Time-Bounded Reachability in Tree-Structured QBDs by Abstraction

Daniel Klink, Anne Remke, Boudewijn R. Haverkort, *Fellow, IEEE,*
and Joost-Pieter Katoen, *Member, IEEE Computer Society*

***Abstract*—This paper studies quantitative model checking of infinite tree-like (continuous-time) Markov chains. These tree-structured quasi-birth death processes are equivalent to probabilistic pushdown automata and recursive Markov chains and are widely used in the field of performance evaluation. We determine time-bounded reachability probabilities in these processes — which with direct methods, i.e., uniformization, results in an exponential blow-up— by applying abstraction. We contrast abstraction based on Markov decision processes (MDPs) and interval-based abstraction; study various schemes to partition the state space, and empirically show their influence on the accuracy of the obtained reachability probabilities. Results show that grid-like schemes, in contrast to chain- and tree-like ones, yield extremely precise approximations for rather coarse abstractions.**

I. Introduction

Probabilistic model checking is a verification technique for Kripke structures in which time and transition probabilities are specified stochastically. Popular models are discrete- and continuous-time Markov chains (DTMC and CTMCs, respectively) and variants thereof that exhibit nondeterminism such as Markov decision processes (MDPs). CTMCs are heavily used in the field of performance evaluation, and since model checking offers various advantages to traditional CTMC analysis techniques, tools such as SMART [8], GreatSPN [9], PEPA Workbench [14], and so on, have been enriched with model-checking facilities. Time-bounded reachability probabilities are at the heart of these model-checkers and are reduced to transient analysis [3]. Hence, efficient algorithms are available for finite-state CTMCs, however, they are not applicable to classes of infinite-state Markov chains such as *tree-structured* quasi-birth death (QBD) processes [27], [31]. Tree-structured QBDs have been applied to model single-server queues with a LIFO service discipline [17], to analyze random access algorithms [28], as well as priority queueing systems [29]. Discrete-time tree-structured QBDs are equivalent to probabilistic pushdown automata [6] and recursive Markov chains [12]. The analysis of (tree-structured) QBDs mostly focuses on steady-state probabilities as these can be obtained using matrix-geometric techniques [5]. Transient analysis has received scant attention; the only existing approach is approximate [30]. Recently, direct techniques based on uniformization or variants thereof [15], have been proposed for reachability properties for general infinite-state CTMCs [32] and for highly structured infinite-state CTMCs, such as normal QBDs [24] and Jackson queueing networks [23]. However, they all lead to an exponential blow-up when applied to tree-structured QBDs. Other related work includes model checking discrete-time infinite-state probabilistic systems [1], [6], [25], [13]. Other abstraction techniques for MDPs such as magnifying lens abstraction [11] and game-based abstraction [21] cannot be easily adapted to the continuous-time setting and are thus not applicable here. Whereas our technique guarantees to yield upper and lower bounds of reachability probabilities, [30] yields arbitrary approximations. In addition, applying transient analysis to compute timed reachability probabilities requires an amendment of the CTMC which destroys the tree structure; therefore [30] cannot be directly applied to our setting.

In this paper, we determine time-bounded reachability probabilities in tree-structured QBDs by applying CTMC abstraction. To that end, we consider two techniques, interval-based [19] and MDP-based abstraction [10], and compare them. A major issue in state-based abstraction is to come up with an adequate, i.e., small though precise, partitioning of the state space. In fact, we identify various possibilities to partition the state space of a tree-structured QBD, relate them formally using simulation relations [4], and empirically investigate their influence on the accuracy of the obtained time-bounded reachability probabilities. The partitioning methods range from simple schemes such as counter abstraction (group states with equal number of customers) to more advanced schemes in which the ordering of customers, and/or the service phase is respected. This yields tree-, chain-, and grid-like abstractions. We perform extensive experiments for phase-type service distributions of different orders and analyze the influence of parameter setting and partitioning scheme on the quality of the results and on the size of the resulting abstract state space. Our experiments show that grid-like schemes yield extremely precise approximations for rather coarse abstractions.

Organization of the paper. Section II introduces CTMCs and tree-structured QBDs and summarizes how to compute time-bounded reachability properties. Section III contrasts interval and MDP abstraction and provides some theoretical underpinnings. Different state-space partitionings of tree-structured QBDs are considered in Section IV. Experimental results on these abstractions are provided in Section V. Finally, Section VI concludes the paper.

D. Klink has been funded by the DFG Research Training Group 1298 (AlgoSyn) and the EU FP7 project QUASIMODO. A. Remke has been funded by the NWO project MC=MC (612.000.311) and by 3TU.CeDiCT.

D. Klink and J.-P. Katoen are with the RWTH Aachen University, Germany. E-mail: {klink, katoen}@cs.rwth-aachen.de.

A. Remke is with the University of Twente, The Netherlands. E-mail: anne@cs.utwente.nl.

B. Haverkort is with the University of Twente and the Embedded System Institute, The Netherlands. E-mail: brh@cs.utwente.nl.

II. TREE-STRUCTURED QBDS

Notation. Let X be a countable set. For $x \in X$, $X' \subseteq X$ and $f : X \times X \to \mathbb{R}_{\geq 0}$ (and similarly for n-dimensional functions), let $f(x, X') = \sum_{x' \in X'} f(x, x')$ and let $f(x, \cdot)$ be given by $y \mapsto f(x, y)$ for all $x, y \in X$. A probability distribution $\mu : X \to [0, 1]$ assigns a probability to each $x \in X$ such that $\sum_{x \in X} \mu(x) = 1$. The set of all distributions on X is denoted *distr*(X).

Continuous-time Markov chains. A *CTMC* is a Kripke structure with randomized transitions and exponentially distributed delays. Formally it is denoted as tuple $(\mathcal{S}, \mathbf{R}, \mu_0)$ with countable state space $\mathcal{S}$, transition rate function $\mathbf{R} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ with $\mathbf{R}(s, \mathcal{S}) \in \mathbb{R}_{\geq 0}$ for all $s \in \mathcal{S}$ and initial distribution $\mu_0 \in$ *distr*$(\mathcal{S})$. Figure 1 shows a CTMC with $\mathcal{S} = \{s_1, s_2, u\}$, $\mathbf{R}(s_1, s_2) = L$, $\mathbf{R}(u, \mathcal{S}) = 0$, $\mu_0(s_1) = \frac{L\downarrow}{L\downarrow + R\downarrow}$ and so forth.

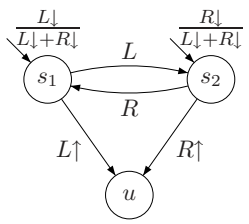


Fig. 1. A CTMC.

By adding self-loops, a CTMC can be transformed into a weakly bisimilar, *uniform* CTMC $(\mathcal{S}, \mathbf{R}', \mu_0)$ where for all $s \in \mathcal{S}$, $\mathbf{R}'(s, \mathcal{S}) = \lambda$ for some $\lambda \in \mathbb{R}_{>0}$ with $\lambda \geq \sup_{s \in \mathcal{S}} \mathbf{R}(s, \mathcal{S})$, cf. [4]. Such a uniform CTMC can be seen as a discrete-time Markov chain $(\mathcal{S}, \mathbf{P}, \mu_0)$ with transition probabilities $\mathbf{P}(s, s') = \mathbf{R}(s, s')/\lambda$ where the probability to perform a certain number of steps within $[0, t]$ follows a Poisson distribution with parameter λt [16]. The probability to reach state s' within t time units is now given by:

$$\sum_{s \in \mathcal{S}} \mu_0(s) \cdot \mathbf{P}(s, s', t), \quad \text{with}$$
$$\mathbf{P}(s, s', t) = \sum_{i=0}^{\infty} \mathbf{P}^i(s, s') \cdot e^{-\lambda t} \frac{(\lambda t)^i}{i!},$$

where $\mathbf{P}^i$ is the i-th power of $\mathbf{P}$. To avoid the infinite summation over the number of steps i, the sum needs to be truncated. For an a priori defined maximum error bound $\varepsilon > 0$, a given time bound $t \in \mathbb{R}_{\geq 0}$ and the uniformization rate λ one can compute the truncation point n_ε. The overall time complexity for computing reachability probabilities up to ε is in $\mathcal{O}(N^2 \lambda t)$ where N is the number of states in the CTMC that are reachable in n_ε steps.

Queuing theory. Large classes of distributions can be approximated arbitrarily closely by *phase-type distributions*. A phase-type distribution of order d (written PH_d) is a probability distribution that can be characterized as the time until absorption in an $(d+1)$-state CTMC [22]. The time until absorbtion in the CTMC in Figure 1 is PH_2 distributed. As example, an M|PH|1 queueing station describes a station with *one* processing unit serving jobs according to a phase-type distribution and with *Markovian*, i.e., exponentially distributed job arrival times.

CTMCs representing PH|PH|1 queuing stations with *first in first out* (FIFO) service discipline correspond to QBD processes [22], therefore their state spaces just grow linearly with the number of queued jobs. This stems from the fact that jobs are not preempted, i.e., jobs are served until completion and only the service phase of the job currently in service needs to be encoded in the state. For such systems, *uniformization with representatives* is a feasible technique to compute transient probabilities [24]. For queuing stations with *preemptive last in first out* (LIFO) service discipline, however, the underlying CTMCs are so-called *tree-structured* QBDs, whose size grows exponentially with the queue length. In the following, for simplicity we restrict ourselves to M|PH|1 queues, however, our approach can also be applied to PH|PH|1 queues in the same manner.

In principle, uniformization with representatives [24] can be adapted to the analysis of tree-structured QBDs. However, for PH_d distributed service times, n uniformization steps and a single starting state, one would have to consider $\mathcal{O}(d^n)$ states, which is practically infeasible. The same holds for techniques based on uniformization like in [7], [32].

Definition 1: A *d-ary tree-structured QBD* $\mathcal{T}$ is a CTMC $(\mathcal{S}, \mathbf{R}, \mu_0)$ with state space $\mathcal{S} = \{(x_1, \ldots, x_n) \mid n \in \mathbb{N} \wedge \forall i \leq n : x_i \in \{1, \ldots, d\}\}$. A state $\vec{x} = (x_1, \ldots, x_n)$ represents a queue of length n where jobs $1, 2, \ldots, n-1$ have been preempted in phase $x_i \in \{1, \ldots, d\}$ and job n is currently in service in phase $x_n \in \{1, \ldots, d\}$.

Transitions, represented by positive entries in $\mathbf{R}$, can only occur between a state and its parent, its children and its siblings (cf. Figure 2). For $\vec{x}, \vec{y} \in \mathcal{S}$:

$$\mathbf{R}(\vec{x}, \vec{y}) = \begin{cases} r_{x_{m+1}}\downarrow & \text{if } \vec{x} = (x_1, \ldots, x_m), \text{ and} \\ & \quad \vec{y} = (x_1, \ldots, x_{m+1}) \\ r_{x_{m+1}}\uparrow & \text{if } \vec{x} = (x_1, \ldots, x_{m+1}), \text{ and} \\ & \quad \vec{y} = (x_1, \ldots, x_m) \\ r_{x_m, y_m} & \text{if } \vec{x} = (x_1, \ldots, x_m), \text{ and} \\ & \quad \vec{y} = (x_1, \ldots, x_{m-1}, y_m) \\ 0 & \text{otherwise} \end{cases}$$

The underlying state space of a preemptive LIFO $\text{M}|\text{PH}_2|1$ queue with overall arrival rate $L\downarrow + R\downarrow$ and service time distribution as depicted in Figure 1 is the (binary) tree-structured QBD shown in Figure 2, where $r_1\downarrow = L\downarrow$, $r_2\downarrow = R\downarrow$, $r_1\uparrow = L\uparrow$, $r_2\uparrow = R\uparrow$, $r_{1,2} = L$, $r_{2,1} = R$. Note that, in contrast to ordinary trees, in tree-structured QBDs, transitions between siblings are allowed.

State $\varnothing = (\ )$ represents the empty queue. Arriving jobs can either enter service phase 1 or 2, represented by the states (1) and (2). Due to the preemptive LIFO service discipline, a

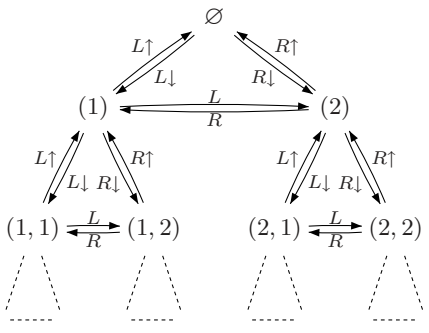


Fig. 2. An example (binary) tree-structured QBD.

new job arrival causes the preemption of the job currently in service and the service phase of the preempted job needs to be stored. This results in a tree-structured state space.

Note that it has been shown in [26], that every tree-structured QBD can be embedded in a binary tree-structured Markov chain with a special structure.

A measure of interest for performance evaluation of M|PH|1 queues is: "if the queue is filled up to a certain level, what is the probability for the system to process all but k jobs within t time units?". New jobs that arrive while serving older jobs should be completed as well. This cannot be answered with steady-state analysis that is typically conducted on such queues. Hence, we resort to computing time-bounded reachability probabilities on tree-structured QBDs.

III. Abstraction

In the previous section, it was argued that direct methods like uniformization are not feasible for analysing transient behavior of tree-structured QBDs. Here, we discuss two abstraction techniques for CTMCs that preserve time-bounded reachability probabilities and allow for huge reductions of the state space. The first technique has been introduced in [19] and will be referred to as *interval abstraction* in the following. The second one is based on [10] and is referred to as *MDP abstraction*.

As a preprocessing step for both techniques, the given concrete CTMC is transformed into a weakly bisimilar *uniform* CTMC. This can be done in linear time. For d-ary tree-structured QBDs, the uniformization rate λ can be chosen as $\max_{i\in\{1,\ldots,d\}} \left(r_i{\uparrow} + \sum_{j\in\{1,\ldots,d\}}(r_j{\downarrow}+r_{i,j})\right)$, then the self-loop probabilities need to be adapted accordingly. To explain the abstraction concepts, in the remainder of the section a finite-state example will be considered.

Interval abstraction. The main idea behind interval abstraction is to partition the state space and to keep track of the minimal and maximal probabilities for taking exactly one transition leading from a partition to a successor partition (possibly the same). In the abstract model, these minimal and maximal probabilities form the lower and upper bounds of transition probability intervals.

To exemplify how to obtain an abstraction of a concrete model, we consider the uniform CTMC depicted in Figure 3 (left). Given the partitioning $\mathcal{A}$ defined by abstraction function $\alpha : \mathcal{S} \to \mathcal{A}$ with $\alpha(s_i) = s$, $\alpha(u_i) = u$ and $\alpha(v_i) = v$ for all $i \in \{1,\ldots,5\}$, the abstract model is depicted in Figure 3 (right) with abstract states s, u, and v. To compute, say, the probability bounds for taking a transition from abstract state s to abstract state u, the minimal and maximal probabilities to take corresponding transitions in the concrete model have to be determined. The minimal probability of such a transition in the concrete model is $\frac{1}{5}$ for leaving s_4 towards u_2. The maximum is $\frac{4}{5}$ as for s_2 there are two ways to reach a state in u, the overall probability to do so is just the sum of the probabilities ($\frac{1}{5}+\frac{3}{5}$), and there is no other state in s for which there is a larger probability to reach states in u. This yields the transition probability interval $[\frac{1}{5},\frac{4}{5}]$ for the abstract transition from s to u. The intervals for the other abstract transitions are computed similarly.

The probability to start in an abstract state is just the sum of probabilities to start in the represented concrete states (the same applies to MDP abstraction).

Formally, an *abstract CTMC* (ACTMC) is a tuple $(\mathcal{S}, \mathbf{P}_l, \mathbf{P}_u, \lambda, \mu_0)$ where $\mathcal{S}$ is the set of states, $\mathbf{P}_l, \mathbf{P}_u : \mathcal{S}\times\mathcal{S} \to [0,1]$ are the lower and upper transition probability bound matrices such that for all $s \in \mathcal{S} : \mathbf{P}_l(s,\mathcal{S}) \leq 1 \leq \mathbf{P}_u(s,\mathcal{S})$, $\lambda \in \mathbb{R}_{>0}$ is the uniform exit rate and $\mu_0 \in \mathit{distr}(\mathcal{S})$ is the initial distribution. The set of transition probability distributions for ACTMC $\mathcal{M}$ is given by $\mathbf{T}_\mathcal{M} : \mathcal{S} \to 2^{\mathit{distr}(\mathcal{S})}$ where for all $s \in \mathcal{S}$,

$$\mathbf{T}_\mathcal{M}(s) = \big\{\mathbf{P}(s,\cdot) \in \mathit{distr}(\mathcal{S}) \mid \forall s' \in \mathcal{S} : \mathbf{P}_l(s,s') \leq \mathbf{P}(s,s') \leq \mathbf{P}_u(s,s')\big\}.$$

MDP abstraction. The idea behind MDP abstraction is to include *sets of distributions* for each state in the abstract model. Instead of keeping track of the extreme behavior in terms of minimal and maximal transition probabilities, for *each* concrete state that is represented by an abstract one, the transition probabilities have to be stored. Note that this technique is not applicable when infinitely many probability distributions have to be associated to an abstract state.

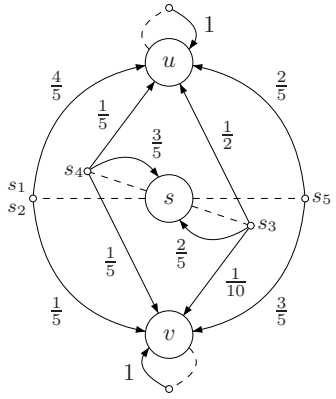


Fig. 4. An MDP abstraction.

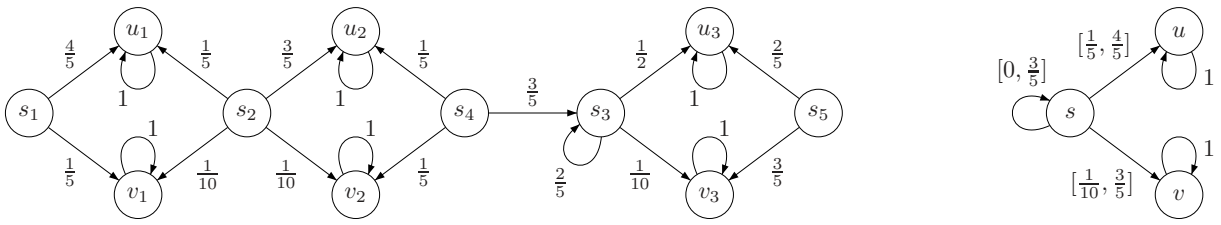


Fig. 3. A uniform CTMC (left) and an interval abstraction (right).

Applying MDP abstraction to the example CTMC from Figure 3 (left) yields the same state space as for the interval abstraction, however, the transition structure is quite different. The resulting abstract model is a uniform continuous-time Markov decision process [2] (CTMDP; see Figure 4, dashed arcs connect states to the associated distributions). For s_1 and s_2 we obtain the same probability distributions for taking a transition to the sets of states that are mapped to s, u and v by abstraction function α, therefore, they can be collapsed in the abstract model. For all other states in s, a distinct distribution has to be added to the abstract state s.

Formally, a uniform CTMDP is a tuple $(\mathcal{S}, A, \mathbf{P}, \lambda, \mu_0)$ with set of states $\mathcal{S}$, action set A, the three-dimensional probability matrix $\mathbf{P} : \mathcal{S} \times A \times \mathcal{S} \to [0,1]$ with $\mathbf{P}(s,a,\mathcal{S}) = 1$ for all $s \in \mathcal{S}, a \in A$, uniform exit rate $\lambda \in \mathbb{R}_{>0}$ and initial distribution $\mu_0 \in \mathit{distr}(\mathcal{S})$. The set of transition distributions for CTMDP $\mathcal{M}$ is given by $\mathbf{T}_{\mathcal{M}} : \mathcal{S} \to 2^{\mathit{distr}(\mathcal{S})}$ where for all $s \in \mathcal{S}$,

$$\mathbf{T}_{\mathcal{M}}(s) = \big\{\mathbf{P}(s,a,\cdot) \in \mathit{distr}(\mathcal{S}) \mid a \in A\big\}.$$

Note that ACTMCs and uniform CTMDPs are conservative extensions of uniform CTMCs, i.e., a uniform CTMC can be represented as an ACTMC with $\mathbf{P}_l = \mathbf{P}_u$ and as uniform CTMDP with $|A| = 1$.

Nondeterminism. Both abstract models have a nondeterministic component. In ACTMCs, the transition probabilities have to be chosen from the intervals in each step and in CTMDPs an action has to be chosen, i.e., a distribution. Depending on how these choices are made by the so-called scheduler (also-called strategy, policy, adversary) the system may behave differently. Also time-bounded reachability probabilities depend on the scheduler [2]. However, by computing the minimal and maximal reachability probabilities in the abstract model, one obtains firm lower and upper bounds for the value in the concrete model. If the partitioning of the state space has been chosen properly, enough information is preserved in the abstract model to guarantee the desired minimal/maximal reachability probabilities, also for the concrete model.

Comparison. First we intuitively explain the relation between both abstractions using the examples above. Then we formalize this relationship.

In principle, interval abstraction has more potential for reduction of the model's size and MDP abstraction preserves more information from the original model. This can be observed in the diagram in Figure 5 where the possible choices for transition probabilities to leave s towards u and v, respectively, are plotted for both abstractions. The choices of MDP abstraction are marked by the concrete states they represent (cf. Figure 4). For interval abstraction, all possible choices μ are given by the *intersection* of the rectangle (all choices for $\mu(u)$ and $\mu(v)$ out of $[\frac{1}{5}, \frac{4}{5}]$ and $[\frac{1}{10}, \frac{3}{5}]$) and the trapezoid shape (the probability mass left for distribution amongst $\mu(u)$ and $\mu(v)$ after choosing the self-loop probability $\mu(s)$ from $[0, \frac{3}{5}]$, i.e. $1 - [0, \frac{3}{5}] = [\frac{2}{5}, 1]$).

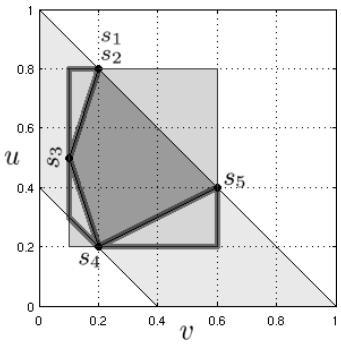


Fig. 5. Interval vs. MDP abstraction.

Removing the dark area (all the behavior under a randomized scheduler in MDP abstraction) from that intersection yields the three marked triangles that represent all the behavior in interval abstraction that is not present in MDP abstraction.

We formalize this observation using the concept of probabilistic simulation and give a variant of the definition in [19] that is compatible with ACTMCs and uniform CTMDPs. Intuitively, simulation relations are used to describe relations between concrete and abstract models. The main idea is that whenever some behavior occurs in the concrete model, it can be mimicked in the abstract model. Also different abstractions $\mathcal{M}$ and $\mathcal{M}'$ can be related in the sense that, if $\mathcal{M}$ is *simulated* by $\mathcal{M}'$, its abstract behavior can be mimicked by $\mathcal{M}'$.

Definition 2 (Simulation): Let $\mathcal{M}$, $\mathcal{M}'$ be abstract models with state spaces $\mathcal{S}$, $\mathcal{S}'$, and transition distributions $\mathbf{T}$, $\mathbf{T}'$. Relation $\mathcal{R} : \mathcal{S} \times \mathcal{S}'$ is a simulation on $\mathcal{S}$ and $\mathcal{S}'$, iff for all $s \in \mathcal{S}, s' \in \mathcal{S}'$, $s\mathcal{R}s'$ implies: For any $\mu \in \mathbf{T}(s)$ there exists $\mu' \in \mathbf{T}'(s')$ and $\Delta : \mathcal{S} \times \mathcal{S}' \to [0,1]$ such that for all $u \in \mathcal{S}$, $u' \in \mathcal{S}'$:

(a) $\Delta(u,u') > 0 \Rightarrow u\mathcal{R}u'$,
(b) $\Delta(u,\mathcal{S}') = \mu(u)$,
(c) $\Delta(\mathcal{S},u') = \mu'(u')$.

We write $s \preceq s'$ if $s\mathcal{R}s'$ for some simulation relation $\mathcal{R}$ and $\mathcal{M} \preceq \mathcal{M}'$ if the initial distribution μ of $\mathcal{M}$ is simulated by the initial distribution μ' of $\mathcal{M}'$, i.e. if there exists $\Delta : \mathcal{S} \times \mathcal{S}' \to [0,1]$ such that for all $u \in \mathcal{S}$, $u' \in \mathcal{S}'$ conditions (a) to (c) from Def. 2 hold. For CTMC $\mathcal{C}$ and partitioning $\mathcal{A}$ of its state space, we denote the interval abstraction (MDP abstraction resp.) of $\mathcal{C}$ induced by $\mathcal{A}$, by $\mathit{abstr}_{\mathit{Int}}(\mathcal{C},\mathcal{A})$ ($\mathit{abstr}_{\mathit{MDP}}(\mathcal{C},\mathcal{A})$ resp.).

Note that simulation preserves time-bounded reachability probabilities [20], more precisely, lower and upper bounds thereof.

Lemma 1: Let $\mathcal{C}$ be a uniform CTMC with state space $\mathcal{S}$ and let $\mathcal{A}$ be a partitioning of $\mathcal{S}$ such that there exists $\mathit{abstr}_{\mathit{MDP}}(\mathcal{C},\mathcal{A})$, then:

$$\mathit{abstr}_{\mathit{MDP}}(\mathcal{C},\mathcal{A}) \preceq \mathit{abstr}_{\mathit{Int}}(\mathcal{C},\mathcal{A}).$$

The above lemma suggests that as long as $\mathit{abstr}_{\mathit{MDP}}(\mathcal{C},\mathcal{A})$ does exist and is not significantly larger than $\mathit{abstr}_{\mathit{Int}}(\mathcal{C},\mathcal{A})$, MDP abstraction is to be favored since the abstract model is at least as accurate as in case of interval abstraction. Otherwise interval abstraction would be the first choice.

IV. Partitioning a Tree-Structured QBD

In order to apply abstraction to a tree-structured QBD, first a suitable partitioning of the state space has to be found. Recall that the state-space of the tree-structured QBD results from a PH service distribution with preemptive LIFO scheduling. Every state of the tree represents (i) the number of jobs in the queue, (ii) the service phases of the preempted jobs, (iii) the phase of the job that is currently in service and (iv) the precise order of jobs in the queue. The states with m jobs, that are situated in the same layer of the tree, have the form $\vec{x} = (x_1, x_2, \ldots, x_m)$ where x_i gives the service phase of the

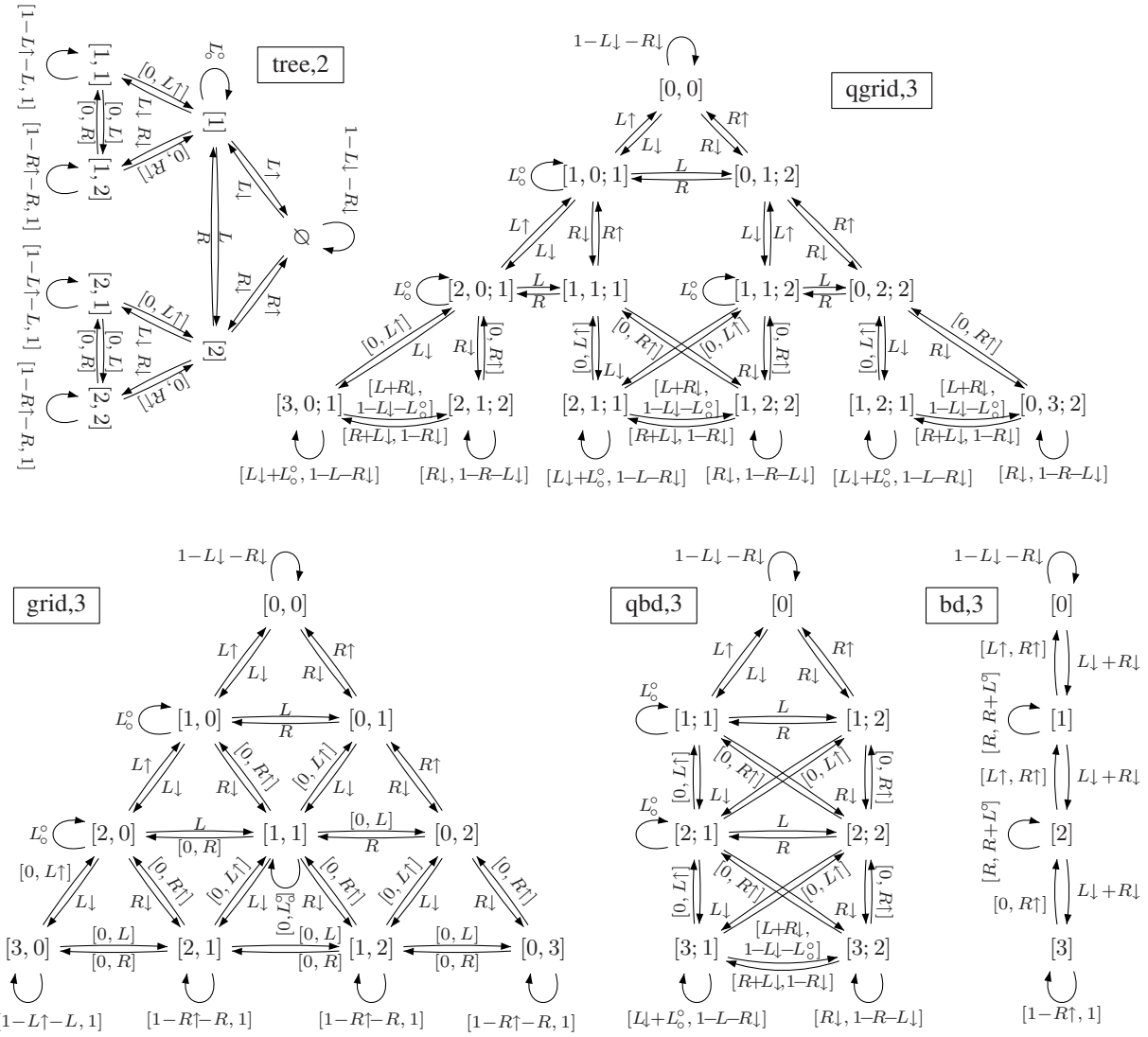


Fig. 6. Interval abstractions $\mathcal{M}_{tree,2}$, $\mathcal{M}_{qgrid,3}$, $\mathcal{M}_{grid,3}$, $\mathcal{M}_{qbd,3}$ and $\mathcal{M}_{bd,3}$ for $L\uparrow + L^\circ = R\uparrow$, $L + L_\circ = R$, and $L^\circ_\circ = L^\circ + L_\circ$.

i-th job in the queue. We abbreviate the prefix of $\vec{x}$ of length n by $\vec{x}{\downarrow_n}$ and the number of jobs in $\vec{x}$ in phase i by $\#_i\vec{x}$.

In the following, we present abstractions that preserve several of the above mentioned properties from (i) to (iv). In order to obtain a *finite* abstract state space, we also have to apply counter abstraction, i.e., we cut the state space at layer n (denoted cut level in the following), which implies that property (i) is only preserved for less than n customers.

Let $\mathcal{T}$ be a tree-structured QBD with state space $\mathcal{S}$. For partitioning scheme $ps \in \{tree, qgrid, grid, qbd, bd\}$ and cut level $n \in \mathbb{N}^+$, we define partitioning $\mathcal{A}_{ps,n}$ by abstraction function $\alpha_{ps,n} : \mathcal{S} \to \mathcal{A}_{ps,n}$. For $\vec{x} = (x_1, x_2, \ldots, x_m) \in \mathcal{S}$, let

$$\alpha_{tree,n}(\vec{x}) = \begin{cases} [\vec{x}] & \text{if } m < n, \\ [\vec{x}{\downarrow_n}] & \text{otherwise;} \end{cases}$$

$$\alpha_{qgrid,n}(\vec{x}) = \begin{cases} [\#_1\vec{x}, \ldots, \#_d\vec{x};\ x_m] & \text{if } m < n, \\ [\#_1\vec{x}{\downarrow_{n-1}} + \#_1(x_m), \ldots, \\ \ \#_d\vec{x}{\downarrow_{n-1}} + \#_d(x_m);\ x_m] & \text{otherwise;} \end{cases}$$

$$\alpha_{grid,n}(\vec{x}) = \begin{cases} [\#_1\vec{x}, \ldots, \#_d\vec{x}] & \text{if } m < n, \\ [\#_1\vec{x}{\downarrow_n}, \ldots, \#_d\vec{x}{\downarrow_n}] & \text{otherwise;} \end{cases}$$

$$\alpha_{qbd,n}(\vec{x}) = \begin{cases} [m;\ x_m] & \text{if } m < n, \\ [n;\ x_m] & \text{otherwise;} \end{cases}$$

$$\alpha_{bd,n}(\vec{x}) = \begin{cases} [m] & \text{if } m < n, \\ [n] & \text{otherwise.} \end{cases}$$

For example, when using the *grid* scheme, all states with the same number of jobs (up to the n-th queued job) in phases 1, 2, …, d, respectively, are grouped together.

Scheme *bd* preserves property (i) only, *grid* additionally preserves (ii), whereas *qbd* additionally preserves (iii). Scheme *tree* preserves all properties and *qgrid* preserves all but (iv). *Interval* abstractions of the binary tree-structured QBD from Figure 2 are shown in Figure 6. From those, it becomes clear that the partitioning schemes are named after the structure of the abstract models. Schemes *bd* and *qbd* yield chain-like structures similar to (quasi)-birth death

ps	tree	qgrid	grid	qbd	bd
$\lvert\mathcal{A}_{ps,n}\rvert$	$\mathcal{O}(d^n)$	$\mathcal{O}(d\cdot\binom{d+n}{d})$	$\mathcal{O}(\binom{d+n}{d})$	$\mathcal{O}(d\cdot n)$	$\mathcal{O}(n)$
#distrs	$< 1.5\times$	$< d\times$	$< d\times$	$< d\times$	$\leq d\times$

TABLE I
SIZES OF ABSTRACT MODELS AND AVERAGE NUMBERS OF DISTRIBUTIONS PER STATE (FOR $d > 1$)

processes, where the *qbd* scheme enhances the *bd* scheme by storing the phase of the job currently in service. Similarly the *qgrid* scheme enhances the *grid* scheme. For sufficiently large n, the size of the abstract models decreases in the order of the partitionings as presented in the first row in Table I. Positive results on the relationship of abstractions induced by the partitionings proposed above are given in the following lemma. Note that for *grid* and *qbd* abstraction, a formal relationship in terms of *probabilistic simulation* cannot be established as each abstraction preserves information that is not preserved by the other, (ii) and (iii) respectively.

Lemma 2: Let $\mathcal{T}$ be a tree-structured QBD, then for $x \in \{Int, MDP\}$ and $n \in \mathbb{N}^+$:

$$\begin{aligned}
&\mathit{abstr}_x(\mathcal{T}, \mathcal{A}_{tree,n}) \preceq \mathit{abstr}_x(\mathcal{T}, \mathcal{A}_{qgrid,n})\\
&\mathit{abstr}_x(\mathcal{T}, \mathcal{A}_{qgrid,n}) \preceq \mathit{abstr}_x(\mathcal{T}, \mathcal{A}_{grid,n}) \preceq \mathit{abstr}_x(\mathcal{T}, \mathcal{A}_{bd,n})\\
&\mathit{abstr}_x(\mathcal{T}, \mathcal{A}_{qgrid,n}) \preceq \mathit{abstr}_x(\mathcal{T}, \mathcal{A}_{qbd,n}) \preceq \mathit{abstr}_x(\mathcal{T}, \mathcal{A}_{bd,n})
\end{aligned}$$

V. RESULTS

We compute lower and upper bounds for the probability to reach a certain set of states within a given amount of time. We stress that our abstractions guarantee that these bounds are always safe, i.e., the results for the concrete model lie within the computed bounds. To get an impression of how the system evolves over time, we compute probability bounds for gradually increasing time bounds up to a point where the system approaches an equilibrium. The average number of distributions per state never exceeds d (cf. second row in Table I). Since we investigate phase-type service distributions of the order $d \leq 5$, with MDP abstraction, we obtain models that are not much larger than the ones obtained by interval abstraction. Due to Lemma 1, we therefore stick to MDP abstraction in the following. Abstract models are generated using a Java tool, and the analysis is done using MRMC version 1.4 [18]. All experiments were run on a standard desktop computer (Athlon X2 3800+, 2GB RAM).

Recall from Section II that we analyze the probability of an M|PH|1 queue with preemptive LIFO scheduling, with a given number of waiting jobs, to process all but k jobs within t time units. We carry out the following six experiments. We also provide the utilizations for each setting, i.e., the fraction of time the server is busy serving jobs.

E1. Firstly, we compare the *precision* of the abstractions induced by the partitionings presented in the previous section, check if the results are consistent with Lemma 2 and provide results for two sets of rates. We choose a low cut level for all partitioning schemes and $k = 0$. Note that $k = 0$ is a special case as the ordering of the jobs is of no relevance in the concrete model as all jobs need to be served anyway before k is reached. Hence, for all partitioning schemes preserving the number of customers per phase, as *tree*, *qgrid* and *grid*, the imprecisions have to result from the choice of the cut level.

E2. In the second experiment, we investigate the influence of the cut level on the imprecisions occuring for the special case $k = 0$. We focus on the *grid* scheme in this experiment.

E3. In a third experiment we compare the *precision* of the *grid*-abstractions for varying goal levels k and fixed cut level.

E4. We then present a refinement of the grid abstraction, where the tree is built in full breadth up to level c and the grid abstraction is applied for higher levels. We present results for varying c, fixed $k = 4$ and given cut level.

E5. In an additional experiment, we choose the cut levels such that the resulting abstract state spaces are of approximately the same size. Apart from the quality of the results, we compare the time it takes to generate and analyse the abstract models.

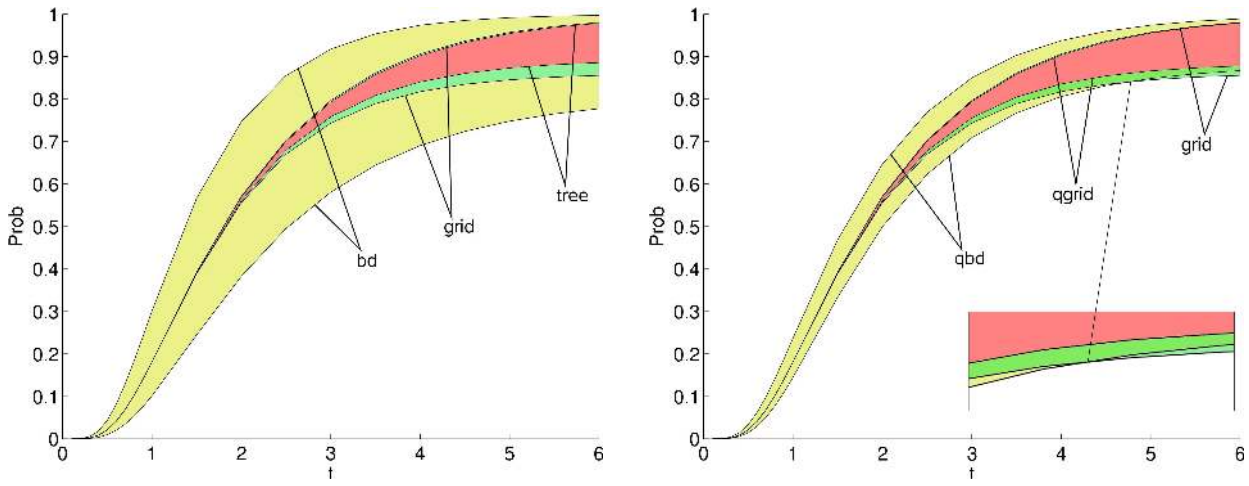


Fig. 7. E1: Upper and lower bounds for *bd*, *grid*, *tree* abstractions (left) and *qbd*, *qgrid*, *grid* abstractions (right) with the same cut level and $k = 0$.

E6. To emphasize the generality of our approach we present results for the refined grid abstraction, as presented in Experiment 4, for a phase-type 5 distribution.

As standard parameters we choose $L\downarrow = 2$, $R\downarrow = 3$, $L = 4$, $R = 5$, $L\uparrow = 7.5$, $R\uparrow = 10$, with a resulting utilization of $\rho = 57\%$.

E1. Partitioning schemes:

We compare three abstractions, $\mathcal{M}_{tree,12}$, $\mathcal{M}_{grid,12}$ and $\mathcal{M}_{bd,12}$, where we choose initial states that simulate $(1, 2, 1, 2, 1, 2, 1, 2)$ in the concrete model, namely $[1, 2, 1, 2, 1, 2, 1, 2]$, $[4, 4]$, and $[8]$ respectively, and take the empty state as goal state. The resulting lower and upper probability bounds, for these abstractions are shown in Figure 7 (left) where the x-axis shows the time bounds. As expected, the most accurate results are obtained using *tree* abstraction. However, the difference between the obtained upper and lower bound is rather large for medium and large time bounds, since the cut level 12 is too close to the initial state. The second best abstraction is *grid* abstraction, that is almost as good, especially for low time bounds. The *bd* abstraction, while having the least memory requirements, is clearly outperformed on the whole range of time bounds. Note that, for large time bounds, results are not as bad as for medium ones. This comes from the fact that upper bounds are derived by assuming that jobs are processed rather quickly while for lower bounds it is assumed that jobs are processed slowly. However, in both cases all jobs are processed eventually, due to a utilization which is less than one. The results we described above reflect the relation between the different abstractions as stated in Lemma 2.

The results for the other partitionings are presented in Figure 7 (right). The initial states $[4, 4; 2]$, $[4, 4]$, and $[8; 2]$ simulate $(1, 2, 1, 2, 1, 2, 1, 2)$. *Qbd* abstraction performs significantly better than *bd* abstraction (left), but still is outperformed by *grid* abstraction for low and medium time bounds. For large time bounds (starting at 95), *qbd* abstraction yields better lower bounds, even though the state space is significantly smaller than in case of *grid* abstraction. This comes from the fact that *qbd* abstraction *is aware* of the job that is currently processed. If that job is processed rather quickly, a transition to a higher level (away from the cut level) is more probable. This shows that *grid* and *qbd* abstractions are incomparable (cf. Lemma 2). Finally, combining the advantages of *grid* and *qbd* yields the inner pair of curves (*qgrid*). This scheme outperforms the other two abstractions and comes close to the *tree* scheme's results (left), however, with a drastically lower memory consumption (with a factor of 217).

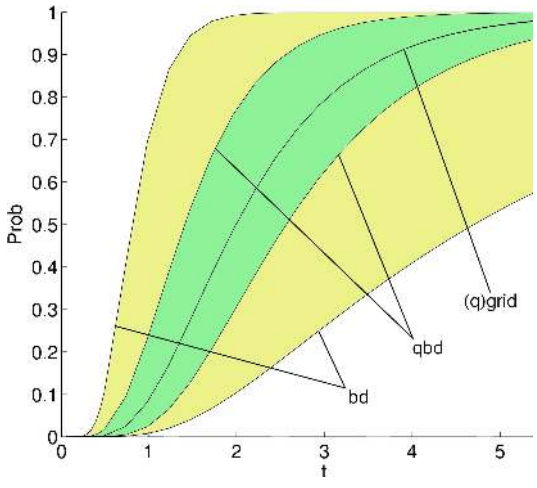


Fig. 8. E1 with an alternative parameter set: Upper and lower bounds for *bd*, *qbd* and *(q)grid* abstractions with the same cut level and $k = 0$.

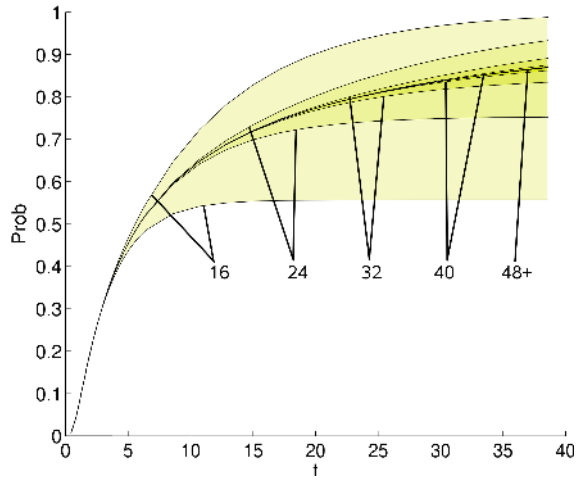


Fig. 9. E2: Upper and lower bounds for *grid* abstractions with increasing cut levels and $k = 0$.

Now, we consider an alternative set of parameters where rates are much more diverse: $L\downarrow = 1$, $R\downarrow = 5$, $L = 3$, $R = 15$, $L\uparrow = 8$, $R\uparrow = 20$, with a resulting utilisation of $\rho = 46\%$. As Figure 8 indicates, both *bd* and *qbd* abstractions perform badly with these parameters. The reason is that only the total number of jobs is stored and therefore in the best (worst) case, it is assumed that only short (long) jobs are present in the queue. On the contrary, *grid* and *qgrid* abstraction perform excellent (lower and upper bounds are overlapping in the graph).

E2. Influence of the cut level:

In our second experiment, we investigate the influence of the cut level on the quality of the results. We show upper and lower bounds on the probability to reach the empty state from initial state $[4, 4]$ with *grid* abstraction and increasing cut level. The parameters are chosen as in experiment 1, except $L\downarrow = \frac{3}{10}$ and $R\downarrow = 5$ are larger, resulting in a very high utilization of $\rho = 96\%$ and, hence, a much larger range on the time-axis.

Figure 9 shows that *grid* abstraction is capable of providing lower and upper bounds that differ marginally, for large enough cut levels. As all the jobs need to be served to reach the empty state, the ordering of the jobs does not matter. Hence, for cut level $n \to \infty$, *grid* abstraction contains all the necessary information, if the empty state is chosen as goal state.

E3. Influence of the goal level:

We compare results obtained with *grid* abstraction for a large enough cut level for increasing goal levels k. Figure 10 shows that with growing goal levels the difference between upper and lower probability bounds increases. This is due to

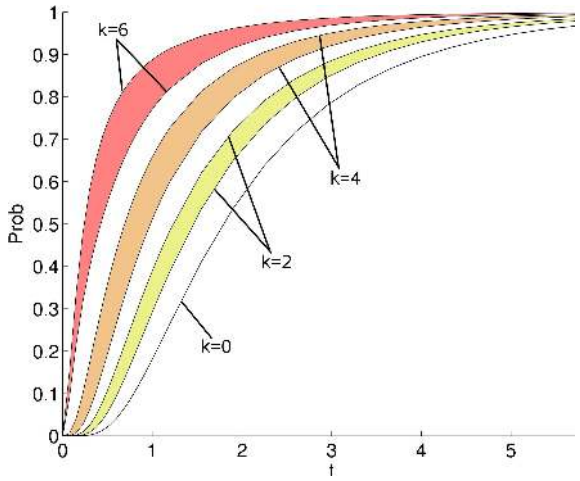


Fig. 10. E3: Upper and lower bounds for *grid* abstraction with increasing goal levels k and fixed cut level.

the fact, that in this setting *grid* abstraction does not contain all necessary information, as the ordering of the jobs in the queue influences the time that is needed to serve all but k jobs.

E4. Refinement:

To obtain more precise results, the *grid* partitioning scheme is *refined*. Lack of information on the first jobs in the queue leads to less tight bounds. Hence, the abstract states are refined according to the ordering of the first jobs. We define $\alpha_{grid,c,n}$ such that states are merged for which the order of the first c job phases is the same and where the numbers of job phases for jobs $c+1$ up to n are equal as well:

$$\alpha_{grid,c,n}(\vec{x}) = \begin{cases} [\vec{x}\downarrow_c; \#_1(x_{c+1},\ldots,x_m), & \\ \ldots, \#_d(x_{c+1},\ldots,x_m)] & \text{if } m < n, \\ [\vec{x}\downarrow_c; \#_1(x_{c+1},\ldots,x_n), & \\ \ldots, \#_d(x_{c+1},\ldots,x_n)] & \text{otherwise.} \end{cases}$$

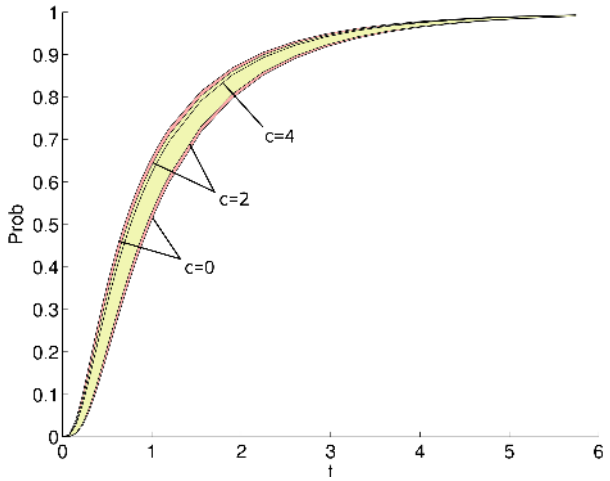
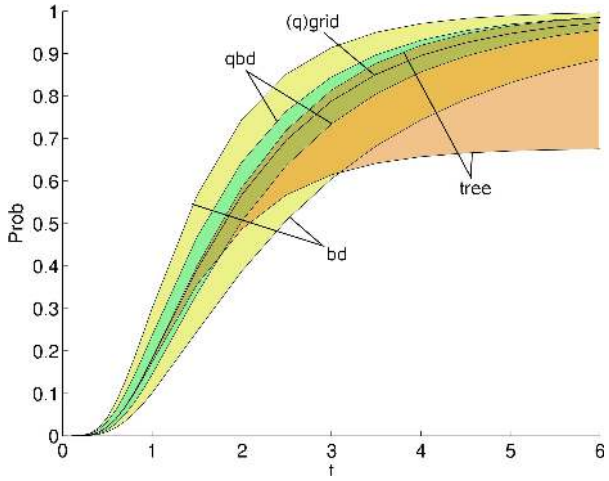


Fig. 11. E4: Upper and lower bounds for *refined grid* abstraction with increasing c for $k = 4$ and fixed cut level.



	depth	states	time/ms
tree	10	2047	989
qgrid	45	2071	1546
grid	63	2080	1378
qbd	1023	2047	1434
bd	2046	2047	1092

Fig. 12. E5: Upper and lower bounds for *bd*, *tree*, *qbd* and *(q)grid* abstractions with similar sized state spaces for $k = 0$.

This still is a very natural partitioning scheme. For example, for a binary tree-structured QBD, $\alpha_{grid,2,8}((1)) = [(1); 0, 0]$ and $\alpha_{grid,2,8}((1,2,1,1,2,1)) = [(1,2); 3, 1]$.

In Figure 11, results for goal level $k = 4$ and refinement levels $c \in \{0, 2, 4\}$ are shown. If the refinement level c equals the goal level k, and if the cut level is large enough, the obtained lower and upper bounds are extremely tight.

E5. Similar sized state spaces:

While in the first experiment the cut level remained constant, we now compare the different abstractions, while keeping the size of the abstract state space approximately the same. We select the same initial states and the same goal state as in the first experiment, as well as the first set of rates. The goal of this experiment is to determine which properties are more important for the quality of the results, given a maximal number of states.

The cut level, the number of states in the abstract model and the time for generating the abstract state space and for computing probability bounds are given in the table in Figure 12. While *tree* and *bd* abstraction use the least computation time, their results, shown in Figure 12, are rather imprecise compared to the other abstractions. Long-term behavior is not captured very well by *tree* abstraction as the cut level is close to the initial state. For small time bounds, *bd* abstraction is the worst. Yet, it catches up with increasing time bounds, as it has the largest depth and therefore very little probability mass is *lost* in the cut level. However, neither one can compete with *grid* and *qgrid* abstractions for which the lower and upper bounds are almost identical. Here, the *grid* scheme is favorable as the computation time is 12% smaller than for *qgrid*.

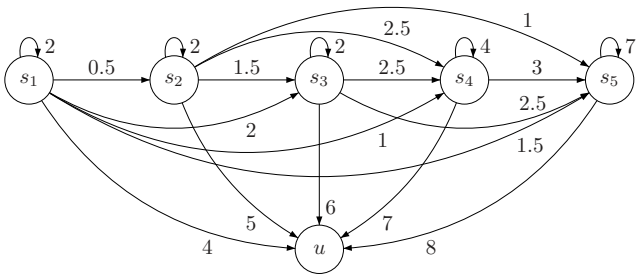


Fig. 13. Phase-type 5 service distribution

E6. Phase-type 5 service distribution:

To emphasize the generality of our approach, we present results on a $\mathrm{M}|\mathrm{PH}_5|1$ queue with preemptive LIFO and a utilization of $\rho = 77\%$. The phase-type distribution is depicted in Figure 13 and the parameters are shown below.

$$r{\downarrow} = \begin{pmatrix} 0.5 & 1 & 0.75 & 1.25 & 1.5 \end{pmatrix}$$

$$r{\uparrow} = \begin{pmatrix} 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

$$r = \begin{pmatrix} 2 & 0.5 & 2 & 1 & 1.5 \\ 0 & 2 & 1.5 & 2.5 & 1 \\ 0 & 0 & 2 & 2.5 & 2.5 \\ 0 & 0 & 0 & 4 & 3 \\ 0 & 0 & 0 & 0 & 7 \end{pmatrix}$$

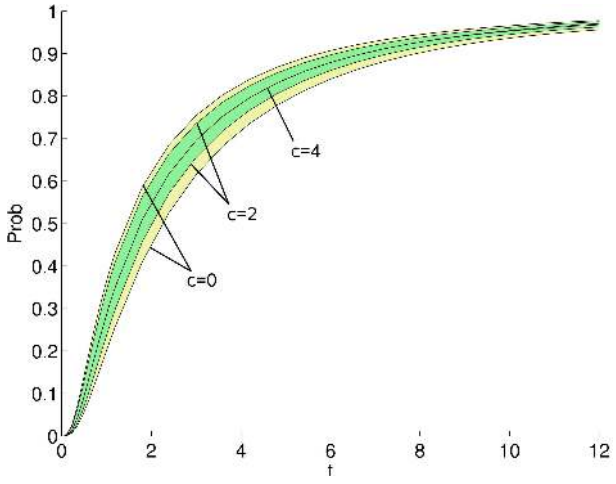


Fig. 14. E6: Upper and lower bounds for *refined grid* abstractions with increasing c for $k = 4$ and fixed cut level.

For initial state $(2, 2, 2, 1, 1)$, goal level $k = 4$ and cut level 24, Figure 14 shows upper and lower bounds for the *refined grid* abstraction with $c = 0, 2, 4$. Again, for matching goal and refinement level, the results differ only marginally.

In the following, we investigate how the time for abstraction and for the computation of probability bounds scales with the cut level. The goal level is fixed to $k = 0$.

The results in Figure 15 and in Table II show that for large enough cut level, the *grid* abstraction is capable of providing upper and lower bounds that differ only marginally and scale well with the size of the abstract model.

Note that the differences between upper and lower bounds of reachability probabilities listed in Table II cannot get smaller than $\varepsilon = 10^{-6}$ because MRMC computes ε-approximations of the probability bounds. In contrast, when using the uniformization method (cf. Section II), a massive number of states would have to be considered (see Table II, right part). Even with a terabyte of available memory, CTMCs with more than 10^{14} states cannot be dealt with. This shows evidently that abstraction is an attractive approach to computing time-bounded reachability probabilities in tree-structured QBDs.

VI. Conclusions

In this paper we have shown that abstraction techniques allow us to quickly compute highly-accurate time-bounded reachability probabilities for a class of infinite-state CTMCs, namely tree-structured QBDs. In queueing theory this model

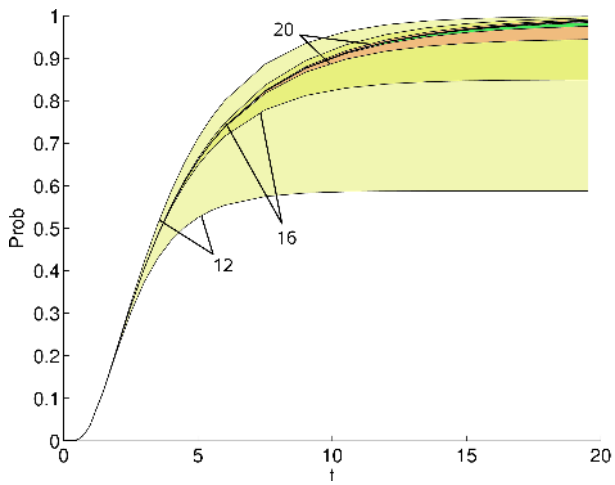


Fig. 15. E6: Upper and lower bounds for *grid* abstractions with increasing cut level and $k = 0$.

		grid abstraction								uniformization	
	diff	grid 12	grid 16	grid 20	grid 24	grid 28	grid 32	grid 36	grid 40	trunc	$\approx$ states
	2.5	0.0224	0.001	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	185	10^{129}
t	7.5	0.3117	0.0580	0.0062	0.0004	10^{-5}	10^{-6}	10^{-6}	10^{-6}	270	10^{188}
	15	0.4054	0.1345	0.0376	0.0086	0.0015	0.0002	$2 \cdot 10^{-5}$	$3 \cdot 10^{-6}$	398	10^{278}
	states	6188	20349	53130	118755	237336	435894	749398	1221759		
	distributions	28666	96901	256796	579151	1164206	2146761	3701296	6047091		
	time (h:m:s)	0:00:26	0:01:33	0:04:15	0:09:50	0:20:14	0:38:13	1:07:57	2:06:04		

TABLE II
E5: Differences between lower and upper bounds in *grid* abstractions vs. size of the state space for uniformization with error bound $\varepsilon = 10^{-6}$.

class is widely used, however, mostly for steady-state analysis. Computing time-bounded reachability probabilities for tree-structured QBDs has not been possible before. We present non-trivial partitionings for the infinite state-space of tree-structured QBDs that preserve different properties and discuss how they formally relate. In six experiments we thoroughly compare the efficiency of these partitionings, regarding the size of the resulting state space and the difference between upper and lower probability bounds, resulting from abstraction. Grid-like schemes that neglect the order of the jobs in the queue, have shown to perform best when the empty state is chosen as goal state. When the cut level is chosen according to the utilization and the choice of the initial and the goal state, we achieve differences between the upper and the lower probability bound of only 10^{-6}. To reach the empty state, all jobs need to be processed, regardless of their ordering, hence *grid* abstraction contains all necessary information and hence produces very tight bounds. In case goal states belong to higher levels $k > 0$, we present a *refinement* of the *grid* abstraction, where the state-space up to level k is not abstracted and *grid* abstraction is used for all higher levels. This produce extremely tight bounds for this setting as well.

Even though the tree-structured QBDs in this paper result from M|PH|1 queues, our approach can also be used for the more general PH|PH|1 queues. Future work will include the automation of the parameterization, as well as finding partitionings for other highly-structured, fast growing state spaces, such as for multi-class queueing networks.

REFERENCES

[1] Abdulla, P., Jonsson, B., Nilsson, M., Saksena, M.: A survey of regular model checking. In: *CONCUR*. *LNCS*, 3170. (2004) 35–48

[2] Baier, C., Hermanns, H., Katoen, J. P., Haverkort, B. R.: Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *TCS* **345** (2005) 2–26

[3] Baier, C., Haverkort, B., Hermanns, H., Katoen, J.-P.: Model-checking algorithms for continuous-time Markov chains. *IEEE TSE* **29** (2003) 524–541

[4] Baier, C., Katoen, J.-P., Hermanns, H., Wolf, V.: Comparative branching-time semantics for Markov chains. *Information and Computation* **200** (2005) 149–214

[5] Bini, D., Latouche, G., Meini, B.: Solving nonlinear matrix equations arising in tree-like stochastic processes. *Lin. Alg. and its Applications* **366** (2003) 39–64

[6] Brázdil, T., Kucera, A., Strazovský, O.: On the decidability of temporal properties of probabilistic pushdown automata. In: *STACS*. *LNCS*, 3404. (2005) 145–157

[7] Ciardo, G.: Discrete-time Markovian stochastic Petri nets. In: *Computations with Markov Chains*. Raleigh (1995) 339–358

[8] Ciardo, G., III, R. L. J., Miner, A., Siminiceanu, R.: Logic and stochastic modeling with SMART. In: *Computer Performance Evaluation*. *LNCS*, 2794. (2003) 78–97

[9] D'Aprile, D., Donatelli, S., Sproston, J.: CSL model checking for the GreatSPN tool. In: *Computer and Information Sc., ISCIS*. *LNCS*, 3280. (2004) 543–553

[10] D'Argenio, P. R., Jeannet, B., Jensen, H. E., Larsen, K. G.: Reachability analysis of probabilistic systems by successive refinements. In: *PAPM–PROBMIV*. *LNCS*, 2165. (2001) 39–56

[11] de Alfaro, L., Roy, P.: Magnifying-lens abstraction for Markov decision processes. In: *CAV*. *LNCS*, 4590. (2007) 325–338

[12] Etessami, K., Wojtczak, D., Yannakakis, M.: Quasi-birth-death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems. In: *QEST*. IEEE CS Press (2008) 243–253

[13] Etessami, K., Yannakakis, M.: Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. In: *STACS*. *LNCS*, 3404. (2005) 340–352

[14] Gilmore, S., Hillston, J.: The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In: *Computer Performance Evaluation*. *LNCS*, 794. (1994) 353–368

[15] Grassmann, W.: Finding transient solutions in Markovian event systems through randomization. In: *Numerical Solution of Markov Chains*. Dekker (1991) 411–420

[16] Gross, D., Miller, D.: The randomization technique as a modeling tool and solution procedure for transient Markov chains. *Operations Research* **32** (1984) 343–361

[17] He, Q., Alfa, A.: The discrete time MMAP[K]/PH[K]/1/LCFS-GPR queue and its variants. In: *3rd Int. Conf. on Matrix Analytic Methods*. (2000) 167–190

[18] Katoen, J.-P., Khattri, M., Zapreev, I. S.: A Markov reward model checker. In: *QEST*. IEEE CS Press (2005) 243–244

[19] Katoen, J.-P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for continuous-time Markov chains. In: *CAV*. *LNCS*, 4590. (2007) 316–329

[20] Katoen, J.-P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for continuous-time Markov chains. In: *CAV*. *LNCS*, 4590. Springer (2007) 316–329

[21] Kwiatkowska, M., Norman, G., Parker, D.: Game-based abstraction for Markov decision processes. In: *QEST*. IEEE CS Press (2006) 157–166

[22] Latouche, G., Ramaswami, V.: *Introduction to Matrix Analytic Methods in Stochastic Modeling*. SIAM (1999)

[23] Remke, A., Haverkort, B.: A uniformization-based algorithm for model checking the CSL until operator on labeled queueing networks. In: *FORMATS*. *LNCS*, 4763. (2008) 336–351

[24] Remke, A., Haverkort, B., Cloth, L.: CSL model checking algorithms for QBDs. *Theoretical Computer Science* (2007)

[25] Schnoebelen, P.: The verification of probabilistic lossy channel systems. In: *Validation of Stochastic Systems*. *LNCS*, 2925. (2004) 445–465

[26] Spaey, K., van Houdt, B., Blondia, C.: On the generality of binary tree-like Markov chains. In: *Markov Aniversary Meeting*. Boston Books (2006) 79–88

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

[28] van Houdt, B., Blondia, C.: Throughput of Q-ary splitting algorithms for contention resolution in communication networks. *Commun. Inform. Sys.* **4** (2005) 135–164

[29] van Houdt, B., Blondia, C.: Analyzing priority queues with 3 classes using tree-like processes. *Queueing Systems* **54** (2006) 99–109

[30] van Velthoven, J., van Houdt, B., Blondia, C.: Transient analysis of tree-like processes and its application to random access systems. *SIGMETRICS/Performance* **34** (2006) 181–190

[31] Yeung, R., Alfa, A.: The Quasi-Birth-Death type Markov chain with a tree structure. *Stochastic Models* **15** (1999) 639–659

[32] Zhang, L., Hermanns, H., Moritz Hahn, E., Wachter, B.: Time-bounded model checking of infinite-state continuous-time Markov chains. In: *ACSD*. IEEE CS Press (2008) 98–107