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

Lijun Zhang, Holger Hermanns, E. Moritz Hahn, Björn Wachter
Department of Computer Science, Saarland University, Saarbrücken, Germany
{zhang,hermanns,emh,bwachter}@cs.uni-sb.de

Abstract

The design of complex concurrent systems often involves intricate performance and dependability considerations. Continuous-time Markov chains (CTMCs) are widely used models for concurrent system designs making it possible to model check such properties. In this paper, we focus on probabilistic timing properties of infinite-state CTMCs, expressible in continuous stochastic logic (CSL). Such properties comprise important dependability measures, such as timed probabilistic reachability, performability, survivability, and various availability measures like instantaneous availabilities, conditional instantaneous availabilities and interval availabilities. Conventional model checkers explore the given model exhaustively which is not always possible either due to state explosion or because the model is infinite. This paper presents a method that only explores the infinite (or prohibitively large) model up to a finite depth, with the depth bound being computed on-the-fly. We provide experimental evidence showing that our method is effective.

1 Introduction

Continuous-time Markov chains (CTMCs) [23], together with their extensions with rewards, are popular means to model performance and dependability of computing systems and the behavior of biological systems. In the context of CTMCs, properties of interest can be specified using continuous stochastic logic (CSL) [1, 3], which is a branching-time temporal logic inspired by CTL [7]. In CSL, the until operator is equipped with a time interval to capture probabilistic timing properties. CSL allows one to quantify the probability of paths that satisfy a certain (e.g. nested until-) path formula.

[*]This work is supported by the NWO-DFG bilateral project VOSS and by the DFG as part of the Transregional Collaborative Research Center SFB/TR 14 AVACS

We consider the model checking problem for CSL over *infinite* CTMCs. For highly structured infinite CTMCs, this has been already studied in [22, 21]. We consider arbitrary infinite CTMCs, including rates unbounded CTMCs. We focus on the CSL [3] formulas where we do not admit steady-state operator. The resulting logic can express (possibly nested) probabilistic timing properties such as: *"is the probability to reach Ψ-states along Φ-states within time interval [6.5,8.5] 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 formulae. They can express many performance measures, including timed probabilistic reachability, various availability measures like instantaneous availabilities, conditional instantaneous availabilities and interval availabilities [3]. We present model checking algorithms for such properties over infinite CTMCs. In practice, infiniteness occurs in the form of unbounded behavior, such as quantities of substances in biological/chemical models or unbounded queues in queueing systems.

CSL model checking for finite CTMCs amounts to performing analyses of the transient (time-dependent) probability vector [18, 3], usually carried out via the *uniformization* technique [23]. Via uniformization, the transient probability can be expressed by a weighted infinite sum (of transient probabilities computed in a finite discrete-time Markov chain). The weights are given by a jump process which is Poisson distributed. This infinite sum is in practice *truncated* up to some pre-specified accuracy. For a given accuracy, the truncation-point can be computed using the Fox-Glynn algorithm [8].

For infinite CTMCs, a variation of uniformization, called *dynamic* uniformization [10], has been introduced, and further developed into *adaptive* uniformization [26]. The basic idea is to truncate not only the computation of the infinite sum, but also the matrix that represents the system during its construction. This idea has the same flavor as the principle of Bounded Model Checking [5] (BMC). For dynamic uniformization, one has to *guess* a satisfactory uniformization parameter beforehand. As indicated in [25], a too small parameter means that the transition matrix is not a stochastic matrix, while a larger-than-necessary value makes the algorithm inefficient. Adaptive uniformization alleviates this, which carries out the uniformization-on-the-fly without a priori knowledge of the uniformization rate. In general, the truncation-point is smaller than the one obtained via dynamic uniformization. But the price to pay is that the jump process is no longer Poisson and hence complicates the computation.

In this paper, we lift the principal idea of truncated construction to the CSL model checking context, based on the work of Grassmann. We present an effective method to compute the optimal dynamic uniformization parameter, if it exists, in an on-the-fly manner. This enables a truncated construction of an otherwise infinite (or very large) CTMC, and the computation of transient probability up to the prespecified accuracy, which is the basis for our CSL model-checking routine. We also consider Markov reward models (MRMs) which extend CTMCs with state or transition rewards, allowing to express cost- or bonus-related properties [2, 19]. We show that our method can be generalized to model check timed reward properties against infinite-state MRMs. A typical reward property is *survivability* [6], which refers to the ability of a system to recover from disastrous circumstances.

The crucial characteristic of our method is that it applies to arbitrarily structured (finite or infinite) CTMC models (unlike [22, 21]) and that it avoids exploring that portion of the state space that is not needed for deciding the formula, just like in BMC. The difference to standard BMC is that the truncation point is determined from model parameters as they are explored, not from the evaluation of properties. The truncation point is however dependent on the time bounds appearing in the formula, and might get excessive for very large time bounds.

We have implemented a prototype that computes the truncation point on-the-fly. We have assessed the effectiveness of truncation on a number of infinite-state CTMCs, including a protein synthesis model, a Jackson queueing 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 that are not directly amenable to finite-state methods, in the finite-state case, significant speedups can be achieved over model checking without truncation.

Contributions. We introduce time-bounded model checking techniques for infinite CTMCs and infinite MRMs. By a set of case studies – including a protein synthesis model, a Jackson queueing network, and a job processing system – we provide experimental evidence showing that our method is effective. Further, we present a case study showing that our method is also beneficial for finite but prohibitively large CTMCs.

Outline of the paper. After recalling the established model checking algorithm for CSL over CTMCs in Section 2, we present the proposed extension to infinite CTMCs and infinite MRMs in Section 3. In Section 4, we report experimental results. Section 5 concludes. Proofs are given in the Appendix.

2 Preliminaries

We recall the definitions of continuous-time Markov chains (CTMCs), continuous stochastic logic (CSL), and the model checking algorithm for CSL over CTMCs. For more detail, we refer to [3].

2.1 Continuous-time Markov chains

If $f : X \times Y \to \mathbb{R}_{\geq 0}$ is a function over two countable domains, we let $f(x, A) = \sum_{y \in A} f(x, y)$ for all $x \in X$ and finite subset $A \subseteq Y$. Let AP denote a set of atomic propositions.

Definition 2.1 *A labelled continuous-time Markov chain (CTMC) is a tuple $\mathcal{C} = (S, \mathcal{I}, \mathbf{R}, L)$ where S is a countable set of states, $\mathcal{I} \subseteq S$ is a set of initial states, $\mathbf{R} : (S \times S) \to \mathbb{R}_{\geq 0}$ is a* rate *matrix, and $L : S \to 2^{AP}$ is a labelling function.*

In this paper, we consider only finitely branching transitions: $|\{s' \mid \mathbf{R}(s, s') > 0\}| < \infty$ for all $s \in S$. 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[1]. A state s is called *absorbing* if $\mathbf{R}(s, S) = 0$. The labelling 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', denoted by $s \to s'$. For $s \in S$, let $Succ(s) = \{s' \mid s \to s'\}$ denote the set of states directly reachable from s. For $A \subseteq S$, let $Succ(A) = \cup_{s \in A} Succ(s)$.

The transition probabilities are exponentially distributed over time. If $s \to s'$ is the only transition starting from s, the probability that the transition $s \to s'$ can be triggered within time t is $1 - e^{-\mathbf{R}(s,s')t}$. Furthermore, if $s \to s'$ 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 $1 - e^{-\mathbf{R}(s,S)t}$. The probability of taking a particular transition $s \to s'$ from s within time t is $\frac{\mathbf{R}(s,s')}{\mathbf{R}(s,S)} \left(1 - e^{-\mathbf{R}(s,S)t}\right)$, and in this case we say that $s \to s'$ *wins the race*.

[1]In this paper we consider only rate unbounded CTMCs which do not explode [24, 25]. Roughly speaking, a CTMC does not explode implies that in finite time with probability one only a finite number of states can be reached.

Paths and probabilistic measure. An (infinite) path is an infinite sequence $\sigma = s_1 t_1 s_2 t_2 \ldots$ satisfying $s_i \rightarrow s_{i+1}$, 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, $\delta(\sigma, i) = t_i$ 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 $Path^{\mathcal{C}}_{\infty}$ denote the set of all paths, and $Path^{\mathcal{C}}_{\infty}(s)$ denote the set of all paths starting from s. For state $s \in S$, a probability measure, denoted by $Pr^{\mathcal{C}}_s$, on the set $Path^{\mathcal{C}}_{\infty}(s)$ can be defined. We omit the subscript $\mathcal{C}$ if it is clear from context.

A finite path is a finite sequence $\sigma = s_1 t_1 s_2 t_2 \ldots s_k$ for $k \geq 0$ satisfying $s_i \rightarrow s_{i+1}$ 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 $Path_f$ denote the set of all finite paths.

2.2 The logic CSL

The logic we consider is CSL [3] without steady-state operators. Let $I = [a, b]$ be an interval with $a, b \in \mathbb{R}_{\geq 0}$ and $a \leq b$. The syntax of CSL is given by:

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

where $p \in [0, 1]$ and $\trianglelefteq \in \{\leq, <, >, \geq\}$. Let $\sigma \in Path_{\infty}$, the semantics of the path formula is defined as follows:

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

The semantics of the state formulae $a, \neg\Phi, \Phi_1 \wedge \Phi_2$ is defined in the same way as for CTL: $s \models a$ iff $a \in L(s)$, $s \models \neg\Phi$ iff $s \not\models \Phi$, $s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$. 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, $s \models \mathcal{P}_{\trianglelefteq p}(\phi)$ iff $Pr_s(\phi) \trianglelefteq p$ where $Pr_s(\phi)$ denotes the probability $Pr_s\{\sigma \in Path_{\infty}(s) \mid \sigma \models \phi\}$. The model $\mathcal{C}$ satisfies a formula if all initial states do: $\mathcal{C} \models \mathcal{P}_{\trianglelefteq p}(\phi)$ iff $Pr_{s_0}(\phi) \trianglelefteq p$ for all $s_0 \in \mathcal{I}$.

Dependability measures. We give several important dependability measures [3] that can be expressed in CSL. First, 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 be a set of goal states and at_B denote the atomic proposition characterizing that the system is in a B-state. Then, $\mathcal{P}_{\trianglelefteq p}(\Diamond^I at_B)$ expresses that the probability of reaching a B-states 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 $\mathcal{P}_{\trianglelefteq p}(\Phi\, \mathcal{U}^{[t,t]}\, up)$.

2.3 Model checking CSL on finite CTMCs

For a CTMC $\mathcal{C}$ and a CSL state formula Φ, we recall how to check whether $\mathcal{C} \models \Phi$. Throughout this subsection we assume that the set of states S is finite. The basic model checking strategy is as for CTL [7]. For every state sub-formula Ψ of Φ, we recursively compute the set $Sat(\Psi)$ satisfying the state formula Ψ. 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 the same as in [3] 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 and uniformization. For a CTMC $\mathcal{C} = (S, \mathcal{I}, \mathbf{R}, L)$, a *uniformization rate* $q \in \mathbb{R}_{\geq 0}$ is any value satisfying $q \geq \max_{s \in S} \mathbf{R}(s, S)$. 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 discrete-time Markov chain (DTMC). This (so-called) uniformized DTMC $uni(\mathcal{C}) = (S, \mathcal{I}, \mathbf{P}, L)$ can be obtained from $\mathcal{C}$ as follows: $\mathbf{P}(s, s')$, which describes the one-step probability to move from s to s', equals $\frac{\mathbf{R}(s,s')}{q}$ if $s \neq s'$ and $1 - \frac{\mathbf{R}(s, S\setminus\{s\})}{q}$ otherwise. We observe that, for an absorbing state s, $\mathbf{P}(s, s')$ equals 1 if $s = s'$, and 0 otherwise. Given $\mathcal{C}$, we will directly use $\mathbf{P}$ to refer to the transition matrix in its uniformized DTMC $uni(\mathcal{C})$.

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. If $t = 0$, we have $\vec{\pi}(s, 0)(s') = 1$ if $s = s'$ and 0 otherwise. We recall the *uniformization* algorithm [23] for the computation of transient probabilities. Let $t > 0$, then:

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

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.

Time-bounded until. Let $\mathcal{C} = (S, \mathcal{I}, \mathbf{R}, L)$ 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 it works, we consider first the simple case $I = [0, t]$. We let $\mathcal{C}[\Psi]$ denote the CTMC obtained by making Ψ-states absorbing. More precisely, $\mathcal{C}[\Psi] = (S, \mathcal{I}, \mathbf{R}^{\mathcal{C}[\Psi]}, L)$ where $\mathbf{R}^{\mathcal{C}[\Psi]}(s, s')$ equals $\mathbf{R}(s, s')$ if $s \not\models \Psi$, equals 0 otherwise. As explained in [3], the transformation 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 $\Phi\, \mathcal{U}^I\, \Psi$. 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 in general $Pr_s^{\mathcal{C}}(\Phi\, \mathcal{U}^I\, \Psi)$ can be computed via transient analysis.

Theorem 2.2 ([18]) *For any $\mathcal{C}$, $Pr_s^{\mathcal{C}}(\Phi\, \mathcal{U}^I\, \Psi)$ equals*

1. $\sum_{s' \models \Psi} \vec{\pi}^{\mathcal{C}[\neg\Phi \vee \Psi]}(s, t)(s')$ *if* $I = [0, t]$,
2. $\sum_{s' \models \Psi} \vec{\pi}^{\mathcal{C}[\neg\Phi \wedge \neg\Psi]}(s, t)(s')$ *if* $I = [t, t]$,
3. $\sum_{s' \models \Phi} \sum_{s'' \models \Psi} \vec{\pi}^{\mathcal{C}[\neg\Phi]}(s, t)(s') \cdot \vec{\pi}^{\mathcal{C}[\neg\Phi \vee \Psi]}(s', t' - t)(s'')$ *otherwise, i.e., if* $I = [t, t']$ *with* $0 < t < t'$.

By Equation 1, the transient probability is expressed by an infinite sum of the product of Poisson probability and the transient probability in the uniformized DTMC. Using the Fox-Glynn algorithm [8], the Poisson probabilities $\varphi(i, qt)$ can be computed. For any given accuracy ε, we want to compute $\vec{\pi}(s, t)$ up to ε. As explained in [9, 23], the number of terms to be taken out of Equation 1 is the minimal number $k(qt)$ satisfying: $\sum_{i=0}^{k(qt)} e^{-qt} \frac{(qt)^i}{i!} \geq 1 - \varepsilon$. We fix now an arbitrary accuracy, for example $\varepsilon = 10^{-6}$. If the product qt is large, $k(qt)$ is in the order of $\mathcal{O}(qt)$ [8, 3]. The complexity [3] of the above method is $\mathcal{O}(M \cdot k(qt'))$ where M denotes the number of transitions and $t' = \sup I$.

3 Model checking based on truncations

In this section, we discuss how to model check CSL formulae against infinite CTMCs. To this end, our goal is to operate on a finite truncation instead of the original infinite CTMC. From the previous section, we know that only the first $k(qt)$-terms of Equation 1 need to be considered for computing transient probabilities. We now consider those states that can not be reached within at most $k(qt)$ transitions from any initial state. For such a state s, we have $\vec{\pi}(s_0, i)(s) = 0$ for all $s_0 \in \mathcal{I}$ and $i = 0, 1, \ldots, k(qt)$. Because of the Poisson probability, the probability of reaching s with more than $k(qt)$ is neglected. This suggests that for computing the transient probabilities up to accuracy ε we can truncate the model at depth $k(qt)$. We will follow this intuition and discuss how to compute the truncation point for arbitrary CSL formulas.

Let $\mathcal{C} = (S, \mathcal{I}, \mathbf{R}, L)$ be a CTMC. First, we introduce the notion of *depth* which corresponds to the minimal distance between two states.

Definition 3.1 *For a given CTMC $\mathcal{C}$, the* depth *function $d : S \times S \to \mathbb{N}$ is defined by $d(s, s') = \min\{len(\sigma) \mid \sigma \in Path_f \wedge first(\sigma) = s \wedge last(\sigma) = s'\}$.*

We also write $d_s(s')$ to denote $d(s, s')$. Intuitively, $d_s(s')$ corresponds to the minimal length of any finite path starting from s and ending in s'. We let $d_I(s) = \min_{s_0 \in \mathcal{I}}\{d_{s_0}(s)\}$ denote the minimal depth starting from any initial state to s. For $k \in \mathbb{N}$, we define the k-truncated CTMC as follows:

Definition 3.2 *For CTMC $\mathcal{C}$ and $k \in \mathbb{N}$, we define the k-truncated CTMC of $\mathcal{C}$ by: $\mathcal{C}|_k = (S|_k, \mathcal{I}, \mathbf{R}_k, L_k)$ where $S|_k = \{s \in S \mid d_{\mathcal{I}}(s) \leq k\}$ and $\mathbf{R}_k, L_k$ are restricted to the truncated state space $S|_k$.*

Recall that we only consider CTMCs which are finitely branching. Not surprisingly, the k-truncated CTMC $\mathcal{C}|_k$ is always finite.

Now we define the truncation point function, which assigns a natural number $k(\mathcal{I}, \Phi)$ to a set of initial states $\mathcal{I}$ and a CSL state formula Φ, such that the $k(\mathcal{I}, \Phi)$-truncated model is sufficient to check the formula Φ, up to accuracy ε if Φ is of the form $\mathcal{P}_{\trianglelefteq p}(\phi)$. That is, $k(\mathcal{I}, \Phi)$ is a truncation point such that $Pr_s^{\mathcal{C}|_{k(\mathcal{I}, \Phi)}}(\phi)$ equals $Pr_s^{\mathcal{C}}(\phi)$ up to ε for all $s \in \mathcal{I}$. Note that we overload k for both truncation point of the Poisson probabilities, and truncation point for CSL formulae.

In Section 3.1 we discussed how to compute the transient probability at time t for infinite CTMCs. Based on that, we discuss in 3.2 how to compute the truncation point. In 3.3, the extension to Markov reward models will be discussed.

3.1 Dynamic uniformization for transient analysis

Let $\mathcal{C} = (S, \mathcal{I}, \mathbf{R}, L)$ be a infinite CTMC. We want to compute the transient probability at time t up to accuracy ε, starting at some initial state $s \in \mathcal{I}$. Since the model is infinite, we must truncate the model, otherwise it is impossible. Thus, we want to find a truncation point tr such that the transient probability can be computed in the finite truncation $\mathcal{C}|_{tr}$. This truncation point tr depends on the set of initial states $\mathcal{I}$, and on the time bound t, thus, we let $tr(\mathcal{I}, t)$ denote this number.

Grassmann [10] introduced the notion of *dynamic* uniformization for the transient analysis of CTMCs where one has to *guess* a satisfactory rate of the dynamic uniformization beforehand. A too small uniformization rate means that the transition matrix is not stochastic, while a too high uniformization rate makes the algorithm inefficient [25]. We are striving for an effective method to compute the adequate dynamic uniformization rate. Our concrete goal in this subsection is to determine the minimal rate for state s automatically, if it exists. The truncation point $tr(\mathcal{I}, t)$ should satisfy the following conditions:

1. for all $s' \in S$, $d_{\mathcal{I}}(s') \leq k(qt)$ implies that $k(\mathbf{R}(s, S) \cdot t) \leq tr(\mathcal{I}, t)$,

2. there exits $s \in S$ with $d_{\mathcal{I}}(s) \leq k(qt)$ such that $q = \mathbf{R}(s, S)$.

Such a q can be determined on-the-fly, i.e. during the exploration of the state space starting from the initial states. The idea is as follows. Let $q_0 = \max\{\mathbf{R}(s, S) \mid s \in \mathcal{I}\}$. Initially we let $k(q_0 t)$ serve as candidate truncation point derived by only looking at the set of initial states of $\mathcal{C}$. Now we explore the state space corresponding to the truncated CTMC $\mathcal{C}|_{k(q_0 t)}$ and check whether candidate q_0 happens to be the uniformization rate of $\mathcal{C}|_{k(q_0 t)}$ indeed. If yes, we are finished. If not, we have found $q_1 = \max\{\mathbf{R}(s, S) \mid s \in S|_{k(q_0 t)}\}$, and check whether $k(q_0 t) = k(q_1 t)$, in which case we are finished as well. Otherwise $k(q_0 t) < k(q_1 t)$ and we thus take the latter as our second candidate, and continue the state space exploration to determine the truncated CTMC $\mathcal{C}|_{k(q_1 t)}$. For rate bounded models, iterating this way must terminate at some point – because after every iteration $k(q_i t)$ is strictly greater than the previous truncation point $k(q_{i-1} t)$ and is bounded by $k(\max_{s \in S} \mathbf{R}(s, S) \cdot t)$, hence the iteration must terminate. However, this need not be true anymore for rate unbounded models, where the right truncation point could keep increasing because new explored states have ever higher exit rates. In any case, this strategy gives us a partial algorithm to determine the optimal dynamic uniformization rate on-the-fly, and terminates if such exists. We refer to the case study *Protein Synthesis* for further discussions.

3.2 Truncation Point for CSL formulae

Let $\mathcal{C} = (S, \mathcal{I}, \mathbf{R}, L)$ be a CTMC, and let Φ be a CSL state formula. We compute the truncation point $k(\mathcal{I}, \Phi)$ for CSL formula Φ recursively as follows.

- $\Phi = a$: we set $k(\mathcal{I}, \Phi) = 0$,

- $\Phi = \neg\Psi$: the truncation point $k(\mathcal{I}, \Phi)$ is the same as $k(\mathcal{I}, \Psi)$,

- $\Phi = \Phi_1 \wedge \Phi_2$: we take the maximum $k(\mathcal{I}, \Phi) = \max\{k(\mathcal{I}, \Phi_1), k(\mathcal{I}, \Phi_2)\}$ to guarantee the correctness of the result,

Consider the probabilistic operator $\Phi = \mathcal{P}_{\trianglelefteq p}(\phi)$. First, consider the next state path formula $\phi = \mathcal{X}^I \Psi$. It is sufficient to figure out the set of states satisfying Ψ which can be reached from $\mathcal{I}$ directly. This suggests to take $k(\mathcal{I}, \Phi) = 1 + k(Succ(\mathcal{I}), \Psi)$. Notably, $k(Succ(\mathcal{I}), \Psi)$ can be computed recursively in $\mathcal{C}$ with another set of initial states $Succ(\mathcal{I})$.

The most difficult part is the until path formula: $\Phi = \mathcal{P}_{\trianglelefteq p}(\Phi_1 \, \mathcal{U}^I \, \Phi_2)$ where $I = [t, t']$ with $t \leq t'$. The shape of the interval I forces us to distinguish three cases, two of which can be grouped together:

The cases $I = [0, t]$**.** Recall that to check whether $\mathcal{C} \models \Phi$, it is sufficient to calculate the probability $Pr_{s_0}(\Phi_1 \mathcal{U}^{[0,t]} \, \Phi_2)$ for all $s_0 \in \mathcal{I}$. By Theorem 2.2 it equals $\sum_{s' \models \Phi_2} \vec{\pi}^{\mathcal{C}[\neg\Phi_1 \vee \Phi_2]}(s_0, t)(s')$. We thus need to compute transient probability at time t, starting from state $s_0 \in \mathcal{I}$. For this, all states with depth smaller than $tr(\mathcal{I}, t)$ 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-state formulae Φ_1 and Φ_2. For this, we define:

$$k(\mathcal{I}, \Phi) = tr(\mathcal{I}, t) + \max\{k(\mathcal{I}', \Phi_1), k(\mathcal{I}', \Phi_2)\} \qquad (2)$$

where $\mathcal{I}' = \{s \in S \mid d_{\mathcal{I}}(s) \leq tr(\mathcal{I}, t)\}$ is the set of already explored states in $\mathcal{C}|_{tr(\mathcal{I},t)}$. Consider the truncation $\mathcal{C}|_{k(\mathcal{I},\Phi)}$. The sum of Equation 2 enables us to check whether states s with $d_{\mathcal{I}}(s) \leq tr(\mathcal{I}, t)$ satisfies subformulae Φ_1 and Φ_2 which is needed for Theorem 2.2.

The case $I = [t, t]$ can be analyzed in a very similar way as $I = [0, t]$, with also the same truncation point as Equation 2.

The case $I = [t, t']$ **with** $0 < t < t'$**.** For $s \in \mathcal{I}$, we shall compute the probability $Pr_s(\Phi_1 \, \mathcal{U}^{[t,t']} \, \Phi_2)$ using the third equation of Theorem 2.2. For the first transient probability $\vec{\pi}^{\mathcal{C}[\neg\Phi_1]}(s, t)(s')$ in $\mathcal{C}[\neg\Phi_1]$, the $tr(\mathcal{I}, t)$-truncated CTMC is required. Again $\mathcal{I}'$ denote the set of states in the truncation $\mathcal{C}|_{tr(\mathcal{I},t)}$. For every state $s' \models \Phi_1$ with $d_{\mathcal{I}}(s') \leq tr(\mathcal{I}, t)$, 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 $\mathcal{I}'$ as the set of initial states, all states with depth smaller or equal than $tr(\mathcal{I}', (t' - t))$ need to be explored. Let $\mathcal{I}''$ denote the set of states in the truncation $\mathcal{C}|_{k^*}$ where $k^* =$

$tr(\mathcal{I}, t) + tr(\mathcal{I}, t' - t)$. Moreover, as in the case $I = [0, t]$, to know whether the sub-formulae Φ_1 and Φ_2 are satisfied, we need explore additionally $\max\{tr(\mathcal{I}'', \Phi_1), tr(\mathcal{I}'', \Phi_2)\}$ starting from $\mathcal{I}''$. Altogether, we have that

$$k(\mathcal{I}, \Phi) = k^* + \max\{tr(\mathcal{I}'', \Phi_1), tr(\mathcal{I}'', \Phi_2)\}$$

For probabilistic operator the probability can be computed in the truncation with the given accuracy:

Lemma 3.3 *For any CTMC $\mathcal{C}$ and formula $\Phi = \mathcal{P}_{\unlhd p}(\phi)$, $Pr_s^{\mathcal{C}|_{tr(\mathcal{I},\Phi)}}(\phi)$ equals $Pr_s^{\mathcal{C}}(\phi)$ up to ε, for all $s \in \mathcal{I}$.*

The complexity of checking Φ is $\mathcal{O}(k(\mathcal{I}, \Phi) \cdot M|_{k(\mathcal{I},\Phi)})$. This complexity could, however, be exponential in $k(\mathcal{I}, \Phi)$. Hence, for large time bound or large uniformization rate, we still might have the state-explosion problem.

3.3 Handling rewards

In CTMCs, states and/or transitions can be equipped with rewards (or costs), yielding so-called continuous-time Markov reward models (MRMs). Baier *et. al.* [2] 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 which represents a bound for the accumulated rewards. Besides performability [20] and other important dependability measures, Cloth *et. al.* [6] showed that survivability properties[2] can be expressed in CSRL.

Effective CSRL model checking is restricted to cases where either time is unbounded, but rewards are bounded [2], 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 [4] 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 developed for model checking properties expressed in CSL can be extended to handle the relevant fragment of CSRL in a rather straightforward way, provided we are dealing with reward-bounded models, i.e. models where the supremum of state and transition rewards considered are finite. This also holds for other reward properties [2] such as instantaneous rewards and cumulative rewards.

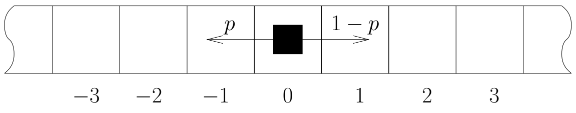
4 Experimental Results

This section presents experimental results with two different implementations of this approach. One is dedicated to infinite state models, the other is a variation of the PRISM model checker [14], for very large finite models. All experiments were run on a Linux machine with an AMD Athlon(tm) XP 2600+ processor at 2 GHz equipped with 2GB of RAM.

4.1 Infinite-State Models

We have implemented a model checker for infinite Markov (reward) models. The models are specified by probabilistic programs in a guarded command language with unbounded integers and unbounded reals. This language is used in PRISM [14] for finite models and was later extended [27, 13] to infinite models. In the first analysis phase, the model is explored as needed up to the truncation point, according to the on-the-fly strategy to determine the dynamic uniformization rate as described in Subsection 3.1. The resulting uniformization rate and the rate matrix are then submitted to the MRMC model checker [16].

Random Walk. We consider a discrete-space, continuous-time random walk model, depicted in the figure below. A walker starts from the initial position 0. With a rate of λ, the walker changes position. With a probability of p the field left to the current position is chosen and with a probability of $1 - p$ the one to the right.



We have considered two properties: (1) a cumulative 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 more than n fields to the right within a certain time bound t, expressed by $\mathcal{P}_{=?}(\Diamond^{\leq t} n \leq 10)$. 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 for the second property was no larger than 10 milliseconds. As apparent from the data, the size of the model is proportional to the depth and the time bound t.

Jackson Queueing Networks [15]. A Jackson Queueing Network (JQN) is a system consisting of a number of n interconnected queueing stations. Jobs arrive from the environment with a negative exponential inter-arrival time and are distributed to station i with probability $r_{0,1}$. 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

[2]Survivability refers to the ability of a system to recover from disastrous circumstances.

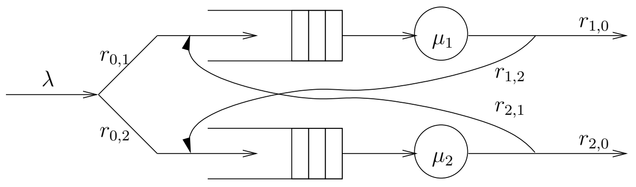
Table 1. Random walk

t	constr. time (ms)	mem. (MB)	depth	# states	# transitions	analysis time (ms)	exp. dis.	Probability
10	26.7	1.16	153	307	610	14.6	5.5	0.0022
20	29.0	1.16	163	327	650	16.0	10.5	0.0266
30	30.4	1.27	173	347	690	18.1	15.5	0.0688
100	42.8	1.29	243	487	970	34.0	50	0.3173

Table 2. Jackson Queueing Networks with two queues

t	constr. time (ms)	mem. (MB)	depth	# states	# transitions	analysis time (ms)	Probability
10	4661.8	11.32	243	6887	40158	154	0.0224554
20	14937.1	15.88	343	9887	57758	314	0.2691432
30	37237.1	20.36	443	12887	75358	514	0.5351491
40	67305.7	24.95	543	15887	92958	766	0.7106415
50	111637.6	30.23	660	19397	113550	1124	0.8192941
60	161903.4	35.49	775	22847	133790	1535	0.8867635

queue j with probability $r_{i,j}$. It is possible for a job to be put back into the same queue again, i.e. $r_{i,i} > 0$ is allowed. A JQN with two queues is depicted in the figure below:



JQNs have an infinite state space because the queues are unbounded. In this paper, we consider two JQN models. The first one is a JQN with two queues where we fix the values of the model to $\lambda = 5$, $\mu_1 = 2$, $\mu_2 = 3$ and

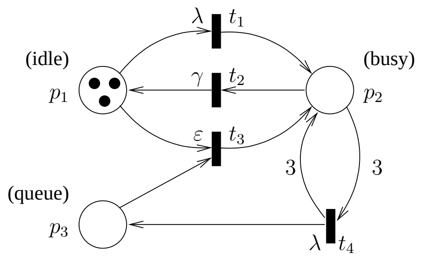
$$R = \begin{bmatrix} 0.0 & 0.4 & 0.6 \\ 0.8 & 0.0 & 0.2 \\ 0.7 & 0.3 & 0.0 \end{bmatrix}$$

which is the matrix of the $r_{i,j}$. We compute the probability that, within t time units, a state is reached in which 10 or more jobs are in the first and 20 or more of them are in the second queue. Results are given in Table 2. The other model is a JQN with three queues with parameters: $\lambda = 5$, $\mu_1 = 3$, $\mu_2 = 1$, $\mu_3 = 1$ and

$$R = \begin{bmatrix} 0.0 & 0.5 & 0.25 & 0.25 \\ 0.6 & 0.4 & 0.0 & 0.0 \\ 0.6 & 0.4 & 0.0 & 0.0 \\ 0.7 & 0.3 & 0.0 & 0.0 \end{bmatrix}$$

We compute the probability, within time t, of reaching a state in which two or more jobs are in queue 1 and also there is at least one job in queue 2 or queue 3. Results are given in Table 3. As we can observe, the number of explored states in this case study is approximately linear to the depth and the time bound, while in principle it is quadratic for two queues and cubic for three queues, respectively. The linearity is owed to the fact that our model checker does not further explore goal states.

Quasi-Birth-Death Processes. In [17], 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 nonempty, 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 [17] is depicted in the figure above for the case $m = 3$. Tokens in p_1 represent the number of idle processors, whereas p_2 describes the number of busy processors and 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

Table 3. Jackson Queueing Networks with three queues

t	constr. time (ms)	mem. (MB)	depth	# states	# transitions	analysis time (ms)	Probability
1	325541.5	53.99	153	23870	244641	561	0.4933299
2	428169.5	61.04	163	27060	277746	680	0.8288087
3	550725.1	68.65	173	30450	312951	9520	0.9454921

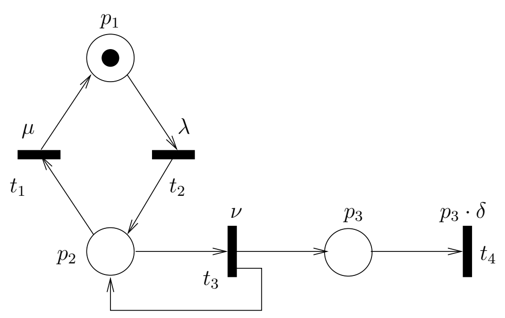
Table 4. Quasi-Birth-Death Process

λ	constr. time (ms)	mem. (MB)	depth	# states	# transitions	analysis time (ms)	Probability
1	1448.0	9.05	1270	5079	12681	280	0.9993348
2	1464.3	9.24	1281	5123	12791	215	0.9483252
3	1496.5	9.24	1292	5167	12901	242	0.6983419
4	1483.9	9.37	1303	5211	13011	212	0.3965853
5	1512.9	9.37	1314	5255	13121	226	0.2147077
6	1530.4	9.49	1325	5299	13231	218	0.1249413

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 an initial state and checking the formula $\mathcal{P}_{=?}(\Diamond^{\leq t} p_1 = 3 \wedge p_3 = 0)$. Results for different λ are given in Table 4 for $\mu = 100, m = 3, t = 10$. The uniformization rate of the underlying CTMC is $100+\lambda$. As we can observe from the table, the depth is linear to the uniformization rate.

Protein Synthesis [11]. We analyze an SPN model of protein synthesis, as depicted in the figure below. 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. This property can expressed by: $\mathcal{P}_{=?}(\Diamond^{[10,t]} p_3 \geq 20 \wedge \mathit{inactive})$ where $\mathit{inactive}$ is an atomic proposition representing inactive genes.

The results for $\lambda = 1, \mu = 5, \nu = 1, \delta = 0.02$ are presented in Table 5. In this case study, the sum of the outgoing 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. For $t \leq 45$, we are still able to determine the minimal dynamic uniformization rate on-the-fly. However, for time bounds of about 50 and beyond, the algorithm does not terminate. In this case, $k(\mathcal{I}, qt)$ grows faster with the depths and the dynamic truncation determination never terminates. We note that for this kind of rate unbounded model the adaptive uniformization method [25] could be used which remains our future work.

4.2 Finite-State Models

We assess our method on models with very large finite state spaces. To this end, we employ PRISM [14], a finite-state model checker incorporating symbolic data structures and algorithms based on *Multi-Terminal Binary Decision Diagrams* (MTBDDs). We compare PRISM in original mode with a modified version of PRISM (denoted by 'truncated' in tables) that employs our truncation method, i.e.

Table 5. Protein Synthesis

t	constr. time (ms)	mem. (MB)	depth	# states	# transitions	analysis time (ms)	Probability
30	355.0	4.02	930	952	2825	10	0.005E-4
35	626.3	6.03	1546	1568	4673	72	0.016E-4
40	1366.6	10.76	3018	3040	9089	259	0.045E-4
45	10484.7	30.96	9243	9265	27764	4208	0.106E-4

it performs transient analysis and reward computation only up to the truncation point. To avoid major changes within PRISM, truncation-based PRISM still constructs the rate matrix of the *entire* model beyond the truncation point.

Workstation cluster. We consider the *dependability of a fault-tolerant workstation cluster* [12]. In the cluster of workstations case study, there are two sub-clusters. Each sub-cluster consists of 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 being repaired by a single repair unit responsible for the entire system. We consider the following two informal quality of service (QoS) constraints:

- $Minimum$ QoS requires at least k ($k < N$) workstations to be operational where $k = \lfloor \frac{3}{4}N \rfloor$.
- $Premium$ QoS requires at least N workstations to be operational.

In both cases, workstations have to be connected via switches. If in each sub-cluster the number of operational workstations is smaller than k (and N respectively), the backbone is required to be operational to provide the required service.

Let $Minimum$ and $Premium$ be two state formulae corresponding the informal QoS constraints described above. We consider the following two bounded properties:

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

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

In Table 6, we give statistics of the model construction using PRISM 3.1 [14] in original mode, i.e. without truncation, and then our version, i.e. with truncation, respectively. Note that as the parameter N increases, all these numbers increase accordingly. In the truncated version, the truncation point is also computed.

For $N = 64$, the two version have the same statistics. This is due to the fact that the truncation point for this case is larger than the depth of all reachable states.

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

Table 6. Model construction statistics.

N	states	transitions	iter.	time (s)
	PRISM			
64	151,060	733,216	133	0.27
128	597,012	2,908,192	261	0.78
256	2,373,652	11,583,520	517	3.11
512	9,465,876	46,235,680	1029	12.25
1024	37,806,100	184,746,016	2053	49.19
2048	151,109,652	738,590,762	4101	193.36
4096	604,209,172	2,953,576,480	8194	814.42
	truncated			
64	151,060	733,216	133	0.29
128	573,743	2,793,153	223	2.73
256	892,021	4,334,849	224	5.27
512	900,053	4,373,999	225	7.96
1024	916,225	4,452,827	227	17.08
2048	949,001	4,612,595	231	50.26
4096	1,016,281	4,940,579	239	215.88

While the number of states and transitions for PRISM increases dramatically with parameter N, growth is slower in the truncated version. The reason is apparent from the column displaying the iterations of the truncated version: much fewer iterations are done compared to PRISM. As N increases, the truncation point grows only slowly, and hence, only a small fraction of the state space needs to be explored in the truncated construction.

As explained earlier, our prototype implementation constructs the rate matrix beyond the truncation point. This is alleviated by the fact that the full untruncated rate matrix is stored symbolically in the form of a multi-terminal BDD and never unfolded explicitly, e.g. as a sparse matrix. As a result, although truncation makes the model size (Table 6) and analysis time (Table 7) grow only marginally in N, this is not the case for overall analysis time, which includes construction of the matrix.

In Table 7, we present experimental results of model checking the properties A and B. We have computed the results in both algorithms up to precision $\varepsilon = 10^{-6}$ and indeed, the corresponding results are the same, up to ε. Using PRISM, the memory and the time used for model checking increase approximately in the same way as the size of the model. For $N \geq 2048$, PRISM cannot model check these

Table 7. Model checking statistics.

	PRISM		truncated		
	Property A				
N	m. (kb)	t. (s)	m. (kb)	t. (s)	result
64	5,077	2.73	5,077	2.73	5.89E-8
128	16,587	11.98	16,005	11.89	5.90E-8
256	62,739	53.37	26,050	35.84	5.92E-8
512	245,086	271.05	26,602	40.44	5.96E-8
1024	969,430	1,602.93	26,802	38.88	6.01E-8
2048	–	–	27,755	41.65	6.07E-8
4096	–	–	29,214	46.75	6.11E-8
	Property B				
64	4,859	5.18	4,859	5.16	0.1190
128	16,446	26.67	15,990	24.02	0.2282
256	62,177	151.19	25,521	44.90	0.4203
512	243,274	1,038.58	25,945	46.17	0.7181
1024	964,916	7,160.43	26,580	49.63	1.0812
2048	–	–	27,531	52.62	1.3844
4096	–	–	29,373	58.29	1.5565

properties within two hours; this is denoted by –. Not surprisingly, the memory and the time used in the truncated version increase very slowly.

5 Conclusions and Future Work

In this paper, we have introduced time-bounded model checking techniques for infinite CTMCs and infinite Markov reward models. Our experiments have shown the feasibility of our method, and that it can be beneficial to apply the truncation approach also to finite, but excessively large models.

In the protein synthesis case study, we observe that the depth grows much faster than the time point. The reason is that, as the state exploration continues, newly explored states have ever higher exit rates, thus also higher uniformization rate. For this kind of model, it stands to reason that adaptive uniformization would perform better since the depth point should be smaller. This remains future work.

References

[1] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton. Model-checking continous-time Markov chains. *ACM Trans. Comput. Log.*, 1(1):162–170, 2000.

[2] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. On the logical characterisation of performability properties. In *ICALP*, pages 780–792, 2000.

[3] 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.

[4] M. D. Beaudry. Performance-related reliability measures for computing systems. *IEEE Trans. on Comp. Sys.*, 27(6):540–547, 1978.

[5] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers*, 58:118–149, 2003.

[6] L. Cloth and B. R. Haverkort. Model checking for survivability. In *QEST*, pages 145–154, 2005.

[7] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.*, 2(3):241–266, 1982.

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

[9] W. K. Grassmann. Transient solutions in markovian queueing systems. *Computers & OR*, 4(1):47–53, 1977.

[10] W. K. Grassmann. Finding transient solutions in Markovian event systems through randomization. In *Numerical Solution of Markov Chains*, pages 357–371, 1991.

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

[12] 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.

[13] H. Hermanns, B. Wachter, and L. Zhang. Probabilistic CEGAR. In *CAV*, 2008.

[14] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism: A tool for automatic verification of probabilistic systems. In *TACAS*, pages 441–444, 2006.

[15] J. Jackson. Networks of waiting lines. *Operations Research*, 5:518–521, 1957.

[16] J.-P. Katoen, M. Khattri, and I. S. Zapreev. A Markov reward model checker. In *QEST*, pages 243–244, 2005.

[17] J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-valued abstraction for continuous-time Markov chains. In *CAV*, pages 311–324, 2007.

[18] J.-P. Katoen, M. Z. Kwiatkowska, G. Norman, and D. Parker. Faster and symbolic ctmc model checking. In *PAPM-PROBMIV*, pages 23–38, 2001.

[19] M. Z. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In *SFM*, pages 220–270, 2007.

[20] J. F. Meyer. Performability evaluation: where it is and what lies ahead. *IEEE Int. Comp. Perf. and Dependability Symp*, pages 334–343, 1995.

[21] A. Remke and B. R. Haverkort. Csl model checking algorithms for infinite-state structured Markov chains. In *FORMATS*, pages 336–351, 2007.

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

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

[24] 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.

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

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

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