination 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. In Section 6 we briefly describe some experimental comparison between the tool PReMo, which implements the decomposed Newton's method, and SMCSolver, a state-of-the-art tool for analysis of QBDs. We conclude in Section 7 with some discussion of open problems.

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. We view a (discrete-time) Markov chain as an object $\mathcal{M} = (V, \Delta)$, where V is a set of states and $\Delta \subseteq V \times [0,1] \times V$ is a probabilistic transition relation.

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 such that the following probabilistic transitions exist in $\mathcal{M}$: (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.*[3]

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

[3] A suitable modification of this definition can be given for continuous-time chains, by assuming that the auxiliary transitions that are mapped to can be given rate ∞. If one wishes to avoid rate ∞ transitions, things are more involved, but suitable definitions can still be given in the settings of relevance in this paper.

of these auxiliary states leading to the target, $f(t')$. This can also be viewed as a very strong form of *weak simulation* of one Markov chain by another (see, e.g., [31]), but is substantially stronger than standard notions of weak simulation.

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 $M(\mathcal{A})$. We now define what it means for different classes of finitely presented Markov chains to be "essentially equivalent" (called M-equivalent).

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.[4]

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 $((s,\gamma), p_{(s,\gamma),(s',C)}, (s', C))$, where based on the control state s and the symbol on top of the stack, γ, with probability $p_{(s,\gamma),(s',C)}$, the transition updates the control state to s', and performs action C on the stack. Specifically, if $C = swap(\gamma')$ then the action swaps the top-of-stack symbol, γ, with symbol γ'. If $C = swap\&push(\gamma', \gamma'')$, then the action both swaps γ with γ' and then pushes γ'' on top of the stack.[5] Lastly, if $C = pop$, then the action pops the top-stack-symbol γ off the stack. Each such transition has an associated probability $p_{(s,\gamma),(s',C)}$, and we assume that for each pair (s,γ) of control state and top of stack symbol, $\sum_{(s',C)} p_{(s,\gamma),(s',C)} = 1$. We assume there is a special stack symbol $\perp \in \Gamma$ that marks the bottom of the stack.

[4] In some cases, an aperiodic irreducible chain may be turn into a periodic one, or vise-versa, by the embedding (because the embedding can convert a transition into a sequence of two or more transitions), but this is a minor issue and the original steady-state distribution, if it exists, can be recovered from the uniquely determined stationary distribution of the embedded chain.

[5] Note that the standard push transition $((s,\gamma), p_{(s,\gamma),(s',push(\gamma'))}, (s', push(\gamma')))$ can be trivially encoded as $((s,\gamma), p_{(s,\gamma),(s',swap\&push(\gamma,\gamma'))}, (s', swap\&push(\gamma,\gamma')))$.

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 $\Delta' = \{((w, s), p, (w', s')) \mid ((s, \gamma), p, (s', C)) \in \Delta$ & applying action C to w yields $w'\}$. It was shown in [14] 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), $\mathcal{A}$, 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: one would typically define them 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, $\mathcal{A}$, is 3-tuple $\mathcal{A} = (S, \delta, \delta_0)$ 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 nonzero, 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 does not 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 $\mathcal{A}$. 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, $\mathcal{A}$, generates a denumerable-state Markov chain $M(\mathcal{A}) = (V', \Delta')$ 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. (Under the insignificant technical assumption that counter values in states of a p1CA are encoded in unary.)

A *1-counter automaton* (1CA) is just a p1CA without probabilities, i.e., the transition relation is non-deterministic. So, a 1CA $\mathcal{A} = (S, \delta, \delta_0)$, has transition relations $\delta \subseteq S \times \{-1, 0, 1\} \times S$, and $\delta_0 \subseteq S \times \{0, 1\} \times S$. To each p1CA, $\mathcal{A} = (S, \delta, \delta_0)$, we can associate an *underlying* 1CA, $\mathcal{A}' = (S, \delta', \delta'_0)$, which ignores probabilities of transitions and treats them non-deterministically. Specifically, a transition $(u, c, v) \in \delta'$ ($\in \delta'_0$) iff $p^{(c)}_{u,v} > 0$, ($q^{(c)}_{u,v} > 0$, 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 *nonzero path* if $n_i > 0$ for all $i \in \{1, \ldots, r-1\}$. (Note

that we allow $n_r = 0$ in nonzero paths.) Such a (nonzero) path is called a *(nonzero) terminating path* if $n_r = 0$, and if so it is said to terminate in state $(s_r, 0)$. For p1CAs, $\mathcal{A}$, we define paths, nonzero paths, etc., as simply the paths, nonzero paths, etc., in the underlying 1CA. Note that for a p1CA, the probability that a particular nonzero 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 $M(\mathcal{A})$ is precisely $\prod_{1 \leq i < r} p_{s_i s_{i+1}}^{(n_{i+1} - n_i)}$.

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:[6]

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

where $B_0, B_1, A_{-1}, A_0, A_1 \in \mathbb{R}_{\geq 0}^{m \times m}$. 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 $1 \leq i \leq m$ 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., [26]), 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 $G \leq G'$ (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 $\leq n$. 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., [24]). Specifically, if for $n \geq 0$ we let $\boldsymbol{\pi}_n$ denote an m-vector whose i'th entry is the steady-state probability of being in (i, n), then $\boldsymbol{\pi}_{n+1} = \boldsymbol{\pi}_1 R^n$, for $n \geq 1$, and $\boldsymbol{\pi}_0, \boldsymbol{\pi}_1$ are

[6] 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.

the unique solution to the following system of equations:

$$\boldsymbol{\pi}_0 = \boldsymbol{\pi}_0 B_0 + \boldsymbol{\pi}_1 A_{-1}$$
$$\boldsymbol{\pi}_1 = \boldsymbol{\pi}_1 B_1 + \boldsymbol{\pi}_1 A_0 + \boldsymbol{\pi}_1 R A_{-1}$$

with the normalization condition $\boldsymbol{\pi}_0 \mathbf{1} + \boldsymbol{\pi}_1 (I - R)^{-1} \mathbf{1} = 1$, where $\mathbf{1}$ is the all 1 vector (provided that $\sum_{i \geq 0} R^i$ converges).

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 (as in [34]), 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 labeled 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) 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 $k, s \in \Gamma$. Thus, in a TL-QBD, there are no direct transitions from a state (wk, i) to a state (ws, j), where $k \neq s$, and if there is a direct transition from state (wk, i) to state (wk, j), with probability p, then there is a direct transition from state (ws, i) to (ws, j) with the same probability p. In other words, the probability of transition from control state i to control state j, while not changing the "stack", does not depend on the topmost (rightmost) symbol on the "stack".

3. Equivalences and basic consequences

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

1. *For every QBD $\mathcal{Q}$, there is an easily (linear time) computable pPDS $\mathcal{P}$, with only one stack symbol, such that $\mathcal{Q}$ efficiently embeds into $M(\mathcal{P})$. Moreover, $|\mathcal{P}| = O(|\mathcal{Q}|)$, where the size of $\mathcal{Q}$ is measured in terms of the size of the input matrices B_0, B_1 and A_{-1}, A_0, A_1.*

2. *For every pPDS $\mathcal{P}$ with one stack symbol we can compute (in linear time) matrices matrices B_0, B_1 and A_{-1}, A_0, A_1, yielding a QBD, $\mathcal{Q}$, such that $M(\mathcal{P})$ efficiently embeds in $\mathcal{Q}$. Moreover, $|\mathcal{Q}| = O(|\mathcal{P}|)$.*

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 $1 \leq i, j \leq k$:

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

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 $swap\&push(\bot, \gamma)$ 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*, $swap(\gamma)$ and $swap\&push(\gamma, \gamma)$ transitions as giving the matrices A_{-1}, A_0, and A_1, respectively. □

Obviously TL-QBDs are a special case of TS-QBDs. Furthermore, TS-QBDs are themselves a special case of pPDSs (equivalently, RMCs [14]), where transitions are constrained as follows:

- For every transition of the form $((s, \gamma), p_{(s,\gamma),(s',C)}, (s', C)) \in \Delta$, where $C = swap\&push(\gamma', \gamma'')$, we must have $\gamma = \gamma'$. In other words, every "swap and push" operation must be just a "push" operation.
- Furthermore, we must have $p_{(s,\gamma),(s',swap\&push(\gamma,\gamma''))} = p_{(s,\gamma'),(s',swap\&push(\gamma',\gamma''))}$ for all stack symbols $\gamma, \gamma' \in \Gamma$. In other words, the probability of the "push" does not depend on the top stack symbol.

It should be clear that pPDSs with the above restriction are isomorphic to TS-QBDs, under the mapping that maps a state (w, i) of a TS-QBD to the state $(\perp w, i)$ of the corresponding pPDS. We shall show that all pPDSs can be efficiently embedded in TL-QBDs, and thus that all three models are M-equivalent.

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

1. *Every TL-QBD as well as every TS-QBD, $\mathcal{Q}$, is a (special form of) pPDS.*
2. *For every pPDS $\mathcal{P}$ we can compute (in quadratic time) a TL-QBD (and thus a TS-QBD), $\mathcal{A}$, such that $M(\mathcal{P})$ efficiently embeds in $\mathcal{A}$, Moreover, $|\mathcal{A}| = O(|\mathcal{P}|)$.*

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, we have 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 mimicked 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: $((q, \gamma), p_{(q,\gamma),(q',C)}, (q', C)$ of the pPDS, where $C = swap\&push(\gamma', \gamma'')$, shall be mimicked 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 $p_{(q,\gamma),(q',C)}$ $(= d_{\gamma}^{(q,\emptyset,1),(q',\gamma'\gamma'',2)})$ 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. □

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* (the SQRT-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 ([17]), with many applications. See ([14]) for more background.

Corollary 3. *1. ([13, 37]) The quantitative model checking problem for QBDs and TL-QBDs, against a linear-time (ω-regular or Linear Time Logic (LTL)) property, is decidable in* PSPACE *in the size of the model.*

2. ([14, 16]) The SQRT-SUM *problem is polynomial time reducible to the problem of approximating the termination probabilities (the analog of the G matrix entries) for TL-QBDs, 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. ([20]) There are TL-QBDs for which at least exponentially many iterations of the (decomposed) Newton's method ([14]), applied to the nonlinear 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.

Theorem 4. *The* SQRT-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 [14] 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 has 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, $G_{u,v} = \sum_i (1/n)\sqrt{d_i}/m = 1/(nm) \sum_i \sqrt{d_i}$. □

4. Structural properties of (p)1CAs (or QBDs)

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 $\mathtt{mp}(s, s')$ denote the length of the shortest

terminating path starting at state $(s, 1)$ and terminating at state $(s', 0)$. Likewise, let $\mathtt{mp_{n-z}}(s, s')$ denote the length of the shortest *nonzero* terminating path starting at $(s, 1)$ and terminating at $(s', 0)$. If there is no such (nonzero) terminating path, then by definition $\mathtt{mp}(s, s') = \infty$ ($\mathtt{mp_{n-z}}(s, s') = \infty$, respectively). By convention, a path with a single state has length 0. The next lemma 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 $\mathtt{mp_{n-z}}(s, s') = \infty$ or else $\mathtt{mp_{n-z}}(s, s') \leq k^3$. 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 nonzero 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 nonzero 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 does not 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 does not 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 nonzero 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 nonzero 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 nonzero terminating because otherwise it would not be the shortest nonzero terminating path. Therefore the length of the shortest nonzero terminating path from $(s, 1)$ to $(s', 0)$ is $\mathtt{mp_{n-z}}(s, s') \leq k^3$.

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 nonzero 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 $|\pi| \leq k^4$. □

Let us now show two examples for which such a shortest terminating path between two control states has length $\Theta(k^2)$:

Example 1. Let us consider the 1CA, $A = (S, \delta, \delta_0)$, where $S = \{s_1, s_2, \ldots, s_{2k}\}$. We have $(s_{2k}, -1, s_{k+1}) \in \delta$, and for $i \leq k$ we have $(s_i, 1, s_{i+1}) \in \delta$, and for $k+1 \leq i \leq 2k-1$ we have $(s_i, 0, s_{i+1}) \in \delta$. (Transitions in δ_0 are irrelevant to our analysis.) The shortest path from $(s_1, 1)$ terminating at $(s_{k+1}, 0)$ has length $k^2 + k$. The length of this path in relation to the number of control states k' (equal to $2k$) is $\frac{1}{4}k'^2 + O(k')$. □

Example 2. Let us consider the 1CA, $A = (S, \delta, \delta_0)$, where $S = \{s_1, \ldots, s_k, s'_1, \ldots, s'_{k+1}\}$, $(s_k, 1, s_1) \in \delta$, $(s_k, 0, s'_1) \in \delta$, $(s'_{k+1}, -1, s'_1) \in \delta$, and for $i \leq k-1$ we have $(s_i, 1, s_{i+1}) \in \delta$, and for $i \leq k$ we have $(s'_i, -1, s'_{i+1}) \in \delta$. In other words:

$$\delta = \{(s_1, 1, s_2), (s_2, 1, s_3), \ldots, (s_k, 1, s_1), (s_k, 0, s'_1),$$
$$(s'_1, -1, s'_2), (s'_2, -1, s'_3), \ldots, (s'_{k+1}, -1, s'_1)\}$$

We would like to find the length of the shortest path between $(s_1, 1)$ and $(s'_1, 0)$. Notice that each such path visits only control states s_i until it reaches for the first time s'_1 from s_k and from that point onwards it visits only control states s'_i. It is not hard to see that since k and $k+1$ are relatively prime the length of the shortest such path is $2k(k+1)$, and that the transition from s_k to s'_1 on this path takes place when the counter value is $k(k+1)$. This shows that the analysis in Lemma 5 of the highest possible value of a counter along any shortest terminating path is tight (up to a multiplicative constant). □

For a p1CA, $A = (S, \delta, \delta_0)$, and a pair of states $s, s' \in S$, recall that the termination probability, $G_{s,s'}$ is the probability that, starting from state $(s, 1)$, a random walk on the chain $M(A)$ will traverse a nonzero path that eventually visits and terminates in state $(s', 0)$. Given the equivalence of p1CAs and QBDs, the probabilities $G_{s,s'}$ yield precisely the well known G matrix associated with the QBD (or, equivalently, p1CA). We now use Lemma 5 to give a "polynomial size" lower bound on positive termination probabilities $G_{s,s'}$, associated with a p1CA (and a QBD).

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.*[7] *For any pair of states $s, s' \in S$, either $G_{s,s'} = 0$ or $G_{s,s'} \geq p_{min}^{k^3}$.*

Proof. Indeed, $G_{s,s'} > 0$ iff there is a nonzero 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 $p_{min}^{k^3}$. □

[7] In other words, we have $(u, p_{min}, c, v) \in \delta$ for some u, v, c, and $p_{min} > 0$, and for any transition $(u', p', c', v') \in \delta$, with $p' > 0$, we have $p_{min} \leq p'$.

For a pair of states $u, v \in S$, let x_{uv} be a variable denoting the (unknown) probability, $G_{u,v}$. It is well known (e.g., [26]) that the termination probability matrix G is the least non-negative solution of the following matrix equation: $X = A_{-1} + A_0 X + A_1 X^2$. We can of course equivalently write this as a system of polynomial equations, one for each variable x_{uv}, of the following form:

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

We can clean up this system of equations for G by removing the variables x_{uv} for which $G_{u,v} = 0$, and also removing the corresponding equation whose left hand side is such a variable. This can be done in polynomial-time, even for more general fixed point equations associated with pPDSs and RMCs (see [14]). (After clean-up, the equations may no longer have the simple matrix form.) Henceforth, we consider only cleaned-up equation systems, where only nonzero variables remain.

Based on this equation system we can build a dependency graph, $D = (\tilde{X}, E)$, whose nodes are all nonzero 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 *nonlinear 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 nonlinear. 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.

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 nonzero 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 nonzero 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 $(s, n_1) \ldots (t, n_l)$ is a nonzero path that witnesses this. Then note that, for any $d > 0$, $(s, n_1 + d) \ldots (t, n_l + d)$ is also a nonzero path in the same (p)1CA. We will exploit this fact repeatedly.

Proposition 7. *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 $\xrightarrow{k_2}$. □

Example 3. 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 at the same time $k_2 \geq 0$. For instance, when $\delta = \{(u, 1.0, -1, v),\ (v, 1.0, 1, w)\}$, we have $u \xrightarrow{-1} v \xrightarrow{1} w$, but not $u \xrightarrow{0} w$. □

Proposition 8. *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 state $(w, 1)$ has to increase the counter since otherwise it would not be the last state on the nonzero path with counter value 1. So let the next state be $(z, 2)$. From that state the nonzero 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 $u \xrightarrow{1} s \xrightarrow{-1} t \xrightarrow{-1} v$.

Theorem 9. *If the clean equation $x_{uv} = \alpha$, for a variable $x_{uv} \in X_i$ is nonlinear in the variables belonging to X_i, and if the clean equation for a variable $x_{st} \in X_j$ is nonlinear 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. This proof is long. We will first prove a sequence of four Lemmas, 10 –13, and only then return to finish off the proof of the Theorem. 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 10. *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 7 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 11. *If x_{wv} is a nonzero variable and $u \xrightarrow{0} w$ then x_{uv} is also nonzero 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 nonzero 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 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 nonzero 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 cannot 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 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 $x_{s_n v} (\equiv x_{wv})$. □

Example 4. Notice that the assumption about the value of x_{wv} being nonzero is crucial even if we know that x_{uv} is nonzero. For instance in the following example: $\delta = \{(u, 0.5, 0, w),\ (u, 0.5, -1, v),\ (w, 1.0, 1, w)\}$ we have that $x_{uv} = 0.5 > 0$ and $u \xrightarrow{0} w$, but x_{uv} does not depend on x_{wv} since its value is zero. □

Lemma 12. *A nonzero variable x_{uv} depends on the value of a nonzero 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 10 for some k, such that $1 - \delta_{vt} \leq k \leq k'$, we have $u \xrightarrow{k} s \xrightarrow{-1} t \xrightarrow{-k} v$.

($\Leftarrow$) Of course x_{uv} and x_{st} are both nonzero 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 11 we get that x_{ut} ($\equiv x_{uv}$) is nonzero 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 nonzero. From Proposition 8 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 look as follows: $u \xrightarrow{0} w \xrightarrow{+} z \xrightarrow{0} s \xrightarrow{-1} t \xrightarrow{-1} v$. Furthermore, $z \xrightarrow{-1} t$ and $w \xrightarrow{-1} v$, so x_{zt} and x_{wv} are nonzero. 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 11 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 8 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 nonzero and depends on x_{st} and from the induction assumption for $k = k'$ we get that x_{uv} is nonzero and depends on x_{wz}. This means that x_{uv} also depends on x_{st}. □

Example 5. 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 13. *If the clean equation for a variable $x_{uv} \in X_i$ is nonlinear 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 nonlinear 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 10 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 7 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 9. Using Lemma 13 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 7.

Since there is a path from x_{uv} to x_{st} then from Lemma 10 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 12 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: $s \underbrace{\xrightarrow{l_0} s \xrightarrow{l_0} s \xrightarrow{l_0} \ldots \xrightarrow{l_0}}_{n \text{ times}} s$. 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 after that 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 $nl_0 > k + k_0 + k_1$ (this can be done since $l_0 \geq 1$) then using Proposition 7 we can rewrite it as $s \xrightarrow{nl_0 - k + k_1} u \xrightarrow{-1} v \xrightarrow{-nl_0 + k - k_1} t$. Essentially, we make the value of the counter sufficiently high at the beginning of the path in order to prevent it from reaching counter value 0 before it reaches the final t state (with $(w, nl_0 - k - k_0 - k_1)$ being the state with the lowest value of the counter before that point). Now finally, since $nl_0 - k + k_1 \geq 1$, it follows from Lemma 12 that x_{st} depends on x_{uv}. □

Corollary 14. *In the DAG, H, along any directed path $X_{i_1} X_{i_2} \ldots X_{i_r}$ of SCCs there is 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 nonlinear 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 9 that there is also a path from y to x. But that implies x and y are in the same SCC. □

In Figure 1, we can see what Corollary 14 implies for the decomposition DAG, H, of the underlying equation system for p1CAs, namely any path in H can contain at most one nonlinear SCC. Notice that this fact does not hold for general pPDSs and RMCs, nor even for 1-exit RMCs (equivalently, pBPAs).

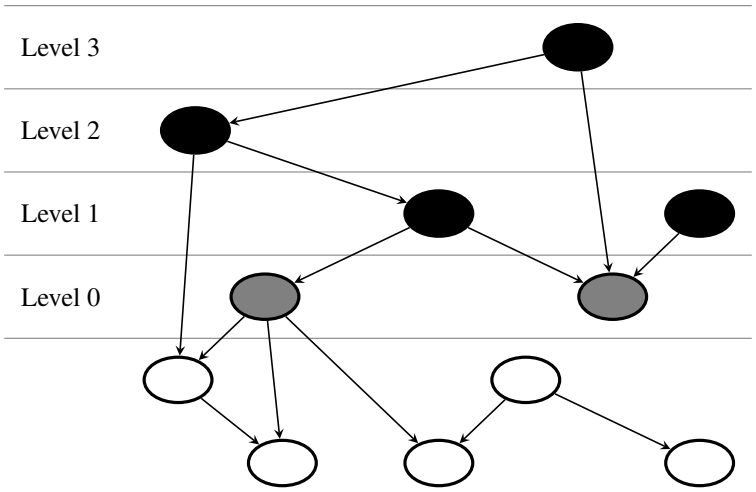

Figure 1: This picture shows how the DAG of SCCs, H, of the dependency graph of the equation system corresponding to a p1CA (QBD) might look. Each node represents one SCC: the equation systems for all white SCCs are (internally) linear, for gray SCCs the equations are (internally) nonlinear and for black SCCs they are (internally) linear again but their constants depend (possibly indirectly) on values in at least one gray nonlinear SCC. Note that along any directed path in H there is at most one gray nonlinear SCC. The length of the longest path from a given black SCC to some nonlinear SCC determines its "level". The "height" of the DAG is the largest level of any black SCC. In this example the height of the decomposition DAG, H, is $h_{max} = 3$.

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 potential blow-ups involved in the encoding size of rational numbers.

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 $F(\mathbf{x}) \doteq P(\mathbf{x}) - \mathbf{x} = \mathbf{0}$. Note that these are polynomial functions, and thus certainly differentiable.

We shall solve this system of equations using the *decomposed Newton's method* of [14], 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 [14]) we form the dependency graph D for the nonzero 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. (See [14] for details.)

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 of p1CAs and QBDs (see Figure 1). By Corollary 14, for any nonlinear 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 [14] (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" to get to within a desired additive error $\epsilon > 0$ of the nonzero termination probabilities $G_{u,v}$ for the variables in a nonlinear SCC? 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 strongly connected 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 15. *([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 nonzero transition probability is $p_{min} \geq 1/2^m$.

Now, using Theorem 15, together with the structural properties we have established for p1CAs, we prove the following strong bound on the number of iterations of Newton's method required to get i valid bits of precision of the termination probabilities $G_{u,v}$, for the nonlinear SCCs of the fixed point equations associated with p1CAs:

Theorem 16. *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 nonlinear equations for a p1CA is quadratic, the smallest coefficient c_{min} of any monomial in the system $X = P(X)$ for this nonlinear 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 15 can be seen to be $k_f \leq n^2 \cdot \log(2^{2mn^3+m} 2^{mn^3} 2^{mn^3}) = 4mn^5 + mn^2$. □

Theorem 16 implies that we can compute i bits of the values $G_{u,v}$ for variables x_{uv} in nonlinear 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 nonlinear SCCs in H. Specifically, there may be a linear SCC X_r, from which there is a path in H to a nonlinear 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 17. *(see, e.g., [19], Chap 2.1.2, Thm 3.*[8]*) 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 [14] about the decomposed systems of equations arising for RMCs (pPDSs) (specifically, see Lemma 17 and Theorem 14 of [14]), 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:

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

Lemma 18. *Let $A \in \mathbb{R}^{n\times n}_{\geq 0}$ and $b \in \mathbb{R}^{n}_{\geq 0}$, such that: $(I-A)^{-1} = \sum_{k=0}^{\infty} A^k$, and we have $(\sum_{k=0}^{\infty} A^k)b \leq \mathbf{1}$, and A is an irreducible non-negative matrix whose smallest nonzero entry is $c > 0$, and $b \neq 0$ and $p > 0$ is the largest entry of b. Then: $\| \sum_{k=0}^{\infty} A^k \|_\infty \leq \frac{n}{pc^n}$.*

Proof. Let a^d_{ij} 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 $a^d_{ij} > 0$. First, notice that it has to be $c < 1$ as otherwise all entries of $(\sum_{k=0}^{\infty} A^k)$ would diverge to ∞. Furthermore, since the smallest nonzero entry of A is c, we have $a^d_{ij} \geq c^d$.

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, $a'_{i1} \leq n a^*_{i1} \leq n/p$. On the other hand:

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

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 $\|A^*\|_\infty \leq \frac{n}{pc^n}$. □

Lemma 19. *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 (again we use m to denote the maximum number of bits required to represent the numerator and denominator of rational transition probabilities in the p1CA). 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. $\mathrm{cond}(I-A) \leq \frac{2n^3}{p_{min}^{5n^5}} \leq 2n^3 \cdot 2^{5mn^5}$
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 $\sum_{c=-1}^{1} \sum_w p_{uw}^{(c)} \leq 1$, the full sum $\sum_j a_{ij}$ of all entries in row i of A corresponding to the variable x_{uv}, cannot 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 nonzero, 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 nonzero 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 $p_{uv}^{(c)}$ or to $p_{uv}^{(c)} \cdot x_{wz}$ where x_{wz} is a variable from a lower SCC that has been substituted by a constant. We now use Lemma 18. 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 18, using the bound $\|b\|_\infty \geq p_{min}^{2n^3+1}$, and the fact that all nonzero entries of A are $\geq (p_{min})^{n^3+1}$, that $\|(I - A)^{-1}\|_\infty = \|\sum_{k=0}^{\infty} A^k\|_\infty \leq \frac{n^2}{p_{min}^{5+n^2+2n^3+1}} \leq \frac{n^2}{p_{min}^{5n^5}}$. 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 cannot 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 $p_{min}^{2n^3+1} \leq \|I - A\|_\infty$. □

For a "higher" linear SCC, X_r, i.e., one which can reach some nonlinear 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 nonlinear SCC that it can reach (again, see Figure 1). Let $h_{max} = \max_r h_r$, where the maximum is taken over all linear SCCs that can reach a nonlinear 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 nonlinear bottom SCCs remain, or all SCCs are solved.

2. For each remaining nonlinear SCC, apply Newton's method (starting with vector $x^0 = 0$) to the nonlinear 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 nonlinear 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 20. *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., discrete 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 nonlinear 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 nonlinear SCCs, by Theorem 16, the values $G_{u,v}$ for variables x_{uv} in nonlinear 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 nonlinear 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 nonlinear 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 nonlinear 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 nonlinear 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 17, 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 19, 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?

First, let us consider the original quadratic equation $x_{uv} = \alpha'$ before some of the variables $x_{s't'}$, those from lower SCCs, have been substituted by their approximate value. A linear expression containing x_{st} in α' can only result from a monomial term in α of the form px_{st} or $px_{s't'}x_{st}$. In the first case the coefficient p would contribute its exact value to $a_{uv,st}$, so it would add zero to the absolute error of $a_{uv,st}$. However, in the second case, since $p \leq 1$ and the value of $x_{s't'}$ is an under-approximation of $G_{s',t'}$ up to W_{h-1} bits of precision, then the coefficient $a_{uv,st}$ could be under-approximated by at most $1/2^{W_{h-1}}$. As we can see, an absolute error of at most $1/2^{W_{h-1}}$ can arise from each such monomial. Next, note that the coefficient $a_{uv,st}$ of x_{st} in the equation (1) for x_{uv} may actually arise as a sum of at most $n + 2$ such monomials:

- ⋆ if $t = v$ (in other words $x_{st} \equiv x_{sv}$) then we can have one of the form: $p^{(0)}_{us}x_{sv}$, and there can be n other monomials, one for each control state w: $p^{(1)}_{uw}x_{ws}x_{sv}$. Moreover, if $t = v$ and $s \neq v$ then we can have one extra monomial term of the form: $p^{(1)}_{us}x_{sv}x_{vv}$ (if $s = v$ then this expression is counted already when $w = s$ above).
- ⋆ if $t \neq v$ then we have at most one monomial involving x_{st} of the form: $p^{(1)}_{us}x_{st}x_{tv}$

We note now that the sum of all these coefficients of x_{st} is always smaller than 2 since:

$$p^{(0)}_{us} + p^{(1)}_{us}x_{vv} + \left(\sum_w p^{(1)}_{uw}x_{ws}\right) \leq \left(p^{(0)}_{us} + p^{(1)}_{us}\right) + \sum_w p^{(1)}_{uw} \leq 1 + 1$$

Furthermore, if $n = 1$ then there can be only one SCC. Hence, in such a case $h_{max} = 0$ and we would be done. As a consequence, from now on, we assume that $n \geq 2$ (which holds, except in the trivial case) allowing us to conclude that: $\mathcal{E}_{uv,st} \leq 2/2^{W_{h-1}} \leq n/2^{W_{h-1}}$.

We can ask a similar question about b. Since a constant term may arise because both variables in a quadratic 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 $(x-\varepsilon)(x'-\varepsilon) \geq xx' - 2\varepsilon$. 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 19, 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 19, part (4.), we can conclude that $\frac{\|\zeta\|_\infty}{\|b\|_\infty} \leq \frac{2n2^{2mn^3+m}}{2^{W_{h-1}}}$. Next, by Lemma 19, 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 19, part (3.), $\mathrm{cond}(I-A) \leq 2n^3 \cdot 2^{5mn^5}$.

Now we use these bounds and apply Theorem 17. 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 17.

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

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). □

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.

6. Comparison of PReMo with SMCSolver

In this section we briefly describe some experiments conducted with the tool PReMo (**P**robabilistic **Re**cursive **Mo**dels analyzer) ([36]). PReMo allows the user to specify and analyze abstract models of probabilistic procedural programs and other systems that involve recursion and probability. PReMo can analyze p1CAs (equivalently QBDs), and more generally it can analyze Recursive Markov Chains (RMCs), or equivalently pPDSs, TL-QBDs, and TS-QBDs, and even more general monotone systems of nonlinear equations. It can also analyze controlled and game extensions of certain important subclasses of RMCs.

As discussed in the introduction, we performed some experiments to compare the performance of the tool PReMo with the state of the art tool for analysis of QBDs — SMCSolver [3] (Structured Markov Chains solver). These two tools are very different in a number of ways. They differ in how equation systems are represented, the implemented numerical algorithms, and the implementation language. PReMo is implemented entirely in JAVA, and each equation in the system of equations corresponding to a model is represented as an explicit algebraic formula (which allows handling arbitrary monotone systems of nonlinear equations, which may even include operators other than standard arithmetic operators). On the other hand, SMCSolver makes use of a concise matrix representation for the entire equation system and is implemented in FORTRAN and Matlab, programming languages geared towards numerical computation. The Matlab version of SMCSolver has many more numerical approximation algorithms than PReMo. PReMo's fastest numerical algorithm in practice is a sparse version of (decomposed) Newton's method, and (undecomposed) Newton's method is implemented in SMCSolver only in its Matlab version. The most efficient numerical methods for analysis of QBDs implemented in SMCSolver are: *Cyclic Reduction* and *Logarithmic Reduction*. These algorithms can be further sped up by using a *shifting technique* to achieve "quadratic" convergence in the case of null recurrent QBDs (see, e.g., [3, 2]). Cyclic and Logarithmic Reduction were later modified and applied to TL-QBDs (see [4, 2]) which as we have observed are equivalent to RMCs, so they can in principle also be used for analysis of RMCs. However, it should be noted that the systems that arise for analysis of probabilistic procedural programs will typically have a very sparse transition structure, which may not suit the matrix equation representation used in SMCSolver. SMCSolver has two implementation: one, with a graphical user interface, written in FORTRAN and another one as a collection of Matlab functions that can be run from the command-line (a Matlab toolbox). The Matlab version appears to usually be slightly faster for small dense matrices and a lot faster for larger matrices. Moreover, it manages memory much better: the FORTRAN version crashes for matrices of size 5000 while the Matlab version does not. Because of this, we will focus in our comparison on the Matlab version of the tool.

To fairly compare the underlying algorithms, we do not include in the running time of PReMo the parsing time of the input equation system. This is because when any iterative solution method in SMCSolver is initiated, the whole equation system is stored already in main memory and does not have to be further preprocessed.

We compared the most efficient solvers of SMCSolver and PReMo, namely Cyclic/Logarithmic Reduction (with and without the shift acceleration), and the Sparse decomposed Newton's method, respectively. The sparse Newton's method was set to use the Biconjugate Gradients method to solve the sparse linear system of equations that occurs in each iteration of Newton's method.

On most dense examples, SMCSolver is far superior to PReMo, often by an order of magnitude or more. In particular, we tested PReMo on SMCSolver's built-in examples, whose transition matrices are dense, and where all the variables form just one big SCC. For SMCSolver's Example 1, with 100 control states, it took Cyclic Reduction about 0.1 seconds and Logarithmic Reduction about 0.2 seconds to converge to

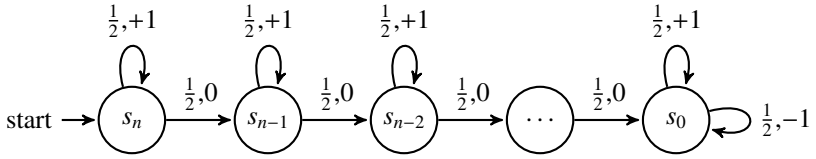

Figure 2: A p1CA with n control states and whose A_{-1} matrix has rank 1.

the solutions with the desired tolerance[9] (which can typically be set to, e.g., 10^{-12}). On the other hand, PReMo's Sparse Newton method (using the Biconjugate Gradients method per iteration) needed 98.5 seconds to converge to the same solution with the same tolerance. For SMCSolver's Example 3 with 100 control states the running times for SMCSolver were about the same, 0.1 second, while in PReMo the Sparse Newton's method finished in 46.8 seconds. To explain this, it has to be noted that in the Example 1, from each control state there is a direct transition of all three possible types (increment,decrement, or keep counter unchanged) to any other control state and this results in a huge equation system, which is represented explicitly in PReMo as algebraic formulas. More precisely, if k is the number of control states then the equation representation as algebraic formulas grows as $O(k^3)$ compared to $O(k^2)$ when represented in SMCSolver's matrix form. Although the transition matrix is not dense for Example 3, the underlying equation system and the G matrix are dense, because from any control state we can terminate at any other control state with positive probability. This leads to both dense equations which require $O(k^3)$ encoding size, and to a dense G matrix whose encoding size is $O(k^2)$.

In order to highlight how sparsity and decomposability of the equations can improve PReMo's performance, we tested both tools on various sparse examples. Two such families of examples are depicted in Figure 2 and Figure 3. In the example from Figure 2, parameterized by the number of control states n, starting at any control state we terminate at control state s_0 with probability 1. In the example from Figure 3, we terminate almost surely either at s_1 or s_0, and the probability of termination at s_0 when starting at s_n converges fast to 1 as n increases. The matrix A_{-1} has rank 1 for the family of examples from Figure 2 and rank 2 for examples from Figure 3. SMCSolver has a special routine for QBDs whose matrix A_{-1} or A_1 has rank 1, and each of its numerical method starts by performing such a check first, as a kind of preprocessing step.

These are extreme examples in several ways, both because the equation systems for them are extremely decomposable, and also because they even have a sparse G matrix (i.e., the solution is also sparse). PReMo was able to find the G matrix to within desired tolerance for 5000 control states in about two seconds, and could easily handle much

[9]Here *tolerance* means maximum change in the value of any variable in one iteration. Such a tolerance threshold is the typical way used in practice for determining when to stop an iterative numerical method which converges in the limit to the desired solution. As we will see later, the actual running time of the numerical methods in question often does not depend significantly on this value once it is below some reasonable threshold, say 10^{-4}.

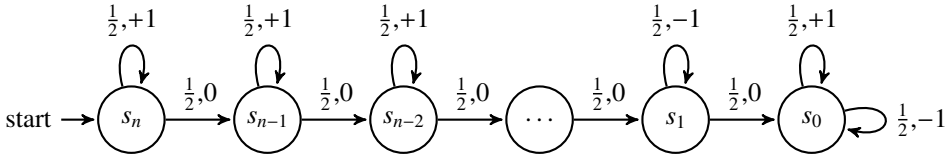

Figure 3: A p1CA with n control states and whose A_{-1} matrix has rank 2.

bigger examples. SMCSolver needed more than an hour for such a big example from Figure 3 and when the number of control states was reduced to 1000, its most efficient method (shifted Cyclic Reduction) took 30.4 seconds to find a solution with the desired tolerance. Even when the tolerance was reduced to 10^{-4} (as compared to the default one: 10^{-14}) the time needed by shifted Cyclic Reduction for 5000 control states was more than an hour (precisely 4476 seconds as compared to the original 4794 seconds needed before). The reason for this is clear. With 5000 control states, there are already 25 million entries in the matrices using SMCSolver's dense matrix representation, most of which are 0. Computing anything with such huge matrices is a problem. But by using decomposition methods we can avoid this. The running times of all mentioned numerical methods for all mentioned examples are presented in Table 1.

Examples *name-size*	SMC Fortran		SMC MATLAB					PReMo
	CR	LR	CRbasic	CR	LRbasic	LR	Newton Sylvester	Sparse Newton
ex1-100	0.064	0.069	0.085	0.059	0.128	0.193	0.078	98.5
ex3-100	0.97	0.110	0.138	0.0865	0.177	0.105	0.334	46.79
fig2-1000	207	290	7.25	7.22	7.22	7.21	7.28	0.260
fig2-5000	!	!	782	785	783	781	785	1.134
fig3-1000	206	198	⋆	30.4	⋆	39.2	1370	0.580
fig3-5000	!	!	(> 5h)	4794	(> 5h)	6416	!	2.130

Table 1: Running times (in seconds) of SMCSolver and PReMo with the default tolerance on dense examples: Example 1 (ex1-100), Example 3 (ex3-100), and sparse examples from Figure 2 (fig2-1000, fig2-5000) and Figure 3 (fig3-1000, fig3-5000). Abbreviations used in the table: SMC – SMCSolver; CR – Cyclic Reduction; LR – Logarithmic Reduction; CRbasic & LRbasic – CR or LR method without the shifting technique; Newton Sylvester – Newton's method implemented by solving Sylvester matrix equations at each step; Sparse Newton – decomposed Newton's method; ! – means that the program crashed or ran out of memory; ⋆ – means the program gave up after exceeding its maximum allowed number of iterations and terminated without converging; (> 5h) – means that the program did not manage to terminate within five hours. The uniform running times of Matlab version of SMCSolver for examples from Figure 2 stems from a special handling of input matrices with rank 1.

In conclusion, we can see that PReMo can be faster than SMCSolver for sparse examples that are very highly decomposable. On the other hand, SMCSolver far outperforms PReMo on dense examples, thanks to its concise matrix representation of the underlying equation system, and by using highly optimized linear matrix algebra software for such matrix equations. This gives rise to the following question: is it possible to combine algorithms that operate on the matrix formulation of the equation systems, together with methods that decompose the equations into SCCs, in order to

gain the benefits of both approaches for (TL-)QBDs and RMCs? Newton's method can 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 each iteration 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. However, even if such an efficient method were found, it remains a difficult challenge to combine this well with decomposition methods, because in general decomposition destroys the matrix form of the equations.

7. Concluding remarks

We began by observing the close relationship between probabilistic models studied in different research communities: in queueing theory and performance evaluation on the one hand, and in the recent research on analysis of probabilistic procedural programs on the other. In particular, we observed the equivalence between QBDs and p1CAs. Our main result was a new upper bound on approximation of central quantities associated with QBDs. Specifically, we showed that, given a QBD and i, the basic G matrix of "termination probabilities" for the QBD can be approximated to within i bits of precision (i.e., with maximum additive error $\leq 1/2^i$) in time polynomial in *both* the encoding size of the QBD and in i, in the unit-cost rational arithmetic RAM (i.e., discrete Blum-Shub-Smale) model of computation. Specifically, we showed that the decomposed Newton's method studied in [14] can be used to achieve this bound.

An important open problem that arises from this work is this: can the polynomial time 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? It was established in [14], that for RMCs and pPDSs, and thus also for Tree-Like QBDs, any non-trivial approximation of the actual termination probabilities of a TL-QBD is at least as hard as the Sqrt-Sum problem and more general arithmetic circuit decision problems. Therefore, no such approximation algorithm can be found for TL-QBDs without a major breakthrough in the complexity of exact numerical analysis. However, this does not rule out the possibility of finding such an algorithm for QBDs. Note that the Sqrt-Sum-hardness result for QBDs that we established in Theorem 4 only applies to the quantitative *decision* problem, which asks whether a termination probability is $\geq p$, and not to the approximation problem. In fact, it is entirely plausible that, using the decomposed Newton's method, but rounding off the computed values after each iteration to some polynomial number of bits, yields such an approximation algorithm for QBDs. Determining whether this is indeed the case will require a detailed analysis of the effect of round-off errors on iterations of Newton's method over the nonlinear equations that arise for QBDs.

On the practical side, at the end of Section 6, we pointed out that an interesting line of future research would be to find methods to combine the benefits of the concise matrix representations employed in, e.g., SMCSolver, and the decomposition methods

employed in PReMo. This is a challenging problem for several reasons. In particular because in general decomposition of equations into SCCs destroys the matrix form of the equations. A related technical challenge in this regard is to find an efficient ($O(n^3)$) method to perform each iteration of Newton's method for RMCs (TL-QBD) based on matrix representations of their equation systems (this is known to be doable for QBDs but not TL-QBDs).

Going beyond the purely stochastic QBD model, in a recent unpublished work with T. Brázdil, V. Brožek and A. Kučera ([5]) we have begun to study the controlled extension of QBDs and p1CAs, 1-counter Markov decision processes (OC-MDPs), and considered the computational complexity of some basic analysis problems for OC-MPDs. In particular we consider the complexity of qualitative termination problems, such as whether there is a strategy under which termination happens with probability 1. Many questions about the complexity of basic analysis problems for this more general OC-MDP model remain open. (For the more general model of (multi-exit) RMDPs, which amount to controlled versions of TL-QBDs and pPDSs, already strong undecidability results have been shown in [15].)

Acknowledgement. Research partly supported by NSF Grant CCF-0728736. We wish to thank an anonymous referee for very helpful comments, particularly about SMC-Solver.

References

[1] R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T. Reps, and M. Yannakakis. Analysis of recursive state machines. *ACM Transactions on Programming Languages and Systems*, 27(4):786–818, 2005.

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

[3] D. Bini, B. Meini, S. Steffe, and B. Van Houdt. Structured Markov chains solver: algorithms/software tools. In *Proceedings 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 and its Applications*, 366:39–64, 2003.

[5] T. Brázdil, V. Brožek, K. Etessami, A. Kučera, and D. Wojtczak. One-counter Markov decision processes. In *Proceedings of 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.

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

[7] T. Brázdil, A. Kučera, and O. Stražovský. On the decidability of temporal properties of probabilistic pushdown automata. In *Proceedings of 22nd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 145–157, 2005.

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

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

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

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

[12] K. Etessami, D. Wojtczak, and M. Yannakakis. Quasi-birth-death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems. In *Proceedings of 5th International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 243–253, 2008.

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

[14] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1), 2009. (Conference version appeared in *Proceedings of 22nd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 340–352, 2005.)

[15] K. Etessami and M. Yannakakis. Recursive Markov decision processes and recursive stochastic games. In *Proceedings of 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 891–903, 2005.

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

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

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

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

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

[21] A. Kučera. The complexity of bisimilarity checking for one-counter processes. *Theoretical Computer Science*, 304:157–183, 2003.

[22] A. Kučera and P. Jančar. Equivalence-checking with infinite-state systems: Techniques and results. In *Proceedings of 28th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 41–73, 2002.

[23] G. Latouche. Newton's iteration for non-linear equations in Markov chains. *IMA Journal on Numerical Analysis*, 14(4):583–598, 1994.

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

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

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

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

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

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

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

[31] R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. In *Proceedings of 5th International Conference on Concurrency Theory (CONCUR)*, pages 481–496, 1994.

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

[33] L. G. Valiant and M. Paterson. Deterministic one-counter automata. In *Automatentheorie und Formale Sprachen*, volume 2 of *Lecture Notes in Computer Science*, pages 104–115, 1973.

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

[35] 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.

[36] D. Wojtczak and K. Etessami. PReMo: an analyzer for probabilistic recursive models. In *Proceedings of 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 66–71, 2007.

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

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

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