Fundamenta Informaticae XXI (2001) 1001–1027
IOS Press

Time-Bounded Model Checking of Infinite-State Continuous-Time Markov Chains*

E. Moritz Hahn

Holger Hermanns

Björn Wachter

Lijun Zhang

Department of Computer Science, Saarland University, Saarbrücken, Germany

Abstract. The design of complex concurrent systems often involves intricate performance and dependability considerations. Continuous-time Markov chains (CTMCs) are a widely used modeling formalism that captures such performance and dependability properties, and makes them analyzable by model checking. In this paper, we focus on time-bounded probabilistic properties of infinite-state CTMCs, expressible in a subset of continuous stochastic logic (CSL). This comprises important dependability measures, such as time-bounded probabilistic reachability, performability, survivability, and various availability measures like instantaneous, conditional instantaneous and interval availabilities. Conventional model checkers explore the given model exhaustively, which is often costly, due to state explosion, and sometimes impossible because the model is infinite. This paper presents a method that only explores the model up to a finite depth. The required depth is determined on the fly by an algorithm that is configurable in order to adapt to the characteristics of different classes of models. We provide experimental evidence showing that our method is effective.

Keywords: Markov chains, model checking, truncation, uniformization

Address for correspondence: Lijun Zhang, zhang@cs.uni-sb.de
*This work is supported by the NWO-DFG bilateral project VOSS, by the DFG as part of the Transregional Collaborative Research Center SFB/TR 14 AVACS and the Graduiertenkolleg "Leistungsgarantien für Rechnersysteme", and has received funding from the European Community's Seventh Framework Programme under grant agreement n^o 214755.

1. Introduction

Continuous-time Markov chains (CTMCs) [29] and their extensions with rewards are popular in performance and dependability analysis, and also in biological modeling. In the context of CTMCs, properties of interest can be specified using continuous stochastic logic (CSL) [4, 6], which is a branching-time temporal logic inspired by CTL [10].

We consider the model checking problem for CSL over *infinite* CTMCs. For highly structured infinite CTMCs, this has already been studied in [28, 27]. In contrast, we consider methods which support arbitrary infinite CTMCs, including CTMCs with unbounded rates, like biological/chemical models with unbounded quantities of substances or queuing systems with unbounded queues. We focus on the CSL formulas without the steady-state operator and unbounded until [6]. The resulting logic can express (nested) time-bounded probabilistic properties such as: *"the probability to reach Ψ-states along Φ-states within time interval [6.5,8.5] is smaller than* 0.1*"* via $\mathcal{P}_{<0.1}(\Phi\,\mathcal{U}^{[6.5,8.5]}\,\Psi)$. For CTMCs, these properties constitute the arguably most important class of CSL formulas. They can express many performance measures, including time-bounded probabilistic reachability, various availability measures like instantaneous availabilities, conditional instantaneous availabilities and interval availabilities [6]. We present model checking algorithms for such properties over infinite CTMCs.

For *finite* CTMCs, Aziz *et al.* [4] proved that *exact* CSL model checking is decidable using results from number theory, however, in practice, the approximate algorithm for CSL model checking presented in [21, 6] is used, which is based on a numerical analysis of the transient time-dependent probability vector. This numerical *transient analysis* is usually carried out using the *uniformization* technique [29] where the transient probabilities are expressed by a weighted infinite sum (of transient probabilities computed in a finite discrete-time Markov chain). The weights are given by a Poisson-distributed jump process. The algorithm *truncates* the infinite sum using the Fox-Glynn algorithm [11] and thus yields an approximation with a pre-specified accuracy.

The CSL model checking algorithm of [21, 6] requires constructing the full matrix of the CTMC, and is thus limited to finite CTMCs. On the other hand, transient-analysis techniques [12, 32, 24, 25, 26] have been developed that truncate this matrix to finite size by only considering the state space of the CTMC up to a certain depth. This admits transient analysis of large or even infinite-state CTMCs, given implicitly in some modeling language. The approximation error can be made arbitrarily small by choosing a sufficiently large but finite depth bound. A sufficient depth can be computed by incremental depth-first exploration of the CTMC and on-the-fly error estimation. Once the depth has been computed, transient analysis is carried out on the truncated model. *Truncation* can lead to a dramatic reduction in model size. In this context, finite state projection was successfully applied to transient analysis of CTMCs in the biological domain [24, 25, 26].

While truncation has been applied to transient analysis, the technical focus of this paper is to enhance the novel CSL model checking algorithm introduced in [35], which, to our knowledge, is the first CSL model checking algorithm that employs truncation. The algorithm enables the automatic analysis of infinite (or very large) CTMCs, and, unlike [28, 27], applies to arbitrarily structured (finite or infinite) CTMC models. Beyond probabilistic properties, our technique also generalizes to time-bounded reward properties [5, 22]. The algorithm proceeds in two phases: first, a finite truncation depth is computed which is sufficient to check the given property up to a given accuracy, and then the resulting truncated model is submitted to CSL model checking for finite CTMCs [21, 6]. Computing a sufficient truncation depth for CSL model checking is more challenging than for transient analysis, since the required depth depends not only on the desired precision, and characteristics of the model: one also has to handle formulas involving until operators and nested sub-formulas. Nevertheless, results on transient analysis form an important building block of our present work.

Our algorithm to compute truncation depths for CSL formulas requires computing truncation depths for transient analysis. To this end, we build on and extend previous work on truncation-based transient analysis. Originally, we employed a truncation technique [35] for transient analysis based on dynamic uniformization by Grassmann [12]. While these truncation-based transient analyses and the ones developed in [31] and [24] follow the same general idea, they differ significantly in terms of the employed error estimation method and can thus yield significantly different truncation depths. For example, when applied to a protein synthesis model, studied in [35], finite state projection terminates at a shorter truncation depth and much faster than our method [35]. However, the more precise error estimation employed by finite state projection comes at a relatively high computational cost, which can outweigh the savings from a shorter truncation on many models. From this finding, we gained the insight that, in order to obtain a general-purpose analysis framework, the error estimation method should be configurable. In this paper, we extend our truncation-based analysis framework [35] in this way. This gives us the opportunity to experiment with different methods, and to explore the cost-precision trade-off of error estimation.

We focus on three complementary error estimation methods and prove that they are strictly ordered in terms of precision. The most costly and most precise one is employed by finite state projection [24, 25, 26] which performs transient analyses on the truncated model. The middle ground is a novel method that performs transient analysis on a chain, approximating the truncated model. It is a generalization of dynamic uniformization [12] which we had employed in [35]. Dynamic uniformization roughly amounts to using a more uniform chain, and, therefore, is very fast and space-efficient. However, being the least precise of the three, it may yield too high error estimates for a certain class of CTMCs. Therefore, although it is often effective in practice, it does not guarantee termination of the entire truncation algorithm unlike the other two methods.

We have implemented the truncation-based CSL model checker `INFAMY` [14]. Compared to the proof-of-concept implementation of [35], `INFAMY` features several error estimation methods and a much faster model exploration. The latter is inspired by the `SPIN` model checker [17]: `INFAMY` produces an executable for each analysis rather than "interpreting" the model. We have assessed the effectiveness of truncation and compared the error-estimation methods on a number of infinite-state CTMCs, including a protein synthesis model, a Jackson queuing network, and a job processing system. Further, we have also considered finite-state CTMCs with very large state spaces ($> 10^8$ states). In the infinite-state case, truncation proves to be a practical approach to verify models not amenable to finite-state methods, while, in the finite-state case, significant speedups can be achieved over model checking without truncation.

Contributions. We develop truncation-based CSL model checking. Our framework is an extension of [35] which is, to our knowledge, the first approach that leverages truncation-based transient analysis to model check CSL (without the steady-state operator and unbounded until). We extend this framework by making it configurable with respect to the error estimation method, and develop improved chain-based error estimation methods inspired by dynamic uniformization [12] and, adopt techniques from finite state projection [24]. We formally relate the different estimation techniques in terms of precision and prove that our novel chain-based techniques are sound. We have implemented the model checker `INFAMY` and conducted experiments on various models, including biological and queuing models, to validate the effectiveness of the approach and evaluate error estimation methods.

Outline. We give background on CTMCs and CSL in Section 2, and recall the established CSL model checking algorithm for finite CTMCs. In Section 3, we give the main contribution of the paper, time-bounded CSL model checking for infinite CTMCs, and also discuss an extension with rewards. Section 4 reports experimental results. Section 5 gives related work and Section 6 concludes. Some proofs have been moved to the appendix.

2. Preliminaries

We first briefly recall some basics of continuous-time Markov chains (CTMCs) and continuous stochastic logic (CSL). In Subsection 2.2, this background will be needed to explain the model checking algorithm for CSL over finite CTMCs which our technique uses on the finite truncation.

Let AP denote a set of atomic propositions. For a countable set S, a distribution μ over S is a function $\mu : S \to [0,1]$ satisfying the condition $\mu(S) \in \{0,1\}$, where $\mu(A)$ denotes the sum $\sum_{s \in A} \mu(s)$ for subset $A \subseteq S$. The *support* of a distribution μ is the set of states on which μ assumes a non-zero value, i.e., $Supp(\mu) = \{s \in S \mid \mu(s) > 0\}$. Let $Dist(S)$ denote the set of distributions over the set S.

Definition 2.1. A labeled discrete-time Markov chain (DTMC) is a tuple $\mathcal{D} = (S, \mathbf{P}, L)$ where S is a countable set of states, $\mathbf{P} : S \times S \to [0,1]$ is a probability matrix such that $\mathbf{P}(s, \cdot) \in Dist(S)$ for all $s \in S$, and $L : S \to 2^{AP}$ is a labeling function.

A state $s \in S$ is called absorbing if $\mathbf{P}(s, S) = 0$. Intuitively, $\mathbf{P}(s, s')$ denotes the probability of moving from s to s' in a single step. For $s \in S$, let $post(s) = \{s' \mid \mathbf{P}(s, s') > 0\}$ be the set of successor states of s. Observe that $post(s) = \emptyset$ iff s is absorbing. A state s' is reachable from s if there exists a sequence of states $s_1, \ldots, s_n$ with $n \geq 1$, $s_1 = s$, $s_n = s'$, and for each $i = 1, \ldots, n-1$, $\mathbf{P}(s_i, s_{i+1}) > 0$.

Definition 2.2. A labeled continuous-time Markov chain (CTMC) is a tuple $\mathcal{C} = (S, \mathbf{R}, L)$ where S, L are defined as for DTMCs, and $\mathbf{R} : (S \times S) \to \mathbb{R}_{\geq 0}$ is the *rate* matrix.

We say that the rate matrix is *rate-bounded* if the supremum $\sup_{s \in S} \mathbf{R}(s, S)$ is finite, otherwise, it is called rate-unbounded. The labeling function L assigns each state s a set of atomic propositions $L(s) \subseteq AP$ which are valid in s. If $\mathbf{R}(s, s') > 0$, we say that there is a transition from s to s'. For $s \in S$, let $post(s) = \{s' \mid \mathbf{R}(s, s') > 0\}$ denote the set of successor states of s. A state s is called *absorbing* if $post(s) = \emptyset$. A CTMC is finitely branching if, for each state s, the set of successors $post(s)$ is finite. In this paper, we consider rate-unbounded, finitely-branching CTMCs which do not explode [3, 30]. Roughly speaking, if the CTMC explodes, then there is a positive, non-zero probability of infinitely many jumps in finite time. On the contrary, the fact that a CTMC does not explode implies that in finite time with probability one only a finite number of states can be reached.

The transition probabilities in a CTMC are exponentially distributed over time. If $s' \in post(s)$ is the only transition starting from s, the probability that this transition will be triggered within time t is $1 - e^{-\mathbf{R}(s,s')t}$. Furthermore, if $s' \in post(s)$ for more than one state s', there is a race condition between the transitions starting from s. In this case, the probability that an arbitrary transition can be triggered within time t is given by $1 - e^{-\mathbf{R}(s,S)t}$, where $\mathbf{R}(s, A) := \sum_{s' \in A} \mathbf{R}(s, s')$ for $A \subseteq S$. The probability of taking a particular transition to s' from s within time t is $\frac{\mathbf{R}(s,s')}{\mathbf{R}(s,S)} \left(1 - e^{-\mathbf{R}(s,S)t}\right)$.

For a CTMC $\mathcal{C}$, a *uniformization rate* $q \in \mathbb{R}_{\geq 0}$ is any value satisfying $q \geq \sup_{s \in S} \mathbf{R}(s, S)$. If we observe the state distribution of $\mathcal{C}$ at time points given by a Poisson process with rate parameter q, we obtain a sequence of state observations, and these observations can be considered as a DTMC. This (so-called) uniformized DTMC can be obtained from $\mathcal{C}$ as follows:

Definition 2.3. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC. The uniformized DTMC of $\mathcal{C}$ with respect to the uniformization rate q is $uni(\mathcal{C}) = (S, \mathbf{P}, L)$ where $\mathbf{P}(s, s') = \frac{\mathbf{R}(s,s')}{q}$ if $s \neq s'$ and $\mathbf{P}(s, s') = 1 - \frac{\mathbf{R}(s,S \setminus \{s\})}{q}$ otherwise.

A CTMC $\mathcal{C} = (S, \mathbf{R}, L)$ is usually equipped with an initial state s_{init} or, more generally, an initial distribution μ.

Paths and probabilistic measure. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC. A (timed) infinite path is an infinite sequence $\sigma = s_1 t_1 s_2 t_2 \ldots$ satisfying $\mathbf{R}(s_i, s_{i+1}) > 0$, and $t_i \in \mathbb{R}_{\geq 0}$ for all $i = 1, 2, \ldots$. For the path σ and $i \in \mathbb{N}$, let $\sigma[i] = s_i$ denote the $(i+1)$-th state, and $\delta(\sigma, i) = t_i$ denote the time spent in s_i. For $t \in \mathbb{R}_{\geq 0}$, let $\sigma@t$ denote $\sigma[i]$ such that i is the smallest index with $t \leq \sum_{j=0}^{i} t_j$. For $\mathcal{C}$, let $Path_\infty^{\mathcal{C}}$ denote the set of all paths, and $Path_\infty^{\mathcal{C}}(s)$ denote the set of all paths starting from s. For state $s \in S$, a probability measure, denoted by $Pr_s^{\mathcal{C}}$, on the set $Path_\infty^{\mathcal{C}}(s)$ can be defined. A finite path is a finite sequence $\sigma = s_1 t_1 s_2 t_2 \ldots s_k$ for $k > 0$ satisfying $\mathbf{R}(s_i, s_{i+1}) > 0$ and $t_i \in \mathbb{R}_{\geq 0}$ for $i = 1, 2, \ldots k-1$. Let $len(\sigma) = k - 1$ denote the length of the path, $first(\sigma) = s_1$ denote the first state, and $last(\sigma) = s_k$ denote the last state of the path. Let $Path_f^{\mathcal{C}}$ denote the set of all finite paths. We omit the superscript $\mathcal{C}$ if it is clear from context.

2.1. The Logic CSL

The logic we consider is CSL [6] without steady-state operators and without unbounded until. Let $I = [t, t']$ be an interval with $t, t' \in \mathbb{R}_{\geq 0}$ and $t \leq t'$. Let $p \in [0, 1]$ and $\trianglelefteq \in \{\leq, <, >, \geq\}$. The syntax of state formula (Φ) and path formulas (ϕ) is:

$$\Phi = a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\trianglelefteq p}(\phi), \qquad\qquad \phi = \mathcal{X}^I \Phi \mid \Phi \, \mathcal{U}^I \, \Phi \, .$$

The semantics of the state formulas $a, \neg\Phi, \Phi_1 \wedge \Phi_2$ is defined in the same way as for CTL: $s \models_{\mathcal{C}} a$ iff $a \in L(s)$, $s \models_{\mathcal{C}} \neg\Phi$ iff $s \not\models_{\mathcal{C}} \Phi$, $s \models_{\mathcal{C}} \Phi_1 \wedge \Phi_2$ iff $s \models_{\mathcal{C}} \Phi_1$ and $s \models_{\mathcal{C}} \Phi_2$. A state s satisfies the formula $\mathcal{P}_{\trianglelefteq p}(\phi)$ if the probability of set of paths satisfying ϕ meets the bound $\trianglelefteq p$. More precisely, we have that $s \models_{\mathcal{C}} \mathcal{P}_{\trianglelefteq p}(\phi)$ iff $Pr_s(\phi) \trianglelefteq p$ where $Pr_s(\phi)$ denotes probability measure $Pr_s\{\sigma \in Path_\infty(s) \mid \sigma \models_{\mathcal{C}} \phi\}$ over $Path_\infty(s)$. Let $\sigma \in Path_\infty$, the semantics of the path formula is defined by:

$$\sigma \models_{\mathcal{C}} \mathcal{X}^I \Phi \text{ iff } \sigma[1] \models_{\mathcal{C}} \Phi \wedge \delta(\sigma, 0) \in I$$
$$\sigma \models_{\mathcal{C}} \Phi \, \mathcal{U}^I \, \Psi \text{ iff } \exists t \in I. (\sigma@t \models_{\mathcal{C}} \Psi \wedge \forall t' \in [0, t). \sigma@t' \models_{\mathcal{C}} \Phi) \, .$$

We say that the model $\mathcal{C}$ satisfies a formula Φ if the initial state s_{init} does. If $\Phi = \mathcal{P}_{\trianglelefteq p}(\phi)$ is a probabilistic formula and $\mathcal{C}$ has an initial distribution μ, $\mathcal{C}$ satisfies a formula Φ if $Pr_\mu(\phi) \trianglelefteq p$ where $Pr_\mu(\phi) := \sum_{s \in Supp(\mu)} \mu(s) Pr_s(\phi) \trianglelefteq p$ denotes the probability measure over $Path_\infty$. If the model $\mathcal{C}$ is clear from the context, we write $\models$ instead.

We introduce two shorthand notations: the *eventually operator* $\Diamond^I \Phi := true \, \mathcal{U}^I \, \Phi$, and the *always operator* $\Box^I \Phi := \neg \Diamond^I \neg \Phi$. Let $B \subseteq S$ be a set of states. For better readability, we use the name of the set B as an atomic proposition in formulas to characterize that the system is in a state contained in B.

Dependability measures. We give several important dependability measures [6] that can be expressed in CSL. $\mathcal{P}_{\trianglelefteq p}(\Diamond^I B)$ expresses that the probability of reaching a B-state within interval I meets the bound $\trianglelefteq p$. Assume that up is an atomic proposition which characterizes all states in which the system is operational. The instantaneous availability at time t is expressed by $\mathcal{P}_{\trianglelefteq p}(\Diamond^{[t,t]}up)$. The interval availability can be expressed by $\mathcal{P}_{\trianglelefteq p}(\Box^{[t,t']}up)$ where $0 \leq t < t'$. Let the state formula Φ denote some condition, then the conditional instantaneous availability at time t is expressed by $\mathcal{P}_{\trianglelefteq p}(\Phi\, \mathcal{U}^{[t,t]}\ up)$.

2.2. Model Checking CSL on finite CTMCs

We review the CSL model checking algorithm introduced by [6]. For a CTMC $\mathcal{C}$, an initial state s_{init}, and a CSL state formula Φ, we want to check whether $s_{init} \models \Phi$. Throughout this section, we assume that the set of states S is finite. The basic model checking strategy is as for CTL [10]: For every state sub-formula Ψ of Φ, we recursively compute the set $Sat(\Psi)$ satisfying the state formula Ψ. Then, $s_{init} \models \Phi$ iff $s_{init} \in Sat(\Phi)$. As for CTL, atomic propositions and Boolean connectives can be handled directly: $Sat(a) = \{s \mid a \in L(s)\}$, $Sat(\neg\Phi) = S \setminus Sat(\Phi)$ and $Sat(\Phi \wedge \Psi) = Sat(\Phi) \cap Sat(\Psi)$. The probabilistic operator $\Phi = \mathcal{P}_{\trianglelefteq p}(\phi)$ is more involved. The case $\phi = \mathcal{X}^I \Psi$ is simple and is omitted here. The most interesting case is the formula $\mathcal{P}_{\trianglelefteq p}(\Phi_1\, \mathcal{U}^I\ \Phi_2)$ which is reducible to transient analysis.

Transient analysis. Starting at state s, the transient probability vector at time t, denoted by $\vec{\pi}(s,t)$, is the probability distribution over states at time t.[1] If $t = 0$, we have $\vec{\pi}(s,0)(s') = 1$ if $s = s'$ and 0 otherwise. The formal solution [29] is given by: $\vec{\pi}(s,t) = \vec{\pi}(s,0)e^{\mathbf{Q}t}$ where $\mathbf{Q} := \mathbf{R} - diag(\mathcal{E})$ is the *generator* matrix and $diag(\mathcal{E})$ denotes the diagonal matrix with $diag(\mathcal{E})(s,s) = \mathbf{R}(s,S)$. Numerical algorithm such as *uniformization* can be used to compute the matrix exponential $e^{\mathbf{Q}t}$. We recall the uniformization algorithm [29] for the computation of transient probabilities. Let q be a uniformization rate satisfying $q \geq \sup_{s\in S} \mathbf{R}(s,S)$. Let $\mathbf{P}$ denote the transition matrix of the uniformized DTMC $uni(\mathcal{C})$, thus it holds that $\mathbf{P} = I + \mathbf{Q}/q$ where I denotes the identity matrix. Let $t > 0$, then:

$$\vec{\pi}(s,t) = \vec{\pi}(s,0)e^{\mathbf{Q}t} = \vec{\pi}(s,0)e^{(\mathbf{P}-I)qt} = \vec{\pi}(s,0)e^{-qt}\sum_{i=0}^{\infty}\frac{(qt)^i}{i!}\mathbf{P}^i = \sum_{i=0}^{\infty}\varphi(i,qt)\vec{\pi}(s,i) \qquad (1)$$

where $\varphi(i,qt) = e^{-qt}\frac{(qt)^i}{i!}$ denotes the i-th Poisson probability with parameter qt. In this formula, the vector $\vec{\pi}(s,i)$ is the transient probability of $uni(\mathcal{C})$ at step i, i.e., $\vec{\pi}(s,i) = \vec{\pi}(s,0)\mathbf{P}^i$. Intuitively, any number of transitions might happen in $\mathcal{C}$ within time t, but the probability to see precisely i transitions within that time is governed by a Poisson probability with parameter qt. Hence, the transient probability of $\mathcal{C}$ equals the infinite sum of the transient probability of $uni(\mathcal{C})$ at step i weighted by the corresponding Poisson probability. Using the Fox-Glynn algorithm [11], the Poisson probabilities $\varphi(i,qt)$ can be computed in a stable way. The number of terms needed in Equation 1 is in the order of $\mathcal{O}(qt)$ [11, 6].

Time-bounded until. Let $\mathcal{C} = (S,\mathbf{R},L)$ be a CTMC, and let $\mathcal{P}_{\trianglelefteq p}(\Phi\, \mathcal{U}^I\ \Psi)$ be a CSL formula. To check the formula $\mathcal{P}_{\trianglelefteq p}(\Phi\, \mathcal{U}^I\, \Psi)$ it is sufficient to compute the probability $Pr_s(\Phi\, \mathcal{U}^I\, \Psi)$. To see how this works, we first consider the simple case $I = [0,t]$. For a state formula Φ', we let $\mathcal{C}[\Phi']$ denote the CTMC obtained by making Φ'-states absorbing. More precisely, $\mathcal{C}[\Phi'] = (S, \mathbf{R}^{\mathcal{C}[\Phi']}, L)$ where $\mathbf{R}^{\mathcal{C}[\Phi']}(s,s')$

[1] Transient probability for infinite states CTMCs is defined exactly the same way.

equals $\mathbf{R}(s, s')$ if $s \not\models \Phi'$, equals 0 otherwise. As explained in [6], the transformation of $\mathcal{C}$ to $\mathcal{C}[\Psi]$ does not change the probability $Pr_s(\Phi\, \mathcal{U}^I\, \Psi)$, as once Ψ-states are reached, it does not matter what happens afterwards. Also the $\neg\Phi \wedge \neg\Psi$-states are made absorbing, as once such a state is reached, the path cannot satisfy $\Phi\, \mathcal{U}^I\, \Psi$ anymore. The computation of the probability $Pr_s^{\mathcal{C}}(\Phi\, \mathcal{U}^I\, \Psi)$ can then be reduced to transient probability analysis in the CTMC $\mathcal{C}[\neg\Phi \vee \Psi]$. Note that $\mathcal{C}[\neg\Phi \vee \Psi]$ is equivalent to $\mathcal{C}[\Psi][\neg\Phi \wedge \neg\Psi]$. The following theorem recalls how to compute $Pr_s^{\mathcal{C}}(\Phi\, \mathcal{U}^I\, \Psi)$ for general time intervals I via transient analysis [6]:

Theorem 2.1. For any $\mathcal{C}$, we have the equality:

$$Pr_s^{\mathcal{C}}(\Phi\, \mathcal{U}^I\, \Psi) = \begin{cases} \sum_{s' \models \Psi} \vec{\pi}^{\mathcal{C}[\neg\Phi\vee\Psi]}(s,t)(s') & \text{if } I = [0,t] \\ \sum_{s' \models \Phi\wedge\Psi} \vec{\pi}^{\mathcal{C}[\neg\Phi]}(s,t)(s') & \text{if } I = [t,t] \\ \sum_{s' \models \Phi} \sum_{s'' \models \Psi} \vec{\pi}^{\mathcal{C}[\neg\Phi]}(s,t)(s') \cdot \vec{\pi}^{\mathcal{C}[\neg\Phi\vee\Psi]}(s',t'-t)(s'') & \text{if } I = [t,t'] (0 < t < t') \end{cases}$$

3. Model Checking based on Truncations

Now we discuss how to check CSL formulas against infinite CTMCs. To this end, our goal is to operate on a finite truncation instead of the original infinite CTMC. In a nutshell, the k-truncation of a CTMC with respect to a truncation depth k is obtained by collapsing all states with depth larger than k to one absorbing state. This is formalized in Subsection 3.1. Truncation introduces an error which decreases with increasing truncation depth k. Subsection 3.2 explains algorithms that compute a sufficiently large truncation depth k such that the transient probability at time t can be approximated in the k-truncation with the desired precision. In Subsection 3.3, we discuss how to check whether a CSL formula is satisfied by a CTMC. Lastly, in Subsection 3.4, extensions to models with rewards are discussed.

3.1. Finite Truncations

In this subsection, we define *finite truncations* of a CTMC. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC. First, we introduce the notion of *depth*. Let S_0 be a finite subset of S with depth 0, i.e., $d(s) = 0$ for $s \in S_0$. For now, one may think of S_0 as being equal to the support of the initial distribution μ. However, S_0 can be an arbitrary finite set. This will allow us to deal with not only the initial distribution, and to compute truncation depths for nested CSL formulas. The depth of state s corresponds to the minimal distance from the set S_0:

Definition 3.1. For $\mathcal{C}$ and $S_0 \subset S$, the *depth* function $d : S \to \mathbb{N}$ is defined by:

$$d_{S_0}(s) = \min\{len(\sigma) \mid \sigma \in Path_f \wedge first(\sigma) \in S_0 \wedge last(\sigma) = s\} .$$

Observe that $d_{S_0}(s) = 0$ for all $s \in S_0$. The subscript is omitted if S_0 is clear from the context. Intuitively, $d(s)$ corresponds to the minimal length of any finite path starting from S_0 and ending in s.

We consider a partition of the state space $S = \bigcup_{i\in\mathbb{N}} S_i$ where $S_i := \{s \in S \mid d(s) = i\}$ is the set of states with depth i. We say that the set S_i is the layer with depth i, and call its elements layer-i states. Assume $\perp \notin S$ is a special state not in S, and furthermore, $\perp$ is also an atomic proposition not in AP. For $k \in \mathbb{N}$, let $S_{>k} \subset S$ denote the set of states with depth greater than k, i.e., $S_{>k} = \{s \mid d(s) > k\}$. Further, we write $S_{\leq k} = S \setminus S_{>k}$. We define the k-truncated CTMC as follows:

Definition 3.2. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC and let S_0 be the layer-zero states. For $k \in \mathbb{N}$, we define the k-truncated CTMC of $\mathcal{C}$ by: $\mathcal{C}|_k = (S|_k, \mathbf{R}_k, L_k)$ where the states are $S|_k = S_{\leq k} \cup \{\bot\}$. The labeling function $L_k : S|_k \rightarrow AP \cup \{\bot\}$ is defined by $L_k(s) = L(s)$ if $s \in S_{\leq k}$, and $L_k(\bot) = \{\bot\}$. The rate matrix is defined by: $\mathbf{R}_k(s, s')$ equals $\mathbf{R}(s, s')$ if $s, s' \in S_{\leq k}$, equals $\mathbf{R}(s, S_{>k}) = \sum_{t \in S_{>k}} \mathbf{R}(s, t)$ if $s \in S_k$, $s' = \bot$, and equals 0 otherwise.

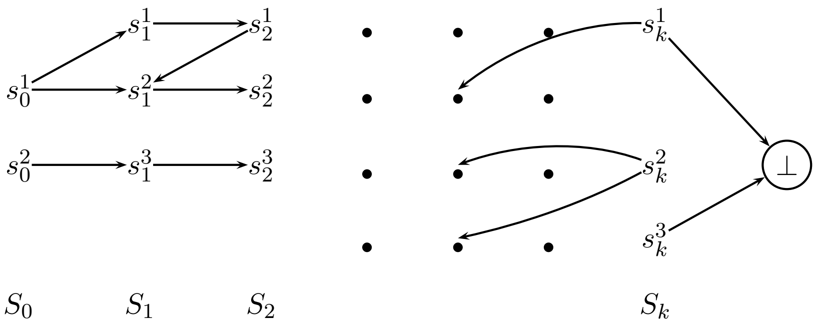

Figure 1. The k-th truncation $\mathcal{C}|_k$ for an infinite CTMC. Rates are omitted. S_i denotes the set of layer i states.

The k-truncation of an infinite CTMC is illustrated in Figure 1. Intuitively, the transition matrix is restricted to the truncated state space $S|_k$, and $\bot$ is the distinguished absorbing state, which, by construction, is only reachable from states with depth k. In state $\bot$ only the atomic proposition $\bot$ holds, which indicates that the system is in state $\bot$. Since we consider finitely branching CTMCs, not surprisingly, the k-truncated CTMC $\mathcal{C}|_k$ is always finite. The absorbing state $\bot$ has been introduced to abstract $S_{>k}$. We assume that $\bot \notin Sat(\Phi)$ for any state formula Φ. We consider the probability of reaching the absorbing state $\bot$ in the k-truncated CTMC $\mathcal{C}|_k$, i.e., $\vec{\pi}^{\mathcal{C}|_k}(s, t)(\bot)$. For mere notational convenience, we extend $\vec{\pi}^{\mathcal{C}|_k}$ to states of $\mathcal{C}$ with depth higher than k: $\vec{\pi}^{\mathcal{C}|_k}(s, t)(s') = 0$ for all $s' \in S$ with $d(s') > k$.

For a fixed k, we define the *forward rate* $\mathit{fr}_k(s)$ of a state $s \in S_{\leq k}$ within $\mathcal{C}|_k$. For $s \in S_{\leq k} \setminus S_k$, it is the sum of the rates that go into the next layer $\mathit{fr}_k(s) = \mathbf{R}(s, S_{d(s)+1})$, and, for $s \in S_k$, it is the sum $\mathit{fr}_k(s) = \mathbf{R}(s, S_{>k})$ of the rates entering states in $S_{>k}$.

3.2. Truncation-based Transient Analysis

We want to compute the transient probability at time t up to accuracy ε, starting at some layer-zero state $s \in S_0$. Since the model may be infinite, we approximate $\vec{\pi}(s, t)$ for $s \in S_0$ by computing the transient probability in the finite truncation. To achieve a sufficiently precise approximation, truncation-based transient analysis enlarges the truncation iteratively until the error is small enough.

The approximation error introduced by truncation can be explained as follows. For a state s in $S|_k$, its transient probability in the original CTMC $\mathcal{C}$ is higher than in $\mathcal{C}|_k$ because, by pruning the matrix, we have summarized probability that flows into $S_{>k}$ and removed flows back to state s. In the worst case, all the probability entering $S_{>k}$ comes back to s. Therefore, the idea behind error estimation is to bound the probability mass entering the truncated states. We discuss several error-estimation methods. One way to obtain an error bound is to directly compute this probability which is the strategy behind finite state projection [24, 26]. The disadvantage is that this incurs the overhead of an additional transient analysis which has to be repeated every time the truncation depth increases. Our dynamic-uniformization-based error estimation method [35] provides a cheaper but, in general, less precise error estimation, which yields an upper bound on this probability. While error estimation is also repeated after each increase of truncation depth, the cost per estimation is reduced significantly because we approximate the truncated model by a chain whose length is linear in the truncation depth. This is sound because the chain converges faster to its absorbing state than the truncated model.

In this section, we first discuss how to obtain the finite state projection algorithm due to [24]. Then, we discuss several chain-based error estimation technique.

3.2.1. Finite State Projection

Recently, Munsky [24, 26] introduced the *finite state projection* algorithm for the transient analysis of CTMCs, in the context of the Chemical Master Equation. The idea is as follows. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be an infinite CTMC and let S_0 be its set of layer-zero states. Let $\mathcal{C}|_k = (S|_k, \mathbf{R}_k, L_k)$ be its k-th truncation. Recall $\perp \in S|_k$ abstracts all states $s \in S$ with depth greater than k. Let $s_0 \in S_0$ and consider an arbitrary state $s' \in S_{\leq k}$. Then, the transient probability $\vec{\pi}^{\mathcal{C}|_k}(s_0, t)(s')$ is a joint probability that the system is at state s' at time t and that the system has never left the set $S_{\leq k}$ within $[0, t)$. The transient probability $\vec{\pi}^{\mathcal{C}}(s_0, t)(s')$ can be considered as the probability that the system $\mathcal{C}$ is at state s' at time t.

Clearly, $\vec{\pi}^{\mathcal{C}|_k}(s_0, t)(s')$ is a lower bound for $\vec{\pi}^{\mathcal{C}}(s_0, t)(s')$. Observe, however, that the system may evolve to states with depth larger than k and then return to s'. This probability gets lost by truncation. The worst-case assumption, in terms of the error, is that all probability that goes into the states $S_{>k}$ goes back to s'. Therefore, $\vec{\pi}^{\mathcal{C}}(s_0, t)(s')$ is bounded from above by $\vec{\pi}^{\mathcal{C}|_k}(s_0, t)(s') + \vec{\pi}^{\mathcal{C}|_k}(s_0, t)(\perp)$. Let $\varepsilon_{\mathcal{C}, s_0, k} := \vec{\pi}^{\mathcal{C}|_k}(s_0, t)(\perp)$. If $\mathcal{C}$ is clear from the context, we write $\varepsilon_{s_0,k}$ instead. We denote by $\mathcal{C}[S_{>k}]$ the model obtained from $\mathcal{C}$ by making the states in $S_{>k}$ absorbing and observe that the k-truncated CTMC $\mathcal{C}|_k$ equals $\mathcal{C}[S_{>k}]$ up to the absorbing goal states. Therefore probability $\varepsilon_{s_0,k}$ is exactly $Pr^{\mathcal{C}}_{s_0}(\Diamond^{[0,t]} S_{>k})$, which also implies that the sequence $(\varepsilon_{s_0,k})_{k \in \mathbb{N}}$ is decreasing with limit 0. Further, observe that the transient probability of any state in $\mathcal{C}|_{k+1}$ is at least as high as in $\mathcal{C}|_k$. More formally it holds that [24, 26]:

Theorem 3.1. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be an infinite CTMC and and let S_0 be the set of layer zero states. For any k, let $\mathcal{C}|_k = (S|_k, \mathbf{R}_k, L_k)$ be its k-th truncation. Then for all $s_0 \in S_0$ and each $s' \in S$, it holds that

$$\vec{\pi}^{\mathcal{C}|_k}(s_0, t)(s') \leq \vec{\pi}^{\mathcal{C}}(s_0, t)(s') \leq \vec{\pi}^{\mathcal{C}|_k}(s_0, t)(s') + \varepsilon_{s_0,k} \tag{2}$$

$$\vec{\pi}^{\mathcal{C}|_k}(s_0, t)(s') \leq \vec{\pi}^{\mathcal{C}|_{k+1}}(s_0, t)(s') \tag{3}$$

Since $(\varepsilon_{s_0,k})_{k \in \mathbb{N}}$ is a decreasing sequence converging to 0, it follows that $(\vec{\pi}^{\mathcal{C}|_k}(s_0, t)(s'))_{k \in \mathbb{N}}$ is an increasing sequence converging to $\vec{\pi}^{\mathcal{C}}(s_0, t)(s')$. The following theorem [24, 26] states that the one-norm error of $\vec{\pi}^{\mathcal{C}|_k}(s_0, t)$ is exactly $\varepsilon_{s_0,k}$:

Theorem 3.2. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be an infinite CTMC, let S_0 be the set of layer zero states, and $\mathcal{C}|_k = (S|_k, \mathbf{R}_k, L_k)$ the k-th truncation of $\mathcal{C}$. T Then, for each $s_0 \in S_0$:

$$\left|\left|\vec{\pi}^{\mathcal{C}}(s_0, t) - \vec{\pi}^{\mathcal{C}|_k}(s_0, t)\right|\right|_1 = \varepsilon_{s_0,k} \tag{4}$$

```
TRANSIENTTRUNC(C, S_0, t, ε)
1  i = 1
2   Construct the truncation C|_i and construct a chain X_i which over-approximates C|_i
3   Compute the probability δ_{X,i}
4  if δ_{X,i} < ε
5     then return C|_i
6     else   Increase i and go to line 2
```

Figure 2. Algorithm for exploring the state space with respect to time bound t. We denote by S_0 the layer-zero states. The chain X_i can be either the uniform chain $\mathcal{C}_{U,k}$ or the layered chain $\mathcal{C}_{L,k}$. If the finite state projection method is used, the probability $\delta_{X,i}$ is the maximal probability of reaching $\perp$ in $\mathcal{C}|_i$, namely $\max_{s_0 \in S_0} \varepsilon_{s_0,i}$. Otherwise, the probability $\delta_{X,i}$ is the probability of reaching the distinguished absorbing state $\perp$ in the chain X_i.

Theorems 3.1 and 3.2 suggest an approach, called *finite state projection* algorithm in [24, 26]. Assume that the initial distribution is μ. Then, taking $S_0 = Supp(\mu)$, the algorithm (with N-step reachability strategies) explores the system $\mathcal{C}$ up to depth k, until $\varepsilon_{s_0,k} < \varepsilon$ for all $s_0 \in S_0$. The algorithm, depicted in Figure 2, yields the minimal truncation depth k. However, the computational costs of computing the error $\varepsilon_{s_0,i}$ for $i = 1, ..., k$ can be prohibitive, since it involves k separate transient analyses. In the following, we discuss techniques that are faster, but compute upper bounds on the error and thus often produce a larger truncation depth.

3.2.2. Chain-Based Error Estimation

Finite state projection (see Subsection 3.2.1) performs error estimation on the current truncated model $\mathcal{C}|_i$, which can be costly. We now develop a procedure, called chain-based error estimation, which is cheaper at the price of producing a potentially less precise error bound. It follows the same general algorithm as the finite state projection approach (see Figure 2). However, as indicated at line 2, a chain is constructed which over-approximates $\mathcal{C}|_i$ and from which the error bound is obtained. Analogous to finite state projection, the error bound is obtained from the probability of reaching the absorbing end-states, however not of the full truncated model, but rather of the chain X_i. The size of the chain is linear in the truncation depth and thus has, in general, a much simpler structure than the truncated model, making transient analysis more efficient. Intuitively, the chain moves into the absorbing state with higher rates than the truncated model, thus yielding an upper bound on the probability to get into the absorbing state of the truncated model. We focus on two particular ways to construct such chains.

Uniform Chain. We denote the maximum over the forward rates by constant $q_k = \max_{s \in S_{\leq k}} fr_k(s)$. The uniform chain of $\mathcal{C}_{U,k}$ is the CTMC with states $S_{U,k} = \{u_0, ..., u_k, \perp\}$, initial state u_0, and the *uniform* rate function $\mathbf{R}_{U,k}(u_i, u_{i+1}) = q_k$ where we assume that $u_{k+1} = \perp$. The uniform chain is depicted in Figure 3. We denote the probability of reaching the absorbing state by $\delta_{U,k} = \vec{\pi}^{\mathcal{C}_{U,k}}(u_0, t)(\perp)$. Value $\delta_{U,k}$ is exactly the $k+1$-stage Erlang distribution with rate q_k, i.e.,

$$\delta_{U,k} = 1 - \sum_{i=0}^{k} \frac{e^{-q_k t}(q_k t)^i}{i!} \tag{5}$$

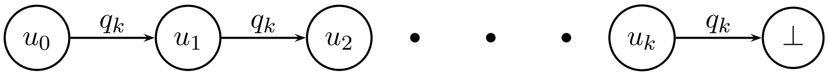

Figure 3. Uniform chain $\mathcal{C}_{U,k}$

The Fox-Glynn algorithm [11] can be used to compute this efficiently. Note that, for doing so, the chain does not have to be built up explicitly.

As shown in Lemma A.3, the error bound $\delta_{U,k}$ obtained from the uniform chain fulfills $\varepsilon_{s_0,k} \leq \delta_{U,k}$ for all $s_0 \in S_0$. If the CTMC is not rate-bounded, however, this can lead to non-termination of the truncation algorithm. To see this, assume that the rates are monotonically increasing in each level, i.e., we have the sequence $q_k = k$ for $k = 1, 2, \ldots$. Assume that $t = 1$. Using Equation 5, one obtains the error bound $1 - \sum_{i=0}^{k} \frac{e^{-k} k^i}{i!}$, which is a sequence in k. One can show that this sequence never decreases [1, p. 263].[2]

Layered Chain. We define the sequence $f_0, \ldots, f_k$ of maximal forward rates $f_i = \max_{s \in S_i} fr_k(s)$ of layer-i states for each layer $i \in \{1, \ldots, k\}$. The layered chain of $\mathcal{C}_{L,k}$ is the CTMC with states $S_{L,k} = \{l_0, \ldots, l_k, \perp\}$, initial state l_0, and rate function $\mathbf{R}_{L,k}(l_i, l_{i+1}) = f_i$ where $l_{k+1} = \perp$. The layered chain is depicted in Figure 4.

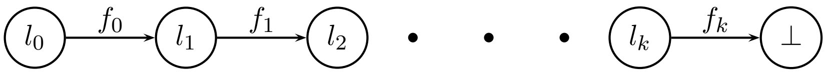

Figure 4. Layered chain $\mathcal{C}_{L,k}$

We denote the probability of reaching the absorbing state by $\delta_{L,k} = \vec{\pi}^{\mathcal{C}_{L,k}}(l_0, t)(\perp)$. Note that $q_k = \max_{i \leq k} f_i$ which means that the layered chain is "slower" than the uniform chain, implying that the layered chain yields an error bound $\delta_{L,k}$ that lies below $\delta_{U,k}$. As shown in Lemma A.3, this bound is still safe, i.e., $\varepsilon_{s_0,k} \leq \delta_{L,k}$ for all $s_0 \in S_0$.

If the layered chain is used, the algorithm Algorithm TRANSIENTTRUNC in Figure 2 is guaranteed to terminate, as the error bound converges to zero. This follows by the construction of the chain and the assumption that $\mathcal{C}$ does not explode. The layered chain can be generated on the fly, and $\delta_{L,k}$ can be computed incrementally using the uniformization technique, provided that the rate f_{k+1} to the newly explored layer does not exceed the uniformization rate q_k. If, after some iterations, the maximal rate f_k does not increase any more, the computation of $\delta_{L,k}$ can be performed very efficiently, as shown later in our case studies.

Both uniform and layered chain ignore rates that go backwards in terms of depth. This can be explained by the intuition that backwards rates do not contribute to reaching the truncated state space.

[2] We thank Michael Burger and Holger Lang (Fraunhofer ITWM Kaiserslautern) for helping us verify that this sequence is indeed increasing. Moreover, they have shown that the limit of this sequence is $\frac{1}{2}$, which is stated in [1]: This limit can be obtained by setting $\alpha = 1$ of the equation 6.5.34 on page 263.

Now we discuss how the chains relate to approaches from the literature, which maximize over the total exit rate when constructing the chains.

A few remarks as regards the relation to previous work in [35], [12] and [31] are in order:

- In [35], we have used *dynamic* uniformization by Grassmann [12]. This corresponds to the uniform chain but with total exit rate, i.e, $q_k = \max_{i=0,\dots,k} \max_{s\in S_i} \mathbf{R}(s, S|_k)$. Therefore the resulting chain has a higher probability of reaching $\perp$, while the computational complexity remains the same as for the uniform chain proposed here.

- The adaptive uniformization method [31] constructs a chain for computing the jump probability. This chain corresponds to our layered chain, but with total exit rate, i.e., $f_i = \max_{s\in S_i} \mathbf{R}(s, S|_k)$. Again, the corresponding chain then has a higher probability of reaching $\perp$. Hence, the resulting chain has a higher probability of reaching $\perp$, while the computational complexity remains the same as the layered chain.

3.3. Truncation-based CSL Model Checking

Given a CTMC $\mathcal{C}$ with an initial state or, more generally, an initial distribution μ, and a CSL formula Φ, we want to check if $\mathcal{C}$ satisfies Φ. This is done in two phases:

- We construct a finite truncation that is sufficient to check the formula. To this end, we employ an algorithm to determine suitable truncation depths. The truncation depth depends on the specific CSL formula to be analyzed. The computation of the truncation depth works by recursive descent into sub-formulas. Different sub-formulas may require a different truncation depth. These depths need to be computed and combined in a way that respects the error bound. The most intricate formulas are the probabilistic operators, which correspond to transient analysis. Here the algorithms from the previous section come into play.

- We check whether $\mu \models \Phi$ holds on the finite truncation. Here we employ the CSL model checking algorithm outlined in Section 2.2.

We now develop the key component of this two-phase approach: the algorithm to compute suitable truncation depths. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC with initial distribution μ, and let Φ be a CSL state formula. Algorithm TRUNCATION$(\mathcal{C}, \mu, \Phi)$, depicted in Figure 5, computes the truncation depth for Φ. This is obtained via a call to subroutine TRUNC$(\mathcal{C}, Supp(\mu), \Phi)$ with the support of μ,

Subroutine TRUNC determines truncation depth for the different CSL constructs. More precisely, TRUNC$(\mathcal{C}, S_0, \Phi)$ returns a truncation depth k so that the truncation $\mathcal{C}|_k$ with layer-zero states S_0 is sufficient, in order to check whether states in S_0 satisfy Φ. The truncation depth for boolean operators, and the probabilistic operator with next operator are obvious. The probabilistic operator employs Algorithm TRANSIENTTRUNC, which gives the truncation depth for transient analysis (see Figure 2), when determining truncation depth for the probabilistic operator. As discussed in the previous section, TRANSIENTTRUNC can be implemented by FSP, Uniform Chain, or Layered Chain error estimation, which gives rise to different *configurations* of the truncation algorithm TRUNCATION.

For the probabilistic operator with until operator, the truncation is the sum of the truncation depth necessary for carrying out the transient analysis, and the truncation depth of the state sub-formulas.

$\textsc{Trunc}(\mathcal{C}, S_0, \Phi)$
switch
  case $\Phi = a$:
    return 0
  case $\Phi = \neg\Psi$:
    return $\textsc{Trunc}(\mathcal{C}, S_0, \Psi)$
  case $\Phi = \Phi_1 \wedge \Phi_2$:
    return $\max\{\textsc{Trunc}(\mathcal{C}, S_0, \Phi_1), \textsc{Trunc}(\mathcal{C}, S_0, \Phi_2)\}$
  case $\Phi = \mathcal{P}_{\trianglelefteq p}(\mathcal{X}^I \Psi)$:
    return $1 + \textsc{Trunc}(\mathcal{C}, S_1, \Psi)$
  case $\Phi = \mathcal{P}_{\trianglelefteq p}(\Phi_1\, \mathcal{U}^I\, \Phi_2)$ with $I = [0, t]$ or $I = [t, t]$:
    $k = \textsc{TransientTrunc}(\mathcal{C}, S_0, t, \varepsilon)$
    return $k + \max\{\textsc{Trunc}(\mathcal{C}, S_k, \Phi_1), \textsc{Trunc}(\mathcal{C}, S_k, \Phi_2)\}$
  case $\Phi = \mathcal{P}_{\trianglelefteq p}(\Phi_1\, \mathcal{U}^{[t,t']}\, \Phi_2)$ with $0 < t < t'$:
    $k = \textsc{TransientTrunc}(\mathcal{C}, S_0, t, \frac{\varepsilon}{2})$
    $k' = \textsc{TransientTrunc}(\mathcal{C}, S_k, t' - t, \frac{\varepsilon}{2})$
    return $k + k' + \max\{\textsc{Trunc}(\mathcal{C}, S_{k+k'}, \Phi_1), \textsc{Trunc}(\mathcal{C}, S_{k+k'}, \Phi_2)\}$

$\textsc{Truncation}(\mathcal{C}, \mu, \Phi)$
return $\textsc{Trunc}(\mathcal{C}, \mathit{Supp}(\mu), \Phi)$

Figure 5. Algorithm for exploring the state space with respect to CSL formulas. The set S_0 denotes the layer zero states. Initially we have $S_0 = \mathit{Supp}(\mu)$ where μ is the initial distribution.

Consider a time interval of the form $[0, t]$. For layer-zero state $s \in S_0$, we want to calculate the probability $Pr_s(\Phi_1\, \mathcal{U}^{[0,t]}\, \Phi_2)$. By Theorem 2.1, it equals $\sum_{s' \models \Phi_2} \vec{\pi}^{\mathcal{C}[\neg\Phi_1 \vee \Phi_2]}(s, t)(s')$. We thus need to compute the transient probability at time t, starting from s. For this, all states with depth smaller than $k := \textsc{TransientTrunc}(\mathcal{C}, S_0, t, \varepsilon)$ must be considered. Moreover, the transient probabilities are computed in the model $\mathcal{C}[\neg\Phi_1 \vee \Phi_2]$, which indicates that we must know whether the states in the truncation satisfy the sub-formulas Φ_1 and Φ_2. Thus, the truncation depth in this case is given by: $k + \max\{\textsc{Trunc}(\mathcal{C}, S_k, \Phi_1), \textsc{Trunc}(\mathcal{C}, S_k, \Phi_2)\}$. Intervals of the form $[t, t]$ can be handled similarly.

Consider the case $I = [t, t']$ with $0 < t < t'$. Again, we compute the probability $Pr_s(\Phi_1\, \mathcal{U}^{[t,t']}\, \Phi_2)$ for $s \in S_0$ using the third case of Theorem 2.1. For the first transient probability $\vec{\pi}^{\mathcal{C}[\neg\Phi_1]}(s, t)(s')$ in $\mathcal{C}[\neg\Phi_1]$, the k-truncated CTMC is required where $k = \textsc{TransientTrunc}(\mathcal{C}, S_0, t, \frac{\varepsilon}{2})$. As we will show later, to guarantee that the truncation induces error up to ε, we need to take $\frac{\varepsilon}{2}$ for the transient analysis with respect to time bound t. Let S_k denote the layer-k states with respect to S_0. For every state $s' \models \Phi_1$ with $d(s') \le k$, the other transient probability $\vec{\pi}^{\mathcal{C}[\neg\Phi_1 \vee \Phi_2]}(s', t' - t)(s'')$ needs to be computed. This indicates, starting from S_k as the new set layer zero states, all states with depth smaller or equal than $k' = \textsc{TransientTrunc}(\mathcal{C}, S_k, t' - t, \frac{\varepsilon}{2})$ need to be explored. Moreover, as in the case $I = [0, t]$, dealing with the sub-formulas Φ_1 and Φ_2 requires $\max\{\textsc{Trunc}(\mathcal{C}, S_{k+k'}, \Phi_1), \textsc{Trunc}(\mathcal{C}, S_{k+k'}, \Phi_2)\}$ additional depth. Altogether, the algorithm takes $k+k'+\max\{\textsc{Trunc}(\mathcal{C}, S_{k+k'}, \Phi_1), \textsc{Trunc}(\mathcal{C}, S_{k+k'}, \Phi_2)\}$ as the truncation depth.

Correctness. The correctness proof of our truncation algorithm is decomposed into two parts. Algorithm TRUNCATION computes the truncation depth for a CSL formula and calls algorithm TRUNC as a subroutine. We first state the correctness of Algorithm TRUNC in Theorem 3.3. Overall correctness follows by Theorem 3.4, which deals with Algorithm TRUNCATION.

Theorem 3.3. Let $\mathcal{C}$ be a CTMC. Let $\Phi = \mathcal{P}_{\trianglelefteq p}(\phi)$ be the probabilistic state formula. Let $S_0 \subset S$ be a set of layer-zero states, and let $k = \text{TRUNC}(\mathcal{C}, S_0, \Phi)$ be the truncation depth obtained by Algorithm TRUNC. Then, for $s \in S_0$,

$$Pr_s^{\mathcal{C}}(\phi) - \varepsilon \leq Pr_s^{\mathcal{C}|_k}(\phi) \leq Pr_s^{\mathcal{C}}(\phi) \tag{6}$$

Proof:
First we consider the path formula $\phi = \mathcal{X}^I \Psi$. It is sufficient to compute the set of states satisfying Ψ which can be reached from $s \in S_0$ directly. By definition of TRUNC, we have $k = 1 + \text{TRUNC}(\mathcal{C}, S_1, \Psi)$, where $\text{TRUNC}(\mathcal{C}, S_1, \Psi)$ is computed recursively. Thus, for $s' \in S_0 \cup S_1$, we can check whether $s' \models \Psi$ holds in the truncation $\mathcal{C}|_k$. Thus $Pr_s^{\mathcal{C}|_k}(\phi) = Pr_s^{\mathcal{C}}(\phi)$ for $s \in S_0$, implying Inequality 6 directly.

Now we consider the until path formula: $\phi = \Phi_1 \, \mathcal{U}^I \, \Phi_2$ where $I = [t, t']$ with $t \leq t'$. Our goal is to check whether Inequality 6 holds for layer-zero states $s \in S_0$. We do a case distinction by the shape of the interval I: First consider the case $I = [0, t]$. We denote the truncation depth for transient analysis with respect to t by $k^* = \text{TRANSIENTTRUNC}(\mathcal{C}, S_0, t, \varepsilon)$. We have that

$$k = k^* + \max\{\text{TRUNC}(\mathcal{C}, S_{k^*}, \Phi_1), \text{TRUNC}(\mathcal{C}, S_{k^*}, \Phi_2)\} \geq k^* \ .$$

The additional depth $\max\{\text{TRUNC}(\mathcal{C}, S_{k^*}, \Phi_1), \text{TRUNC}(\mathcal{C}, S_{k^*}, \Phi_2)\}$ is needed to guarantee that the model $\mathcal{C}|_k[\neg\Phi_1 \vee \Phi_2]$ can be constructed correctly. Define $x = Pr_s^{\mathcal{C}}(\phi) - Pr_s^{\mathcal{C}|_k}(\phi)$. Then, it is sufficient to show that $0 \leq x \leq \varepsilon$. By Theorem 2.1, we can rewrite x as:

$$\sum_{s' \models \Phi_2} \vec{\pi}^{\mathcal{C}[\neg\Phi_1 \vee \Phi_2]}(s,t)(s') - \sum_{s' \models \Phi_2} \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1 \vee \Phi_2]}(s,t)(s') \ .$$

By Theorem 3.1, it holds that $0 \leq x$. For the upper bound:

$$\begin{aligned} |x| &= \sum_{s' \models \Phi_2} |\vec{\pi}^{\mathcal{C}[\neg\Phi_1 \vee \Phi_2]}(s,t)(s') - \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1 \vee \Phi_2]}(s,t)(s')| \\ &\leq \sum_{s' \in S} |\vec{\pi}^{\mathcal{C}[\neg\Phi_1 \vee \Phi_2]}(s,t)(s') - \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1 \vee \Phi_2]}(s,t)(s')| \\ &= \left|\left| \vec{\pi}^{\mathcal{C}[\neg\Phi_1 \vee \Phi_2]}(s,t) - \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1 \vee \Phi_2]}(s,t) \right|\right|_1 \overset{The.\ 3.2}{=} \varepsilon_{\mathcal{C}[\neg\Phi_1 \vee \Phi_2],s,k} \end{aligned}$$

As shown in Lemma A.4, $\varepsilon_{\mathcal{C}[\neg\Phi_1 \vee \Phi_2],s,k}$ is bounded by $\varepsilon_{\mathcal{C},s,k}$. Since the sequence $\varepsilon_{\mathcal{C},s,k}$ is decreasing, we have that $\varepsilon_{\mathcal{C},s,k} \leq \varepsilon_{\mathcal{C},s,k^*} \leq \varepsilon$ holds, and thus $|x| \leq \varepsilon$.

The case $I = [t, t]$ can be analyzed in a very similar way as $I = [0, t]$ and the exact same truncation depth. Now we consider the case $\phi = \Phi_1 \, \mathcal{U}^{[t,t']} \, \Phi_2$ where $t < t'$. We denote the truncation depth for transient analysis with respect to t by $k_1 = \text{TRANSIENTTRUNC}(\mathcal{C}, S_0, t, \frac{\varepsilon}{2})$, and with respect to t' by $k_2 = \text{TRANSIENTTRUNC}(\mathcal{C}, S_{k_1}, t' - t, \frac{\varepsilon}{2})$. We have that

$$k = k_1 + k_2 + \max\{\text{TRUNC}(\mathcal{C}, S_{k_1+k_2}, \Phi_1), \text{TRUNC}(\mathcal{C}, S_{k_1+k_2}, \Phi_2)\} \geq k_1 + k_2 \ .$$

Let $x = Pr_s^{\mathcal{C}}(\phi) - Pr_s^{\mathcal{C}|_k}(\phi)$, and we show that $0 \leq x \leq \varepsilon$. By Theorem 2.1, x can be expressed by a sum over s' and s'':

$$\begin{aligned}
&\sum_{s',s''} \vec{\pi}^{\mathcal{C}[\neg\Phi_1]}(s,t)(s')\vec{\pi}^{\mathcal{C}[\neg\Phi_1\vee\Phi_2]}(s',t'-t)(s'') - \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1]}(s,t)(s')\vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1\vee\Phi_2]}(s',t'-t)(s'') \\
&= \sum_{s',s''} \vec{\pi}^{\mathcal{C}[\neg\Phi_1\vee\Phi_2]}(s',t'-t)(s'') \cdot \left(\vec{\pi}^{\mathcal{C}[\neg\Phi_1]}(s,t)(s') - \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1]}(s,t)(s')\right) \\
&\quad + \sum_{s',s''} \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1]}(s,t)(s') \cdot \left(\vec{\pi}^{\mathcal{C}[\neg\Phi_1\vee\Phi_2]}(s',t'-t)(s'') - \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1\vee\Phi_2]}(s',t'-t)(s'')\right)
\end{aligned}$$

where $s', s'' \in S|_{k_1+k_2}$ and $s' \models \Phi_1$, $s'' \models \Phi_2$: these conditions can be checked as the additional depth $\max\{\text{TRUNC}(\mathcal{C}, S_{k_1+k_2}, \Phi_1), \text{TRUNC}(\mathcal{C}, S_{k_1+k_2}, \Phi_2)\}$ guarantees that the model $\mathcal{C}|_k[\neg\Phi_1 \vee \Phi_2]$ can be constructed correctly. Let x_1 and x_2 respectively denote the above two sums. Both of the sums are positive, thus it holds that $0 \leq x$. Now we establish the upper bound on x:

$$\begin{aligned}
|x_1| &= \left| \sum_{s'\models\Phi_1} \sum_{s''\models\Phi_2} \vec{\pi}^{\mathcal{C}[\neg\Phi_1\vee\Phi_2]}(s',t'-t)(s'') \cdot \left(\vec{\pi}^{\mathcal{C}[\neg\Phi_1]}(s,t)(s') - \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1]}(s,t)(s')\right) \right| \\
&\leq \left| \sum_{s'\in S} \vec{\pi}^{\mathcal{C}[\neg\Phi_1]}(s,t)(s') - \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1]}(s,t)(s') \right| \left| \sum_{s''\in S} \vec{\pi}^{\mathcal{C}[\neg\Phi_1\vee\Phi_2]}(s',t'-t)(s'') \right| \\
&\leq \left\| \vec{\pi}^{\mathcal{C}[\neg\Phi_1]}(s,t) - \vec{\pi}^{\mathcal{C}|_k[\neg\Phi_1]}(s,t) \right\|_1 \overset{The.\ 3.2}{=} \varepsilon_{\mathcal{C}[\neg\Phi_1],s,k} \,.
\end{aligned}$$

Similarly, we can show that $|x_2| \leq \varepsilon_{\mathcal{C}[\neg\Phi_1\vee\Phi_2],s,k}$. Thus, $|x| \leq |x_1| + |x_2|$. Similar to the previous case, both $|x_1|$ and $|x_2|$ are bounded by $\frac{\varepsilon}{2}$, and we have $|x| \leq \varepsilon$. □

Theorem 3.4. Let $\mathcal{C}$ be a CTMC and let μ be the initial distribution. Let $\Phi = \mathcal{P}_{\trianglelefteq p}(\phi)$ be a probabilistic state formula. Let $S_0 := Supp(\mu)$ be the set of layer-zero states, and let $k = \text{TRUNC}(\mathcal{C}, S_0, \Phi)$ be the truncation depth returned by Algorithm TRUNC. Then we have:

$$Pr_\mu^{\mathcal{C}}(\phi) - \varepsilon \leq Pr_\mu^{\mathcal{C}|_k}(\phi) \leq Pr_\mu^{\mathcal{C}}(\phi) \,.$$

Proof:
The lower bound follows by Inequality 6 and the definition $Pr_\mu^{\mathcal{C}}(\phi) = \sum_{s\in Supp(\mu)} \mu(s) Pr_s^{\mathcal{C}}(\phi)$. The upper bound can be established in the same way. A one-norm error estimation, as in the proof of Theorem 3.3, is needed. The proof can be obtained from the proof for Theorem 3.3 in an obvious way. □

Concerning *complexity* of the truncation algorithm, consider a CTMC $\mathcal{C}$ with initial distribution μ, a formula Φ and truncation depth $k = \text{TRUNCATION}(\mathcal{C}, \mu, \Phi)$. The complexity of constructing the truncation for Φ is both linear in $\mathcal{C}|_k$ and $|\Phi|$. The number of states in the truncation could, however, be exponential in k. For probabilistic until formula $\mathcal{P}_{\trianglelefteq p}(\Phi_1\, \mathcal{U}^{[t,t']}\, \Phi_2)$, the complexity of the second phase (the transient analysis) is $\mathcal{O}(|M| \cdot qt')$ [21, 6] where $|M|$ is the number of transitions, q is the uniformization rate in $\mathcal{C}|_k$.

A few remarks on the role of approximation in truncation-based model checking:

- Truncation-based model checking is based on numerical approximations in both of its phases. The truncated model yields an approximation $Pr_{\mu}^{\mathcal{C}|_k}(\phi)$ of $Pr_{\mu}^{\mathcal{C}}(\phi)$ up to some ε'. The truncated model is, in turn, subject to the numerical CSL model-checking algorithm for finite CTMCs [6] (cf. Section 2.2), where the result $Pr_{\mu}^{\mathcal{C}|_k}(\phi)$ is accurate up to some precision ε''. To guarantee a global error bound of ε, one can set precision $\varepsilon' = \varepsilon'' = \frac{\varepsilon}{2}$.

- A difference of truncation-based model checking to finite-state CSL model checking is that the satisfaction sets $Sat(\Phi)$ can in general not be computed, since they are potentially infinite. Instead, $\text{TRUNC}(\mathcal{C}, S_0, \Phi)$ computes the truncation depth k for Φ, such that to check whether layer-zero states in S_0 satisfy Φ can be done in the truncation $\mathcal{C}|_k$. Satisfaction sets in the truncated model are of course finite.

3.4. Extensions

By equipping states and/or transitions with rewards (or costs), one can turn a CTMC into a so-called continuous-time Markov reward model (MRM). Baier *et. al.* [5] introduced a logic CSRL, an extension of CSL, for CTMCs with state rewards. This extends the until path operator $\Phi\, \mathcal{U}_J^I\, \Psi$ with an additional interval J representing a bound for the accumulated rewards. Besides performability [23] and other important dependability measures, Cloth used CSRL to express survivability properties, which refer to the ability of a system to recover from disastrous circumstances. Effective CSRL model checking is restricted to cases where either time is unbounded, but rewards are bounded [5], or intervals of the form $I = [0, t]$ and $J = [0, r]$ occur. In the first case, the duality of time and reward can be used to apply our results directly on a model where time and rewards are swapped [8] and hence time is bounded. In the second case, the time bound t makes the property obviously bounded, implying that only a finite truncation needs to be considered for model checking. As a consequence, our techniques for CSL model checking can be extended to handle the relevant fragment of CSRL, provided we are dealing with reward-bounded models, i.e., models where the supremum of state and transition rewards considered are finite. The induced error of the computed rewards in the truncation is scaled up by the maximal reward. This also holds for other reward properties [5] such as instantaneous rewards and cumulative rewards. The truncation idea can also be applied to discrete-time models for step bounded properties. Here truncation depth can then be calculated in a similar, rather easier, way.

4. Experimental Results

Using several case studies, we assess the effectiveness of truncation-based model checking and compare the different error-estimation techniques. The experimental evaluation is an extended and revised version of [14]. As in Subsection 3, we consider three different strategies for error-estimation, namely the Uniform chain based, Layered chain based, and the finite state projection (FSP) method.

We have implemented `INFAMY` [14], a model checker for infinite Markov (reward) models. `INFAMY` supports probabilistic programs in a guarded command language with unbounded integers. This language is a probabilistic variant of reactive modules [2] used in the popular finite-state model checker `PRISM` [16]. We use an extension of the language [33] admitting models with unbounded integer and real variables. This way infinite-state models can be specified.

Like the `SPIN` model checker [17], `INFAMY` is an analysis generator: given a model checking problem, `INFAMY` produces a dedicated C program that is compiled and carries out the analysis. We thus benefit from compiler optimizations and native execution, which, in particular, speeds up the generation of the truncated CTMC. To represent the truncated CTMC and perform CSL model checking [21, 6], `INFAMY` maintains an efficient sparse-matrix data structure that admits incremental extension as the truncation depth grows. In [35], we employed a proof-of-concept implementation with performance bottlenecks in the generation of truncated models and a loose integration with the CSL model checker `MRMC` [19]. In comparison, `INFAMY` is a complete re-design, works as a standalone tool, and is significantly faster.

The results were obtained on a Linux machine with an AMD Athlon$^{\mathrm{TM}}$ XP 2600+ processor at 2 GHz equipped with 2 GB of RAM. Where convenient, we denote multiples of thousand by "K".

Random Walk. We consider a discrete-space, continuous-time random walk model, in which a walker starts from the initial position 0, and changes position with rate λ. The field directly left to the current position is chosen with a probability of p, and the field to the right with a probability of $1-p$. We consider two properties: (1) an instantaneous reward property: the expected distance from the starting point at a given point in time, and (2) a probabilistic reachability property: the probability that the walker moves at least 10 fields to the right within a certain time bound t, expressed by $\mathcal{P}_{=?}(\Diamond^{\leq t} m \geq 10)$ where m denotes the position of the walker. The corresponding results for $p = 0.25$, and $\lambda = 1$ are given in Table 1, where the analysis time given is only for analyzing the first property after state-space exploration. The time with respect to the second property is in the same order as the first one. As apparent from the data, the size of the model is approximately proportional to the depth and the time bound t. The Layered-chain and FSP configurations achieve consistently smaller truncation depths than Uniform. Due to the simple structure of the model, however, FSP turns out to be slower than the Uniform and Layered chain configurations. This is because, for the FSP configuration, the transient analysis for state exploration consumes a proportionally high amount of time, especially for large time bounds t.

Table 1. Random walk. "t" is the time bound, "dep" the truncation depth. Column "time (s)" gives both the time for constructing the truncation and model checking time in seconds separated by a slash. n is the number of states. A precision of $\varepsilon = 10^{-6}$ is used for this case study.

t	Uniform			Layered			FSP			exp.	prob.
	dep	time (s)	n	dep	time (s)	n	dep	time (s)	n		
50	223	0.9/0.0	447	71	0.9/0.0	143	62	0.9/0.0	125	2.50E+01	0.99
100	273	0.9/0.0	547	121	0.9/0.0	243	101	1.0/0.0	203	5.00E+01	1.00
1K	1.3K	0.9/0.1	2.5K	885	0.9/0.1	1.8K	654	16.3/0.0	1.3K	5.00E+02	1.00
5K	5.6K	0.9/1.8	11.2K	4.0K	1.2/1.2	8.1K	2.8K	1,462/0.7	5.7K	2.50E+03	1.00

Jackson Queuing Networks [18]. A Jackson Queuing Network (JQN) is a system consisting of a number of n interconnected queuing stations. A JQN with two queues is depicted in Figure 6. Jobs arrive from the environment with a negative exponential inter-arrival time and are distributed to station i with probability $r_{0,i}$. Each station is connected to a single server which handles the jobs with a service time given by a negative exponential distribution with rate μ_i. Jobs processed by the station of queue i leave the system with probability $r_{i,0}$ but are put back into queue j with probability $r_{i,j}$. JQNs have an infinite state space because the queues are unbounded. Initially all queues are empty. In this paper, we consider JQN models with $N = 3, 4, 5$ queues. The arrival rate for N queues is λ, which is then distributed to station j (with service rate $\mu_j := j$) with probability $\frac{\mu_j}{\sum_{i=1}^{N} \mu_i}$. The probability out of a service rate is then uniformly distributed. We compute the probability that, within $t = 10$ time units, a state is reached in which 4 or more jobs are in the first and 6 or more jobs are in the second queue.

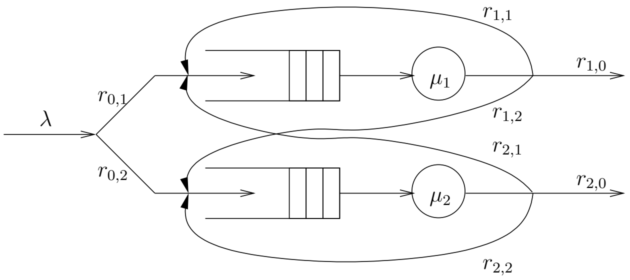

Figure 6. A JQN with two queues.

Table 2. Jackson Queuing Networks with N queues and arrival rate λ. A precision of $\varepsilon = 10^{-6}$ is used.

N/λ	Uniform			Layered			FSP			prob.
	dep	time	n	dep	time	n	dep	time	n	
3 / 2	193	8/19	1,236K	46	1/1	18K	37	4/0	10K	1.98E-01
4 / 2	-	-	-	46	3/11	230K	35	43/4	82K	1.20E-01
5 / 2	-	-	-	46	37/187	2,349K	34	392/50	576K	8.94E-02
3 / 3	203	10/23	1,436K	60	1/1	40K	51	14/1	25K	5.90E-01
4 / 3	-	-	-	60	7/27	635K	48	202/13	271K	4.42E-01
5 / 3	-	-	-	-	-	-	46	2,299/193	2,349K	3.71E-01
3 / 4	212	12/26	1,633K	74	1/2	73K	64	35/1	48K	8.38E-01
4 / 4	-	-	-	74	15/55	1,426K	61	665/29	677K	7.16E-01
3 / 5	223	13/31	1,898K	88	2/3	121K	77	76/2	82K	9.43E-01
4 / 5	-	-	-	88	30/101	2,794K	74	1,772/58	1,426K	8.65E-01

Results for different number of queues N and arrival rate λ are given in Table 2. Observe that, for fixed arrival rate λ, the depth is insensitive to the number of queues in Layered and Uniform configurations. The service rate increases with N, however, transitions corresponding to the service rate only lead back to states which are already explored, thus not contributing to the forward exit rates. For FSP, the depth is smaller than the Layered chain configuration. The number of states, and also the number of transitions, grow very fast with respect to the depth. Uniform chain cannot handle the cases $N = 4, 5$, as opposed to the other two configurations. While in the Layered chain configuration the dominating part is the model checking time in the truncation, the dominating part of the FSP configuration is the state exploration part. For $N = 5$ and $\lambda = 3$, it is the only configuration that still works.

If the truncation grows very fast with respect to the depth, an approach that combines the two configurations might be the only option for large N and λ. Such an approach is left for future work.

Quasi-Birth-Death Processes. In [20], a case study is considered that describes a system consisting of a fixed number of m processors and an infinite queue for storing job requests. The processing speed of a processor is described by the rate γ, while λ describes the incoming rate of new jobs. If a new job arrives while at least one processor is idle, the job will be processed directly. Otherwise, it will be put into a waiting queue. If there are idle processors and the waiting queue is non-empty, a job will be taken from the queue and processed immediately. To model this spontaneous transition, a rate $\mu \gg \lambda$ is used. The stochastic Petri net (SPN) used in [20] is depicted in Figure 7 for the case $m = 3$. Tokens in place p_1 represent the number of idle processors, place p_2 describes the number of busy processors and place p_3 gives the number of jobs in the queue. Transition t_1 models the case of an incoming job given that at least one processors is idle, whereas t_4 describes the case in which all processors are busy, thus the job is put into the queue. Transition t_2 represents the successful termination of a job. Finally, t_3 is the spontaneous transition in case there are idle processors and the queue is non-empty. We consider the probability that, given that all processors are busy and there are no jobs in the queue, within t time units a state will be reached in which all processors are idle and the queue is empty. We can compute the probability by setting $p_1 = 0, p_2 = 3, p_3 = 0$ as the initial state and checking the formula $\mathcal{P}_{=?}(\Diamond^{\leq t} p_1 = 3 \wedge p_3 = 0)$.

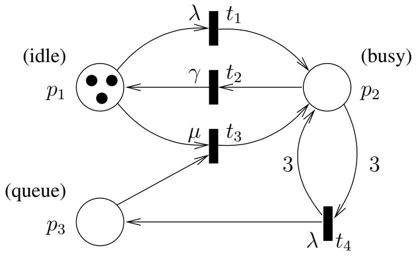

Figure 7. Quasi-Birth-Death Process.

Table 3. Quasi-Birth-Death Process. A precision of $\varepsilon = 10^{-6}$ is used.

λ	Uniform			Layered			FSP exponential			prob.
	depth	time (s)	n	depth	time (s)	n	depth	time (s)	n	
40	609	0.9/1.2	2,434	533	0.9/1.1	2,130	512	2.8/1.3	2,046	4.22E-04
60	846	0.9/1.6	3,382	754	0.9/1.5	3,014	1,024	4.9/2.1	4,094	1.25E-04
80	1,077	0.9/2.1	4,306	971	0.9/1.9	3,882	1,024	4.9/2.0	4,094	5.27E-05
100	1,305	0.9/2.5	5,218	1,187	0.9/2.4	4,746	2,048	9.3/4.0	8,190	2.70E-05

Results for different λ are given in Table 3 for $\mu = 1000, m = 3, t = 10$. The uniformization rate of the underlying CTMC is $1000 + \lambda$. With the increase of λ, the depth grows approximately linearly. Thus, the performance of the FSP configuration suffers from the high cost of repeated transient analysis, and, therefore, does not terminate within two hours. As observed by Munsky [24], this problem can be alleviated by adding more than one layer at each step. We consider a variant in which we double the number of layers we add per step, thus computing an error estimate every $1, 2, 4, 8, \ldots$ layers. We call this configuration *FSP exponential*. In contrary to FSP, FSP exponential configuration works reasonable. It is however not competitive with the Uniform or Layered configuration in time and memory.

Protein Synthesis [13]. We analyze an SPN model of protein synthesis, as depicted in Figure 8. In biological cells, each protein is encoded by a certain gene. If the gene is active, the corresponding protein will be synthesized. Also, proteins may degenerate and thus disappear after a time. Activation and deactivation of genes, protein synthesis (in case of active gene) as well as protein degeneration are modeled by stochastic rates. In the model, the place p_1 corresponds to an inactive gene encoding the protein, p_2 corresponds to an active gene, and p_3 gives the numbers of existing proteins. The transition t_1 deactivates the gene with rate μ, while t_2 activates it with rate λ. If the gene is active, t_3 can produce new proteins with rate ν. Each individual protein degenerates with rate δ, which is modeled by the transition t_4. We consider the property that, within time t but later than 10 time units, a state is reached, in which 20 or more proteins exist and the gene is inactive: $\mathcal{P}_{=?}(\Diamond^{[10,t]} p_3 \geq 20 \wedge \mathit{inactive})$ where $\mathit{inactive}$ is an atomic proposition representing inactive genes.

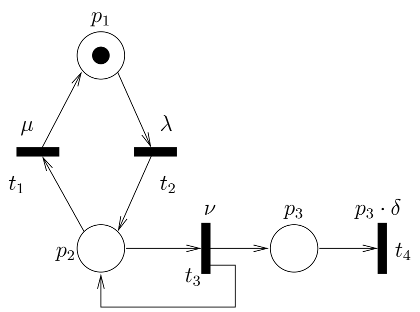

Figure 8. Protein Synthesis.

Table 4. Protein synthesis. A precision of $\varepsilon = 10^{-6}$ is used.

t	Uniform			Layered			FSP			prob.
	depth	time (s)	n	depth	time (s)	n	depth	time (s)	n	
300	2,330	0.9/2.0	4,659	2,047	0.9/1.7	4,093	34	1.0/0.0	67	2.07E-02
500	3,636	0.9/7.8	7,271	3,308	1.0/6.4	6,615	35	1.0/0.0	69	4.39E-02
1000	6,830	0.9/74.5	13,659	6,420	1.4/68.0	12,839	36	1.2/0.0	71	9.99E-02
2000	13,103	1.0/567.0	26,205	12,577	2.7/502.2	25,153	37	1.4/0.0	73	2.02E-01

The results for $\lambda = 1, \mu = 5, \nu = 1, \delta = 0.02$ are presented in Table 4. In this case study, the sum of the out-going rates of a state mainly depends on $p_3 \cdot \delta$. Thus the model is rate-unbounded. On newly explored states where p_3 is higher, the uniformization rate could also be increased. Thus, in [35], only time bounds $t \leq 45$ can be handled. Since in the Uniform and Layered chain configurations only forward exit rates are considered, very large time bounds can now be handled. Remarkably, the depth grows very slowly with respect to time if FSP is used. Even though global rates are unbounded, this is of no consequence, since most of the rates lead back to states at lower depth. This also means that the probability of going out is very small. Both Uniform and Layered chain configuration do not exploit this. Thus the resulting truncation depth is significantly higher.

Grid-World Robot [34]. We consider a grid world in which a robot moves from the lower left to the upper right corner sending signals to a station. A janitor moves around and may block the robot for some time. In contrast to [34], the janitor can leave the $N \times N$ field of the robot, which leads to an infinite state space. We check the property *"The probability that the robot reaches the upper right corner of the field within 100 time units while maintaining a probability of at least 0.5 of communicating periodically (every seven minutes) with a base station, is at least 0.9"*, which can be expressed via the CSL formula $\mathcal{P}_{\geq 0.9}\left(\left(\mathcal{P}_{\geq 0.5} \Diamond^{\leq 7} \text{communicate}\right) \mathcal{U}^{\leq 100} \text{ upperright}\right)$. In Table 5, we give results for different N. Using the FSP method is advantageous, as each layer of the model contains quite a large number of states, and FSP only needs to explore a fraction of the layers compared to the other methods. As seen from the "prob." column, for $N = 2, 3$ the property holds, while, for $N = 4$, it does not. The reason for the decreasing probability is that the distance the robot has to travel increases with n. Thus the probability decreases that the robot will complete its travel in time. In this case study, an implicit representation of the transitions of the model is used to avoid running out of memory. During the analysis, each time we visit a state, we compute its outgoing transitions on the fly. Afterwards, these transitions are removed and the next state is considered. The trade-off of this representation is the increasing runtime.

Table 5. Grid Worlds Robot.

N	Uniform			Layered			FSP			prob.
	depth	time (s)	n	depth	time (s)	n	depth	time (s)	n	
2	912	35/16K	9,960K	576	14/6K	3,969K	31	59/12	11K	9.99E-01
3	912	58/35K	16,563K	576	26/13K	6,591K	31	91/22	17K	9.01E-01
4	912	79/51K	23,137K	576	34/19K	9,194K	32	126/32	24K	8.95E-01

Workstation cluster. While the previously discussed case studies lead to infinite CTMCs, we now assess our method on finite-state models. To compare against a model checker which does not employ truncation, we use `PRISM` [16] in its current version 3.2. We consider the *dependability of a fault-tolerant workstation cluster* [15]. Figure 9 depicts a dependable cluster of workstations. The cluster consists of two sub-clusters, which, in turn, contain N workstations connected via a central switch. The two switches are connected via a backbone. Each component of the system can break down, and is then fixed by a single repair unit responsible for the entire system. Hereby, the quality of service (QoS) constraint $\mathit{Minimum}$ requires at least k ($k < N$) workstations to be operational where $k = \lfloor \frac{3}{4}N \rfloor$. Workstations have to be connected via switches. If in each sub-cluster the number of operational workstations is smaller than k the backbone is required to be operational to provide the required service. We consider two properties:

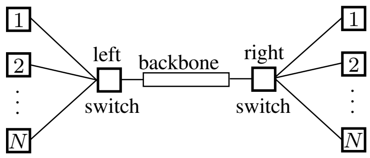

Figure 9. A dependable cluster of workstations.

A) $\mathcal{P}_{=?}(\Diamond^{[0,1]} \neg \mathit{Minimum})$: the probability that the QoS drops below minimum quality within one time unit.

B) the expected number of repairs[3] by time point one.

For both properties, we compare `PRISM` with `INFAMY`. The results are given in Table 6. Results for `INFAMY` are given for the Layered chain and FSP configurations respectively. The Uniform chain configuration is omitted, as it is always dominated by the Layered chain configuration. `PRISM` implements three different engines: a sparse-matrix and two symbolic engines. We used the sparse-matrix engine as it was the fastest one.

In Table 6, the time columns “tm (s)” have the format $t_1/t_2/t_3$ where the first number t_1 denotes the time needed for model construction, t_2 for the time to check property 1 and t_3 that for property 2. While the number of states and transitions for `PRISM` increases dramatically with parameter N, `INFAMY` scales much better in both configurations. The reason is apparent from the column displaying the depth: the depth of the full model is approximately linear in N, while the depth needed is sub-linear in N.

[3]The corresponding property on the PRISM web-site is `R=?[C<=T]`.

Table 6. Cluster performance statistics (properties A and B). A precision of $\varepsilon = 10^{-12}$ is used.

N	PRISM			Layered			FSP			prob./exp.
	dep	tm (s)	n	dep	tm (s)	n	dep	tm (s)	n	
512	1029	12/197/952	9466K	105	2/5/5	193K	20	2/0/0	6K	5.96E-08/0.72
1024	2053	48/1132/-	37806K	109	2/5/5	208K	26	3/0/0	11K	6.01E-08/1.08
2048	-	-	-	116	2/6/6	236K	36	6/1/1	22K	6.07E-08/1.38
4096	-	-	-	129	3/7/8	293K	53	19/1/1	48K	6.11E-08/1.56

For $N \geq 2048$ and $N \geq 1024$ respectively, PRISM cannot model check these properties as it runs out of memory and crashes; this is denoted by –. As N increases, the truncation depth grows only slowly in INFAMY, and hence, only a small fraction of the state space needs to be explored in the truncated construction. Although the FSP configuration is not very effective in terms of the total time, thereby spending the largest part of the running time in the first phase, it shows a lot of potential in terms of depth and number of states.

In Table 7, we give additional performance measures for a model of $N = 512$ workstations for property 1 for larger time bounds. Results are given for PRISM (sparse engine), FSP and FSP exponential respectively. The state space explored by PRISM has depth 1029 and includes 9.4 million states. Up to $t = 20$, INFAMY with FSP is faster than PRISM. However, for larger time bounds, the model construction time dominates, as for each layer exploration the error estimate is recomputed. The FSP variant with exponential layer explorations is suitable for this case, and is consistently the fastest method for $t \leq 50$, as shown in the last column of the table.

Table 7. Cluster performance statistics for different time bounds. A precision of $\varepsilon = 10^{-12}$ is used.

t	PRISM	FSP			FSP exponential			prob.
	time (s)	depth	time (s)	n	depth	time (s)	n	
10.0	11.9/359.5	52	47.5/3.0	46K	64	6.8/4.7	71K	3.79E-06
20.0	11.7/593.8	80	318.1/13.5	111K	128	44.4/35.6	289K	1.01E-05
30.0	11.7/802.2	104	1,016.9/32.5	190K	128	62.5/50.2	289K	1.68E-05
50.0	11.7/1171.8	147	4,660.4/103.7	382K	256	404.9/323.0	1,167K	3.04E-05

5. Related Work

Most closely related is finite state projection (FSP) by Munsky and Khammash [24, 25, 26]. An extension of FSP to handle initial distributions μ with very large support is discussed in [25]. In this case, a smaller subset of $A \subset Supp(\mu)$ is taken as S_0 such that $\mu(A)$ is large enough. The error bound induced in the finite truncation then depends on $\mu(Supp(\mu) \setminus A)$. FSP is especially effective if, during the state exploration, large rates go back to states already explored, since the truncation depth for such models tends to be very small. However, the stopping criterion relies on an error estimation that becomes expensive if the truncation depth is large. Beside FSP, we have discussed two other error estimation approaches: the Uniform chain and the Layered chain, which have significantly lower computational cost but may lead to higher truncation depths. Overall the approaches are complementary: FSP tends to perform well for shallow problems and the chain-based methods perform better for higher depths.

Also closely related is adaptive uniformization for CTMCs by van Moorsel and Sanders [32], which continuously recalibrates uniformization to perform well when exploring the state space on the fly. As discussed, our Layered chain configuration dominates adaptive uniformization in the sense that, for the same computational cost, Layered chains give a smaller truncation depth than adaptive uniformization, since only forward rates are considered. On the other hand, our approach in [35] is dominated by the Uniform chain configuration for the same reason, only forward rates are considered.

In [28, 27], algorithms for model checking CSL against infinite-state CTMCs of QBDs and JQNs are developed. They also considered steady-state properties. We remark that QBDs and JQNs are highly structured infinite CTMCs where the *universal* uniformization rate can still be determined by exploiting the structure. Our approach, however, does not compute the universal uniformization rate, thus with our approach, general time-bounded properties can be model checked over infinite CTMCs by analyzing the finite truncation.

Truncation has parallels to *bounded model checking* (BMC) [9] where bounded properties, such as "no error state is reachable within k steps", are checked. Technically, to check if an error state is reachable, BMC constructs a formula by unrolling the transition relation of the model up to depth k. However, the severity of an error is often independent of the depth at which it occurs. The depth bound in BMC is thus a concession to the limitations of the underlying SAT checking technology. In contrast, we consider CTMCs, where time-bounded properties arise naturally.

6. Conclusion and Future Work

In this paper, we have introduced time-bounded model checking for infinite CTMCs and infinite Markov reward models. We have evaluated our method on models from the biological and the queuing domain. The integration of different estimation methods has proven useful to arrive at efficient analysis techniques for models with different characteristics. Our novel chain-based error estimation methods show consistent improvements over related pre-existing methods: both in theory and practice. On the other hand, these methods have complementary strengths. Our experiments indicate that a promising avenue for future research might be a combination of finite state projection and a chain-based error estimation method. Further, due to its efficient implementation, we anticipate that `INFAMY` may be competitive as a general-purpose CSL model checker. We plan to evaluate `INFAMY` on a broader range of case studies.

Acknowlegment. We thank Michael Burger and Holger Lang (Fraunhofer ITWM Kaiserslautern) for helping us verify that the sequence in Subsection 3.2.2 is indeed increasing with limit $\frac{1}{2}$, and also for providing the relevant reference [1].

References

[1] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions*. 1964.

[2] R. Alur and T. A. Henzinger. Reactive Modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[3] W. J. Anderson. *Continuous-Time Markov Chains: An Applications-Oriented Approach*. Springer Verlag, 1991.

[4] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton. Model-Checking Continuous-Time Markov Chains. *ACM Trans. Comput. Log*, 1(1):162–170, 2000.

[5] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. On the Logical Characterisation of Performability Properties. In *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 780–792. Springer, 2000.

[6] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-Checking Algorithms for Continuous-Time Markov Chains. *IEEE Trans. Software Eng*, 29(6):524–541, 2003.

[7] C. Baier, J.-P. Katoen, H. Hermanns, and V. Wolf. Comparative Branching-Time Semantics for Markov Chains. *Information and computation*, 200(2):149–214, 2005.

[8] M. D. Beaudry. Performance-Related Reliability Measures for Computing Systems. *IEEE Trans. on Comp. Sys*, 27(6):540–547, 1978.

[9] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *TACAS*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 1999.

[10] E. A. Emerson and E. M. Clarke. Using Branching Time Temporal Logic to Synthesize Synchronization Skeletons. *Sci. Comput. Program*, 2(3):241–266, 1982.

[11] B. L. Fox and P. W. Glynn. Computing Poisson Probabilities. *Commun. ACM*, 31(4):440–445, 1988.

[12] W. K. Grassmann. Finding Transient Solutions in Markovian Event Systems Through Randomization. In *Numerical Solution of Markov Chains*, pages 357–371, 1991.

[13] P. J. E. Gross and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proc. Natl. Acad. Sci.*, 95:6750–6755, 1998.

[14] E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. Infamy: An infinite-state markov model checker. In *CAV*, 2009 (to appear).

[15] B. R. Haverkort, H. Hermanns, and J.-P. Katoen. On the Use of Model Checking Techniques for Dependability Evaluation. In *SRDS*, pages 228–237, 2000.

[16] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In *TACAS*, volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer, 2006.

[17] G. J. Holzmann. The Model Checker SPIN. *IEEE Trans. Software Eng.*, 23(5):279–295, 1997.

[18] J. Jackson. Networks of Waiting Lines. *Operations Research*, 5:518–521, 1957.

[19] J.-P. Katoen, M. Khattri, and I. S. Zapreev. A Markov Reward Model Checker. In *QEST*, pages 243–244. IEEE Computer Society, 2005.

[20] J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-Valued Abstraction for Continuous-Time Markov Chains. In *CAV*, volume 4590 of *Lecture Notes in Computer Science*, pages 311–324. Springer, 2007.

[21] J.-P. Katoen, M. Z. Kwiatkowska, G. Norman, and D. Parker. Faster and Symbolic CTMC Model Checking. In *PAPM-PROBMIV*, volume 2165 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2001.

[22] M. Z. Kwiatkowska, G. Norman, and D. Parker. Stochastic Model Checking. In *SFM*, volume 4486 of *Lecture Notes in Computer Science*, pages 220–270. Springer, 2007.

[23] J. F. Meyer. Performability Evaluation: Where It Is and What Lies Ahead. *IEEE Int. Comp. Perf. and Dependability Symp*, pages 334–343, 1995.

[24] B. Munsky and M. Khammash. The Finite State Projection Algorithm for the Solution of the Chemical Master Equation. *Journal of Chemical Physics*, 124(044104), 2006.

[25] B. Munsky and M. Khammash. A Multiple Time Interval Finite State Projection Algorithm for the Solution of the Chemical Master Equation. *Journal of Computational Physics*, 2007.

[26] B. Munsky and M. Khammash. Computation of Switch Time Distributions in Stochastic Gene Regulatory Networks. In *American Control Conference*, pages 2761–2766, 2008.

[27] A. Remke and B. R. Haverkort. CSL Model Checking Algorithms for Infinite-State Structured Markov Chains. In *FORMATS*, volume 4763 of *Lecture Notes in Computer Science*, pages 336–351. Springer, 2007.

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

[29] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[30] N. M. van Dijk. On the finite horizon Bellman equation for controlled Markov jump models with unbounded characteristics. *Stochastic Proc. Appl*, 28:141–157, 1988.

[31] A. P. A. van Moorsel. *Performability Evaluation Concepts and Techniques*. PhD thesis, Universiteit Twente, 1993.

[32] A. P. A. van Moorsel and W. H. Sanders. Adaptive Uniformization. *Communications in Statistics - Stochastic Models*, 10(3):619–647, 1994.

[33] B. Wachter, L. Zhang, and H. Hermanns. Probabilistic Model Checking Modulo Theories. In *QEST*, pages 129–138. IEEE Computer Society, 2007.

[34] H. L. S. Younes. Ymer: A statistical model checker. In *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 429–433. Springer, 2005.

[35] L. Zhang, H. Hermanns, E. M. Hahn, and B. Wachter. Time-Bounded Model Checking of Infinite-State Continuous-Time Markov Chains. In *ACSD*, pages 98–107. IEEE, 2008.

A. Proofs

This subsection recapitulates the notions of simulation relations, which will be used to establish the correctness proof of our algorithm. We now recall the notion of weak simulation [7] on CTMCs. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC and let $R \subseteq S \times S$. In this section, we use $\mathbf{P}$ to refer to the transition matrix of its *embedded* DTMC, which is defined by: $\mathbf{P}(s, s') = \frac{\mathbf{R}(s,s')}{\mathbf{R}(s,S)}$ if $\mathbf{R}(s, S) > 0$, and 0 otherwise. For states $s_1, s_2 \in S$, we say that s_2 weakly simulates s_1 if they have the same labeling, and if their successor states can be grouped into sets U_i and V_i for $i = 1, 2$, satisfying certain conditions. We can view steps to V_i as *stutter* steps while steps to U_i are *visible* steps. With respect to the visible steps, it is required that there exist a weight function for the conditional distributions: $\mathbf{P}(s_1, \cdot)/K_1$ and $\mathbf{P}(s_2, \cdot)/K_2$ where K_i is the probability to perform a visible step from s_i. The stutter steps must respect the weak simulation relations: thus states in V_2 should weakly simulate s_1, and state s_2 should weakly simulate states in V_1.

Definition A.1. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC and let $R \subseteq S \times S$. The relation R is a weak simulation on $\mathcal{D}$ iff for all s_1, s_2 with $s_1\ R\ s_2$: there exist partitions $post(s_i) = U_i \cup V_i$ for $i = 1, 2$ such that:

1. (a) $v_1\ R\ s_2$ for all $v_1 \in V_1$, and (b) $s_1\ R\ v_2$ for all $v_2 \in V_2$

2. There exists a function $\Delta : S \times S \to [0,1]$ such that:
   (a) $\Delta(u_1, u_2) > 0$ implies $u_1 \in U_1, u_2 \in U_2$ and $u_1 \; R \; u_2$.
   (b) Let $K_i = \mathbf{P}(s_i, U_i)$ for $i = 1, 2$ denote the probability of moving to U_i from s_i. Then, $K_1 > 0$ and $K_2 > 0$ implies that:
$$K_1 \cdot \Delta(w, U_2) = \mathbf{P}(s_1, w) \qquad \forall w \in U_1$$
$$K_2 \cdot \Delta(U_1, w) = \mathbf{P}(s_2, w) \qquad \forall w \in U_2$$
3. The *rate condition*: $K_1 \cdot \mathbf{R}(s_1, S) \leq K_2 \cdot \mathbf{R}(s_2, S)$

We say that s_2 weakly simulates s_1 in $\mathcal{C}$, denoted $s_1 \precsim\!\!\approx s_2$, iff there exists a weak simulation R on $\mathcal{C}$ such that $s_1 \; R \; s_2$. We reason about simulation between states if these states are part of the same CTMC. Therefore, for two CTMCs $\mathcal{C}_1, \mathcal{C}_2$, let $\mathcal{C}_1 \oplus \mathcal{C}_2$, we consider the direct sum which is the smallest CTMC containing both C_1 and C_2. We say $s_2 \in S_2$ weakly simulates $s_1 \in S_1$ in their direct sum $\mathcal{C}_1 \oplus \mathcal{C}_2$, denoted by $s_1 \precsim\!\!\approx s_2$, if there exists a weak simulation relation $R \subseteq S_1 \times S_2$ in the direct sum.

A few remarks are in order. In the original definition of weak simulation [7], $U_1 \cap V_1$ may be non-empty. For the purpose of our proof, it is sufficient to assume that $U_1 \cap V_1 = \emptyset$. In our simplified definition, we even allow the pair $(s_1, s_2) \in R$ to have different labeling. Indeed, we ignore the labeling functions in the definition: the reason is that we are only interested to bound the probability of reaching the absorbing state. Now we show that we can relate $\mathcal{C}|_k$, $\mathcal{C}_{U,k}$ and $\mathcal{C}_{L,k}$ with weak simulations. We use the same notations as depicted in Figure 1, Figure 3 and Figure 4.

Lemma A.1. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC and let S_0 be the set of layer-zero states. Then, for all $k \in \mathbb{N}$, there exists a weak simulation relation $R \subseteq S|_k \times S_{U,k}$ in the direct sum $\mathcal{C}|_k \oplus \mathcal{C}_{U,k}$ with $S_0 \times \{u_0\} \subset R$.

Proof:
We define a relation $R \subseteq S|_k \times S_{U,k}$ as follows: $R = \{(s_i^j, u_x) \mid s_i^j \in S_i \wedge i \leq x \leq k\} \cup \{(\bot, \bot)\}$. We show that R is a weak simulation in the direct sum $\mathcal{C}|_k \oplus \mathcal{C}_{U,k}$. Let $(s_i^j, u_x) \in R$. We consider two cases:

- $i < x$: By definition of R, all successor states of s_i^j, i.e., the set $post(s_i^j)$, can be weakly simulated by R, thus we are done.
- $i = x$: For notational convenience, let $S_{k+1} = \{\bot\}$ and $u_{k+1} = \bot$. We partition the set $post(s_i^j) = U_1 \cup V_1$ as follows: $U_1 = post(s_i^j) \cap S_{i+1}$ and $V_1 = post(s_i^j) \setminus U_1$. With respect to u_x, we define $U_2 = \{u_{x+1}\}$ and $V_2 = \emptyset$. For states s_i^j, the transitions to V_1 are all stutter steps, i.e., the corresponding successor state can again be weakly simulated by u_i. The transitions to U_1 can be matched as follows. By definition, we have $\mathbf{R}(s_i^j, U_i) \leq q_k = \mathbf{R}_{U,k}(u_i, u_{i+1})$, thus the rate condition is satisfied trivially. Since u_i has only one successor state which weakly simulates all states in U_1, the weight-function condition also trivially holds.

It follows that the relation R is a weak simulation in the direct sum $\mathcal{C}|_k \oplus \mathcal{C}_{U,k}$. By definition of R, it holds trivially $S_0 \times \{u_0\} \subset R$. □

Lemma A.2. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC and let S_0 be the set of layer-zero states. Then, for all $k \in \mathbb{N}$:

1. there exists a weak simulation relation $R \subseteq S|_k \times S_{L,k}$ with $S_0 \times \{l_0\} \subset R$,

2. there exists a weak simulation relation $R \subseteq S_{L,k} \times S_{U,k}$ with $(l_0, u_0) \in R$.

Proof:
The proof of the first part follows in a very similar way as the proof of Lemma A.1. What to be taken care is the rate condition, which is also guaranteed for the layered chain: $\mathbf{R}(s_i^j, U_i) \leq f_k = \mathbf{R}_{L,k}(u_i, u_{i+1})$. For the second part, we take the relation $R = \{(l_i, u_j) \mid i \leq j \leq k\} \cup \{(\perp, \perp)\}$. Again, similar arguments as the proof of Lemma A.1 shows that R is a weak simulation relation. By definition it holds also $(l_0, u_0) \in R$. $\Box$

In Theorem 63 of paper [7] it is shown that if $s_1 \precsim\!\!\!\approx s_2$, then for all liveness PCTL formulas without next operators Φ, $s_1 \models \Phi$ implies that $s_2 \models \Phi$. We consider a liveness formula of interest: $\mathcal{P}_{\geq p}(\Diamond^{[0,t]} \perp)$, which is satisfied in state s if the probability of reaching $\perp$ within time t is greater than p. Combining Lemma A.1 and Lemma A.2, the following lemma is a direct consequence:

Lemma A.3. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC and let S_0 be the set of layer zero states. Then, for $s_0 \in S_0$ and all $k \in \mathbb{N}$, it holds that:

$$\varepsilon_{\mathcal{C},s_0,k} = Pr_{s_0}^{\mathcal{C}|_k}(\Diamond^{[0,t]} \perp) \leq \delta_{L,k} = Pr_{l_0}^{\mathcal{C}_{L,k}}(\Diamond^{[0,t]} \perp) \leq \delta_{U,k} = Pr_{u_0}^{\mathcal{C}_{U,k}}(\Diamond^{[0,t]} \perp) \tag{7}$$

For a set of states A recall $\mathcal{C}[A]$ denotes the CTMC obtained via making A-states absorbing. In the next lemma we show with weak simulation that the probability of reaching the state $\perp$ in the k-truncation of $\mathcal{C}$ is larger than the probability of reaching $\perp$ in the k-truncation of $\mathcal{C}[A]$. Observe that $\mathcal{C}[A]|_k$ is the same as $\mathcal{C}|_k[A]$.

Lemma A.4. Let $\mathcal{C} = (S, \mathbf{R}, L)$ be a CTMC and let S_0 be the set of layer zero states. Let $A \subseteq S$ be a set of states. Then, for $s_0 \in S_0$ and all $k \in \mathbb{N}$, it holds that:

$$\varepsilon_{\mathcal{C}[A],s_0,k} = Pr_{s_0}^{\mathcal{C}|_k[A]}(\Diamond^{[0,t]} \perp) \leq \varepsilon_{\mathcal{C},s_0,k} = Pr_{s_0}^{\mathcal{C}|_k}(\Diamond^{[0,t]} \perp) \tag{8}$$

Proof:
We use a superscript A over state (or set of states) to indicate state (or set of states) in $\mathcal{C}[A]$. The proof follows via a weak simulation relation R in the direct sum $\mathcal{C}|_k[A] \oplus \mathcal{C}|_k$ with $S_0^A \times S_0 \subset R$. The relation R is given by: $R = \{(s^A, s) \mid s \in S\}$ which relates each state s^A in $\mathcal{C}|_k[A]$ to the corresponding state s in $\mathcal{C}|_k$. All states with s^A with $s \in A$ are made absorbing, thus need not to be matched (via setting $U_1 = V_1 = \emptyset$). It is routine to verify that R is a weak simulation in the direct sum $\mathcal{C}|_k[A] \oplus \mathcal{C}|_k$. $\Box$