Efficient CTMC Model Checking of Linear Real-Time Objectives

Benoît Barbot[2], Taolue Chen[3], Tingting Han[1],
Joost-Pieter Katoen[1,3] and Alexandru Mereacre[1]

[1]MOVES, RWTH Aachen University, Germany
[2] ENS Cachan, France
[3] FMT, University of Twente, The Netherlands

Abstract. This paper makes verifying continuous-time Markov chains (CTMCs) against deterministic timed automata (DTA) objectives practical. We show that verifying 1-clock DTA can be done by analyzing subgraphs of the product of CTMC $\mathcal{C}$ and the region graph of DTA $\mathcal{A}$. This improves upon earlier results and allows to only use standard analysis algorithms. Our graph decomposition approach naturally enables bisimulation minimization as well as parallelization. Experiments with various examples confirm that these optimizations lead to significant speed-ups. We also report on experiments with multiple-clock DTA objectives. The objectives and the size of the problem instances that can be checked with our prototypical tool go (far) beyond what could be checked so far.

1 Introduction

For more than a decade, the verification of continuous-time Markov chains, CTMCs for short, has received considerable attention, cf. the recent survey [7]. Due to unremitting improvements on algorithms, (symbolic) data structures and abstraction techniques, CTMC model checking has emerged into a valuable analysis technique which – supported by powerful software tools– has been adopted by various researchers for systems biology, queueing networks, and dependability.

The focus of CTMC model checking has primarily been on checking stochastic versions of the branching temporal logic CTL, such as CSL [6]. The verification of LTL objectives reduces to applying well-known algorithms [20] to embedded discrete-time Markov chains (DTMCs). Linear objectives equipped with timing constraints, have just recently been considered. This paper treats linear *real-time* specifications that are given as deterministic timed automata (DTA). These include, e.g., properties of the form: what is the probability to reach a given target state within the deadline, while avoiding "forbidden" states and not staying too long in any of the "dangerous" states on the way. Such properties can neither be expressed in CSL nor in dialects thereof [5, 14]. Model checking DTA objectives amounts to determining the probability of the set of paths of CTMC $\mathcal{C}$ that are accepted by the DTA $\mathcal{A}$, i.e., $\mathit{Prob}(\mathcal{C} \models \mathcal{A})$. We recently showed in [12] that this equals the reachability probability in a finite *piecewise deterministic Markov process* (PDP) that is obtained by a region construction on the product $\mathcal{C} \otimes \mathcal{A}$. This paper reports on how to make this approach practical, i.e., how to efficiently realize CTMC model checking against DTA objectives.

As a first step, we show that rather than taking the region graph of the product $\mathcal{C} \otimes \mathcal{A}$, which is a somewhat ad-hoc mixture of CTMCs and DTA, we can apply a standard region construction on DTA $\mathcal{A}$ *prior to* building the product. This enables applying a standard region construction for timed automata. The product of this region graph with CTMC $\mathcal{C}$ yields the PDP to be analyzed. Subsequently, we exploit that for 1-clock DTA, the resulting PDP can be decomposed into subgraphs—each of which is a CTMC [12]. In this case, $\mathit{Prob}(\mathcal{C} \models \mathcal{A})$ is the solution of a system of linear equations whose coefficients are transient probability distributions of the (slightly amended) subgraph CTMCs. We adapt the algorithm for lumping [13, 19] on CTMCs to our setting and prove that this preserves reachability probabilities, i.e., keeps $\mathit{Prob}(\mathcal{C} \models \mathcal{A})$ invariant. As the graph decomposition naturally enables parallelization, our tool implementation also supports the distribution of computing transient probabilities over multiple multi-core computers. Finally, multi-clock DTA objectives –for which the graph decomposition does not apply– are supported by a discretization of the product PDP.

Three case studies from different application fields are used to show the feasibility of this approach. The first case study has been taken from [3] which considers 1-clock DTA as time constraints of until modalities. Although using a quite different approach, our verification results coincide with [3]. The running time of our implementation (without lumping and parallelization) is about three orders of magnitude faster than [3]. Other considered case studies are a randomly moving robot, and a real case study from systems biology [15]. Bisimulation quotienting (i.e., lumping) yields state space reduction of up to one order of magnitude, whereas parallelizing transient analysis yields speed-ups of up to a factor 13 on 20 cores, depending on the number of subgraphs in the decomposition.

The discretization approach for multi-clock DTA may give rise to large models: checking the robot example (up to 5,000 states) against a two-clock DTA yields a 40-million state DTMC for which simple reachability probabilities are to be determined.

Related work. The logic asCSL [5] extends CSL by (time-bounded) regular expressions over actions and state formulas as path formulas. $\mathrm{CSL}^{\mathrm{TA}}$ [14] allows 1-clock DTA as time constraints of until modalities; this subsumes acCSL. The joint behavior of $\mathcal{C}$ and DTA $\mathcal{A}$ is interpreted as a Markov renewal process. A prototypical implementation of this approach has recently been reported in [3]. Our algorithmic approach for 1-clock DTA is different, yields the same results, and is –as shown in this paper– significantly faster. Moreover, lumping is not easily possible (if at all) in [3].

In addition, it naturally supports bisimulation minimization and parallelization. Bisimulation quotienting in CSL model checking has been addressed in [18]. The works [4, 9] provide a quantitative interpretation to timed automata where delays of unbounded clocks are governed by exponential distributions. Brazdil et al. present an algorithmic approaches towards checking continuous-time stochastic games (supporting more general distributions) against DTA specifications [11]. To our knowledge, tool implementations of [4, 9, 11] do not exist.

Organization of the paper. Section 2 provides the preliminaries and summarizes the main results of [12]. Section 3 describes our bisimulation quotienting algorithm. Section 4 reports on the experiments for 1-clock DTA objectives. Section 5 describes the

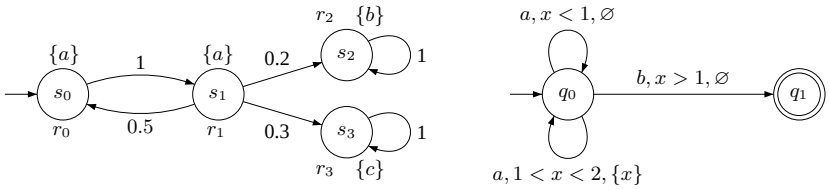
discretization approach and gives experimental results for multi-clock DTA. Finally, Section 6 concludes. (Proofs are omitted and can be found in the full version available at http://moves.rwth-aachen.de/i2/han.)

2 Preliminaries

2.1 Continuous-time Markov chains

Definition 1 (CTMC). *A* (*labeled*) continuous-time Markov chain (CTMC) *is a tuple* $\mathcal{C} = (S, \mathrm{AP}, L, \alpha, \mathbf{P}, E)$ *where* S *is a* finite *set of* states*;* AP *is a finite set of* atomic propositions*;* $L : S \to 2^{\mathrm{AP}}$ *is the* labeling function*;* $\alpha \in Distr(S)$ *is the* initial distribution*;* $\mathbf{P} : S \times S \to [0, 1]$ *is a stochastic matrix; and* $E : S \to \mathbb{R}_{\geqslant 0}$ *is the* exit rate function.

Here, $Distr(S)$ is the set of probability distributions on set S. The probability to exit state s in t time units is given by $\int_0^t E(s) \cdot e^{-E(s)\tau} d\tau$. The probability to take the transition $s \to s'$ in t time units is $\mathbf{P}(s, s') \cdot \int_0^t E(s) e^{-E(s)\cdot\tau} d\tau$. The embedded DTMC of $\mathcal{C}$ is $(S, \mathrm{AP}, L, \alpha, \mathbf{P})$. A *(timed) path* in CTMC $\mathcal{C}$ is a sequence $\rho = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \cdots$ such that $s_i \in S$ and $t_i \in \mathbb{R}_{>0}$. Let $Paths^{\mathcal{C}}$ be the set of paths in $\mathcal{C}$ and $\rho[n] := s_n$ be the n-th state of ρ. The definitions of a *Borel space* on paths through CTMCs,and the *probability measure* Pr follow [20, 6].



(a) An example CTMC (b) An example DTA

Fig. 1. Example CTMC and DTA

Example 1. Fig. 1(a) shows an example CTMC with $\mathrm{AP} = \{a, b, c\}$ and initial state s_0, i.e., $\alpha(s) = 1$ iff $s = s_0$. The exit rates are indicated at the states, whereas the transition probabilities are attached to the edges. An example path is $\rho = s_0 \xrightarrow{2.5} s_1 \xrightarrow{1.4} s_0 \xrightarrow{2} s_1 \xrightarrow{2\pi} s_2 \cdots$ with $\rho[2] = s_0$ and $\rho[3] = s_1$.

Definition 2 (Bisimulation). *Let* $\mathcal{C} = (S, \mathrm{AP}, L, \alpha, \mathbf{P}, E)$ *be a* CTMC*. Equivalence relation* $\mathcal{R}$ *on* S *is a* (strong) bisimulation *on* $\mathcal{C}$ *if for* $s_1 \mathcal{R} s_2$*:*

$$L(s_1) = L(s_2) \text{ and } \mathbf{P}(s_1, C) = \mathbf{P}(s_2, C) \text{ for all } C \text{ in } S/\mathcal{R} \text{ and } E(s_1) = E(s_2).$$

Let $s_1 \sim s_2$ *if there exists a strong bisimulation* $\mathcal{R}$ *on* $\mathcal{C}$ *with* $s_1 \mathcal{R} s_2$.

The quotient CTMC under the coarsest bisimulation $\sim$ can be obtained by partition-refinement with time complexity $\mathcal{O}(m \log n)$ [13]. A simplified version with the same complexity is recently proposed in [19].

2.2 Deterministic timed automata

Clock variables and valuations. Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of *clocks* in $\mathbb{R}_{\geqslant 0}$. An $\mathcal{X}$-valuation (valuation for short) is a function $\eta : \mathcal{X} \to \mathbb{R}_{\geqslant 0}$; $\mathbf{0}$ is the valuation that assigns 0 to all clocks. A *clock constraint* g on $\mathcal{X}$ has the form $x \bowtie c$, or $g' \wedge g''$, where $x \in \mathcal{X}$, $\bowtie \in \{<, \leqslant, >, \geqslant\}$ and $c \in \mathbb{N}$. Diagonal clock constraints like $x - y \bowtie c$ do not extend the expressiveness [8], and are not considered. Let $\mathcal{CC}(\mathcal{X})$ denote the set of clock constraints over $\mathcal{X}$. An $\mathcal{X}$-valuation η *satisfies* constraint g, denoted as $\eta \models g$ if $\eta(x) \bowtie c$ for g of the form $x \bowtie c$, or $\eta \models g' \wedge \eta \models g''$ for g of the form $g' \wedge g''$. The reset of η w.r.t. $X \subseteq \mathcal{X}$, denoted $\eta[X := 0]$, is the valuation η' with $\forall x \in X.\ \eta'(x) := 0$ and $\forall x \notin X.\ \eta'(x) := \eta(x)$. For $\delta \in \mathbb{R}_{\geqslant 0}$, $\eta + \delta$ is the valuation η'' such that $\forall x \in \mathcal{X}.\ \eta''(x) := \eta(x) + \delta$.

Definition 3 (DTA). *A* deterministic timed automaton (DTA) *is a tuple* $\mathcal{A} = (\Sigma, \mathcal{X}, Q, q_0, Q_F, \to)$ *where* Σ *is a finite* alphabet*;* $\mathcal{X}$ *is a finite set of* clocks*;* Q *is a nonempty finite set of* locations*;* $q_0 \in Q$ *is the* initial location*;* $Q_F \subseteq Q$ *is a set of* accepting locations*; and* $\to \in (Q \setminus Q_F) \times \Sigma \times \mathcal{CC}(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ *satisfies:* $q \xrightarrow{a,g,X} q'$ *and* $q \xrightarrow{a,g',X'} q''$ *with* $g \neq g'$ *implies* $g \cap g' = \varnothing$.

We refer to $q \xrightarrow{a,g,X} q'$ as an *edge*, where $a \in \Sigma$ is the input symbol, the *guard* g is a clock constraint on the clocks of $\mathcal{A}$, $X \subseteq \mathcal{X}$ is a set of clocks and q, q' are locations. The intuition is that the DTA $\mathcal{A}$ can move from location q to q' on reading the input symbol a if the guard g holds, while resetting the clocks in X on entering q'. By convention, we assume $q \in Q_F$ to be a sink, cf. Fig. 1(b).

Paths. A finite (timed) path in $\mathcal{A}$ is of the form $\theta = q_0 \xrightarrow{a_0,t_0} q_1 \cdots q_n \xrightarrow{a_n,t_n} q_{n+1}$ with $q_i \in Q$, $t_i > 0$ and $a_i \in \Sigma$ ($0 \leqslant i \leqslant n$). The path θ is *accepted* by $\mathcal{A}$ if $\theta[i] \in Q_F$ for some $0 \leqslant i \leqslant |\theta|$ and for all $0 \leqslant j < i$, it holds that $\eta_0 = \mathbf{0}$, $\eta_j + t_j \models g_j$ and $\eta_{j+1} = (\eta_j + t_j)[X_j := 0]$, where η_j is the clock valuation on entering q_j and g_j the guard of $q_i \to q_{i+1}$. The infinite path $\rho = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \cdots$ in CTMC $\mathcal{C}$ is accepted by $\mathcal{A}$ if for some $n \in \mathbb{N}$, $s_0 \xrightarrow{t_0} s_1 \cdots s_{n-1} \xrightarrow{t_{n-1}} s_n$ induces the DTA path $\hat{\rho} = q_0 \xrightarrow{L(s_0),t_0} q_1 \cdots q_{n-1} \xrightarrow{L(s_{n-1}),t_{n-1}} q_n$, which is accepted by $\mathcal{A}$. The set of CTMC paths accepted by a DTA is measurable [12]; our measure of interest is given by:

$$\mathit{Prob}(\mathcal{C} \models \mathcal{A}) \;=\; \Pr\{\, \rho \in \mathit{Paths}^{\mathcal{C}} \mid \hat{\rho} \text{ is accepted by DTA } \mathcal{A} \,\}.$$

Regions. Regions are sets of valuations, often represented as constraints. Let $\mathcal{R}e(\mathcal{X})$ be the set of regions over $\mathcal{X}$. For regions $\Theta, \Theta' \in \mathcal{R}e(\mathcal{X})$, Θ' is the *successor region* of Θ if for all $\eta \models \Theta$ there exists $\delta \in \mathbb{R}_{>0}$ such that $\eta + \delta \models \Theta'$ and $\forall \delta' < \delta.\ \eta + \delta' \models \Theta \vee \Theta'$. The region Θ *satisfies* a guard g, denoted $\Theta \models g$, iff $\forall \eta \models \Theta.\ \eta \models g$. The *reset operation* on region Θ is defined as $\Theta[X := 0] := \{\eta[X := 0] \mid \eta \models \Theta\}$.

Definition 4 (Region graph [2]). *Let* $\mathcal{A} = (\Sigma, \mathcal{X}, Q, q_0, Q_F, \to)$ *be a* DTA*. The* region graph $\mathcal{G}(\mathcal{A})$ *of* $\mathcal{A}$ *is* $(\Sigma, W, w_0, W_F, \dashrightarrow)$ *with* $W = Q \times \mathcal{R}e(\mathcal{X})$ *the set of states;* $w_0 = (q_0, \mathbf{0})$ *the initial state;* $W_F = Q_F \times \mathcal{R}e(\mathcal{X})$ *the set of final states; and* $\dashrightarrow \subset W \times ((\Sigma \times 2^{\mathcal{X}}) \uplus \{\, \delta \,\}) \times W$ *the smallest relation such that:*

$(q,\Theta) \overset{\delta}{\dashrightarrow} (q,\Theta')$ *if* Θ' *is the successor region of* Θ*; and*
$(q,\Theta) \overset{a,X}{\dashrightarrow} (q',\Theta')$ *if* $\exists g \in \mathcal{CC}(\mathcal{X})$ *s.t.* $q \xrightarrow{a,g,X} q'$ *with* $\Theta \models g$ *and* $\Theta[X := 0] = \Theta'$.

Product. Our aim is to determine *Prob*($\mathcal{C} \models \mathcal{A}$). This can be accomplished by computing reachability probabilities in the region graph of $\mathcal{C} \otimes \mathcal{A}$ where $\otimes$ denotes a product construction between CTMC $\mathcal{C}$ and DTA $\mathcal{A}$ [12]. To ease the implementation, we consider the product $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$.

Definition 5 (Product). *The product of CTMC* $\mathcal{C} = (S, \mathrm{AP}, L, \alpha, \mathbf{P}, E)$ *and DTA region graph* $\mathcal{G}(\mathcal{A}) = (\Sigma, W, w_0, W_F, \dashrightarrow)$*, denoted* $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$*, is the tuple* $(V, \alpha', V_F, \Lambda, \hookrightarrow)$ *with* $V = S \times W$*,* $\alpha'(s, w_0) = \alpha(s)$*,* $V_F = S \times W_F$*, and*

- $\hookrightarrow \subseteq V \times \big(([0,1] \times 2^{\mathcal{X}}) \uplus \{\delta\}\big) \times V$ *is the smallest relation such that:*
  - $(s,w) \overset{\delta}{\hookrightarrow} (s,w')$ iff $w \overset{\delta}{\dashrightarrow} w'$*; and*
  - $(s,w) \overset{p,X}{\hookrightarrow} (s',w')$ iff $p = \mathbf{P}(s,s')$*,* $p > 0$*, and* $w \overset{L(s),X}{\dashrightarrow} w'$.
- $\Lambda : V \to \mathbb{R}_{\geqslant 0}$ *is the* exit rate function *where:*

$$\Lambda(s,w) \;=\; \begin{cases} E(s) & \text{if } (s,w) \overset{p,X}{\hookrightarrow} (s',w') \text{ for some } (s',w') \in V \\ 0 & \text{otherwise.} \end{cases}$$

The (reachable fragment of the) product of CTMC $\mathcal{C}$ in Fig. 1(a) and the region graph of DTA $\mathcal{A}$ in Fig. 1(b) is given in Fig. 2. It turns out that $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$ is identical to $\mathcal{G}(\mathcal{C} \otimes \mathcal{A})$, the region graph of $\mathcal{C} \otimes \mathcal{A}$ as defined in [12].

Theorem 1. *For any* CTMC $\mathcal{C}$ *and any* DTA $\mathcal{A}$*,* $\mathcal{C} \otimes \mathcal{G}(\mathcal{A}) = \mathcal{G}(\mathcal{C} \otimes \mathcal{A})$.

As a corollary [12], it follows that *Prob*($\mathcal{C} \models \mathcal{A}$) equals the reachability probability of the accepting states in $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$. In addition, $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$ is a PDP.

2.3 Decomposition for 1-clock DTA

Let $\mathcal{A}$ be a 1-clock DTA and $\{c_0, \ldots, c_m\} \subseteq \mathbb{N}$ with $0 = c_0 < c_1 < \cdots < c_m$ the constants appearing in its clock constraints. Let $\Delta c_i = c_{i+1} - c_i$ for $0 \leqslant i < m$. The product $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$ can be split into $m+1$ subgraphs, denoted $\mathcal{G}_i$ for $0 \leqslant i \leqslant m$, such that any state in $\mathcal{G}_i$ has a clock valuation in $[c_i, c_{i+1})$ for $0 \leqslant i < m$ and in $[c_m, \infty)$ for $i = m$. Each column in Fig. 2 constitutes such subgraph. Subgraph $\mathcal{G}_i$ thus captures the joint behavior of CTMC $\mathcal{C}$ and DTA $\mathcal{A}$ in the interval $[c_i, c_{i+1})$. All transitions within $\mathcal{G}_i$ are probabilistic; delay-transitions, i.e., δ-labeled transitions, yield a state

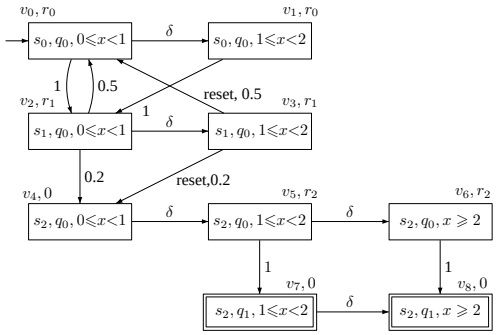


Fig. 2. Example $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$

in $\mathcal{G}_{i+1}$, whereas a clock reset (of the only clock) yields a state in $\mathcal{G}_0$. In fact subgraph $\mathcal{G}_i$ is a CTMC. To take the effect of "reset" transitions into account, define the CTMC $\mathcal{C}_i$ with state space $V_i \cup V_0$, with all edges from V_i to V_0, all edges between V_i-vertices, but no outgoing edges from V_0-vertices.

Definition 6 (Augmented CTMC). *Let $\mathcal{G} = \mathcal{C} \otimes \mathcal{G}(\mathcal{A}) = (V, \alpha, V_F, \Lambda, \hookrightarrow)$. Subgraph $\mathcal{G}_i$ of $\mathcal{G}$ induces the CTMC $\mathcal{C}_i = (V_i \cup V_0, \mathrm{AP}_i, L_i, \alpha_i, \mathbf{P}_i^a, E_i)$ with:*

- $\mathrm{AP}_i = \begin{cases} \{\delta_v \mid v \in V_{i+1}\} \cup \{Rst_v \mid v \in V_0\} & \text{if } i < m \\ \{Rst_v \mid v \in V_0\} \cup \{F\} & \text{if } i = m \end{cases}$
- $L_i(v) = \{\delta_{v'}\}$ *if* $v \overset{\delta}{\hookrightarrow} v'$ *and* $L_i(v) = \{Rst_{v'}\}$ *if* $v \overset{p,X}{\hookrightarrow} v'$, *for* $i \leqslant m$; $L_m(v) = \{F\}$ *if* $v \in V_F \cap V_m$;
- $\alpha_0 = \alpha$, *and for* $0 \leqslant i < m$, α_{i+1} *is the transient distribution of* $\mathcal{C}_i$ *at* Δc_i;
- $\mathbf{P}_i^a(v, v') = \begin{cases} p & \text{if} \quad v, v' \in V_i \wedge v \overset{p,\varnothing}{\hookrightarrow} v' \quad \text{or} \quad v \in V_i \wedge v' \in V_0 \wedge v \overset{p,X}{\hookrightarrow} v' \\ 1 & \text{if} \quad v = v' \in V_0 \\ 0 & \text{otherwise}. \end{cases}$
- $E_i(s, w) = E(s)$ *where* E *is the exit-rate function of CTMC* $\mathcal{C}$.

State v in CTMC $\mathcal{C}_i$ is labeled with $Rst_{v'}$ if a clock reset in v yields v'; similarly it is labeled with $\delta_{v'}$ if a delay results in state v' in the successor CTMC $\mathcal{C}_{i+1}$. These labels are relevant for bisimulation quotienting as explained later. The proposition F indicates the "final" CTMC states in the CTMC $\mathcal{C}_m$.

The matrix $\mathbf{P}_i^a$ can be split into the matrices $\mathbf{P}_i$ and $\hat{\mathbf{P}}_i^a$ where $\mathbf{P}_i$ contains only (probabilistic) transitions inside V_i whereas $\hat{\mathbf{P}}_i^a$ contains transitions from V_i to V_0. The transient probability matrix for $\mathcal{C}_i$ is defined as the solution of the equation $\mathbf{\Pi}_i^a(x) = \int_0^x E_i \cdot e^{-E_i\tau} \cdot \mathbf{P}_i^a \cdot \mathbf{\Pi}_i^a(x-\tau) d\tau + e^{-E_i x}$. The matrices $\mathbf{\Pi}_i$ and $\hat{\mathbf{\Pi}}_i^a$ can be defined in a similar way as $\mathbf{P}$. Let $\mathbf{A}_i := \mathbf{\Pi}_0 \cdot \mathbf{\Pi}_1 \cdot \ldots \cdot \mathbf{\Pi}_i$ for $0 \leqslant i \leqslant m-1$ and $\mathbf{B}_i := \hat{\mathbf{\Pi}}_0^a$ if $i = 0$ and $\mathbf{B}_i := \mathbf{A}_{i-1} \cdot \hat{\mathbf{\Pi}}_i^a + \mathbf{B}_{i-1}$ if $1 \leqslant i < m$. We can now define the linear equation system $\mathbf{x} \cdot \mathbf{M} = \mathbf{f}$ with

$$\mathbf{M} = \left(\begin{array}{c|c} \mathbf{I}_{n_0} - \mathbf{B}_{m-1} & \mathbf{A}_{m-1} \\ \hline \hat{\mathbf{P}}_m^a & \mathbf{I}_{n_m} - \mathbf{P}_m \end{array}\right) \text{ and } \mathbf{f}(v) = \begin{cases} 1 & \text{if } v \in V_m \wedge F \in L_m(v) \\ 0 & \text{othewise} \end{cases}$$

Here $\mathbf{I}_n$ denotes the identity matrix of cardinality n. For details, consult [12].

Theorem 2 ([12]). *For CTMC $\mathcal{C}$ with initial distribution α, 1-clock DTA $\mathcal{A}$ and linear equation system $\mathbf{x} \cdot \mathbf{M} = \mathbf{f}$ with solution $\boldsymbol{U}$, we have that $Prob(\mathcal{C} \models \mathcal{A}) = \alpha \cdot \boldsymbol{U}$.*

Algorithm 1 summarizes the main steps needed to model-check a CTMC against a 1-clock DTA, where as an (optional) optimization step, in line 7 all augmented CTMCs are lumped by the adapted lumping algorithm as explained below.

3 Lumping

It is known that bisimulation minimization is quite beneficial for CSL model checking as experimentally showed in [18]. Besides yielding a state space reduction, it may yield

Algorithm 1 Verifying a CTMC against a 1-clock DTA

Require: a CTMC $\mathcal{C}$ with initial distr. α, a 1-clock DTA $\mathcal{A}$ with constants $c_0, ..., c_m$
Ensure: $\Pr(\mathcal{C} \models \mathcal{A})$

1: $\mathcal{G}(\mathcal{A}) :=$ `buildRegionGraph`($\mathcal{A}$);
2: *Product* $:=$ `buildProduct`($\mathcal{C}, \mathcal{G}(\mathcal{A})$); $\{\mathcal{C} \otimes \mathcal{G}(\mathcal{A})\}$
3: subGraphs $\{\mathcal{G}_i\}_{0 \leqslant i \leqslant m} :=$ `partitionProduct`(*Product*);
4: **for** each subGraph $\mathcal{G}_i$ **do**
5: $\quad \mathcal{C}_i :=$ `buildAugmentedCTMC`($\mathcal{G}_i$); {build augmented CTMC cf. Definition 6}
6: **end for**
7: $\{\mathcal{C}'_i\}_{0 \leqslant i \leqslant m} :=$ `lumpGroupCTMCs`($\{\mathcal{C}_i\}_{0 \leqslant i \leqslant m}$); {lump a group of CTMCs, see Alg. 2}
8: **for** each CTMC $\mathcal{C}'_i$ **do** *TransProb*$_i :=$ `computeTransientProb`($\mathcal{C}'_i, \Delta c_i$); **end for**
9: *linearEqSystem* $:=$ `buildLinearSystem`($\{$*TransProb*$_i\}_{0 \leqslant i \leqslant m}$); {cf. Theorem 2}
10: *probVector* $:=$ `solveLinearSystem`(*linearEqSystem*);
11: **return** $\alpha \cdot$ *probVector*;

substantial speed-ups. The decomposition of the product $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$ into the CTMCs $\mathcal{C}_i$ naturally facilitates a bisimulation reduction of each $\mathcal{C}_i$ prior to computing the transient probabilities in $\mathcal{C}_i$. In order to do so, an amendment of the standard lumping algorithm [13, 19] is needed as the CTMCs to be lumped are connected by delay and reset transitions. Initial states in CTMC $\mathcal{C}_i$ might be the target states of edges whose source states are in a different CTMC, $\mathcal{C}_j$, say, with $i \neq j$. The partitioning of the target states in $\mathcal{C}_i$ will affect the partitioning of the source states in $\mathcal{C}_j$. For delay edges, $i{=}j{+}1$ while for reset transition, $i{=}0$. The intra-CTMC edges thus cause a "cyclic affection" between partitions among all sub-CTMCs. From the state labeling (cf. Def. 6), it follows that for any two states $v, v' \in \mathcal{C}_i$, if their respective successor states in $\mathcal{C}_{i+1}$ (or $\mathcal{C}_0$) can be lumped, then v and v' might be lumped. This implies that any refinement on the lumping blocks in $\mathcal{C}_{i+1}$ might affect the blocks in $\mathcal{C}_i$. Similarly, refining $\mathcal{C}_0$ might affect any $\mathcal{C}_i$, viz., the CTMCs that have a reset edge to $\mathcal{C}_0$.

We initiate the lumping algorithm (cf. Alg. 2) for CTMC $\mathcal{C}_i = (V_i \cup V_0, \mathrm{AP}_i, L_i, \alpha_i, \mathbf{P}_i^a, E_i)$ by taking as initial partition the quotient induced by $\{(v_1, v_2) \in (V_i \cup V_0)^2 \mid L_i(v_1) = L_i(v_2)\}$. This initial partition is successively refined on each $\mathcal{C}_i$ by the standard approach [13, 19], see lines 5–6. We then use the blocks in $\mathcal{C}_{i+1}$ to update AP_i in $\mathcal{C}_i$ and use blocks in $\mathcal{C}_0$ to update AP_i in all the affected $\mathcal{C}_i$'s, cf. lines 7–11. As a result, the new $\mathrm{AP}_i{}'$ may be coarser than the old AP_i:

$$\mathrm{AP}'_i \;=\; \{Rst_{[v]_0} \mid Rst_v \in \mathrm{AP}_i\} \cup \{\delta_{[v]_{i \oplus 1}} \mid \delta_v \in \mathrm{AP}_i\}.$$

Here, $[v]_i$ is the equivalence class in CTMC $\mathcal{C}_i$ containing state v, and $i \oplus 1 = i{+}1$ if $i < m$ and $m \oplus 1 = m$. With the new $\mathrm{AP}_i{}'$, this approach (cf. while-loop) is repeated until all CTMC partitions are stable.

Example 2. Let $v_1 \xrightarrow{\delta} v'_1$ and $v_2 \xrightarrow{\delta} v'_2$ be two delay transitions from CTMC $\mathcal{C}_i$ to $\mathcal{C}_{i+1}$. Then $L(v_1) = \delta_{v'_1}$ and $L(v_2) = \delta_{v'_2}$. Since v_1 and v_2 are labeled differently, they cannot be in one equivalence class. However, if in $\mathcal{C}_{i+1}$ it turns out that v'_1 and v'_2 are in one equivalence class, then we can update AP_i to AP'_i such that now $L(v_1) = L(v_2) = \delta_{v'_1}$. In this case, v_1 and v_2 can be lumped together.

Algorithm 2 LumpGroupCTMCs

Require: a set of CTMCs $\mathcal{C}_i$ with AP_i for $0 \leqslant i \leqslant m$
Ensure: a set of lumped CTMCs $\mathcal{C}'_i$ such that $\mathcal{C}_i \sim \mathcal{C}'_i$

```
1: notStable := true;
2: while notStable do
3:    notStable := false;
4:    for i = m to 0 do
5:       oldAPSize := |AP_i|;
6:       (C_i, AP_i) := lumpCTMC(C_i, AP_i) ;   {lump C_i due to AP_i and update C_i, AP_i}
7:       if oldAPSize > |AP_i|   then            {some states have been lumped in C_i}
8:          notStable := true;
9:          if i = 0   then
10:            for j = 1 to m   do updateResetEdge(AP_0, AP_j) ;     {update AP_j}
11:         else updateDelayEdge(AP_i, AP_{i-1}) ; {update AP_{i-1} according to AP_i}
12: return the new set of CTMCs lumped by the newest AP_i
```

If some states have been updated in $\mathcal{C}_i$ and AP_i has been updated (line 7), there are two cases: if i=0, then we update all AP_j, $j \neq 0$ that have a reset edge to $\mathcal{C}_0$ (line 9-10); otherwise, we update its directly predecessor AP_{i-1} which has a delay edge to $\mathcal{C}_i$ (line 11). This procedure is repeated until all AP_i's are stable.

Theorem 3. *The transient probability distribution in $\mathcal{C}_i$ and its quotient $\mathcal{C}'_i$, obtained by Alg. 2 are equal.*

As a corollary, it follows that the reachability probabilities of the accepting states in $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$, and its quotient obtained by applying Alg. 2 coincide.

4 Experimental Results

Implementation. We implemented our approach in approximately 4,000 lines of OCaml code. Transient probabilities are computed using uniformization, linear equation systems are solved using the Gauss-Seidel algorithm, and lumping has been realized by adapting [13] with the correction explained in [19]. Unreachable states (both forwards from the initial and backwards from the final states) are removed in $\mathcal{C} \otimes \mathcal{G}(\mathcal{A})$ prior to the analysis, and transient probabilities in $\mathcal{C}_i$ are only determined for its initial states, i.e., its entry points. The tool adopts the input format of the MRMC model checker [17]. Thanks to the output facility of PRISM [1], the verification of PRISM models is possible.

Case studies. We conducted extensive experiments with three case studies. The first case study has been taken from [3], and facilitates a comparison with the approach of [14]. The second case study, a random robot, is (to our taste) a nice example showing the need for DTA objectives. We use it for 1-clock as well as multi-clock objectives. The specifications of this example cannot be expressed using any other currently available techniques. The final case study originates from systems biology and is a more realistic case study. We first present experimental results using a sequential algorithm, with

and without lumping. Section 4.4 presents the results when parallelizing the transient analysis (but not the lumping). All results (one and four cores) have been obtained on a 2×2.33GHz Intel Dual-Core computer with 32GB main memory. The experiments on 20 cores have been obtained using a cluster of five such computers with a GigaBit connection. All the results are obtained with precision 10^{-8}.

4.1 Cyclic polling server

This case study facilitates a comparison with [3]. The cyclic polling server (CPS) system [16] is a queuing system consisting of a server and N stations each equipped with a queue of capacity 1, cf. Fig. 3 for $N = 3$. Jobs arrive with rate λ and the server polls the stations in a round-robin order with rate γ. When the server is polling a station with a full queue, it can either serve the job in the queue or it can poll the next station (both with rate γ). Once the server decides to serve a job, it can successfully process the job with rate μ or it will fail with rate ρ. The 1-clock DTA objective (adapted from [3], see

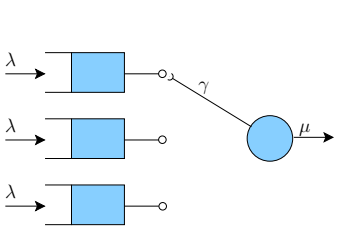


Fig. 3. Cyclic polling server (λ=$\frac{\mu}{N}$, μ=0.5, γ=10, ρ=1)

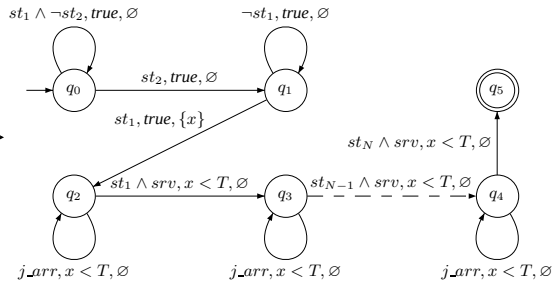


Fig. 4. DTA for the polling server system ($T = 1$)

Fig. 4) requires that after consulting all queues for one round, the server should serve each queue one after the other within T time units. The label st_i indicates that the system is at station i; srv means that the system is serving the job in the current station and j_arr means a new job arrives at some station. The DTA starts from station 1 at q_0 and goes to q_1 when polling the next station (st_2). It stays at q_1 for not polling station 1 –implicitly it goes sequentially from station 2 to N– until it sees station 1 again (and goes to state q_2). Note that the clock is reset before going to state q_2. From state q_2 to state q_5, it specifies serving stations $1, \ldots, N$ one by one within the deadline T. The dashed line indicates the intermediate transitions from station st_2 to station st_{N-1}.

Table 1 summarizes our results, where %transient and %lumping indicate the fraction of time to compute transient probabilities and to lump all CTMCs, respectively. The computed probabilities *Prob*($\mathcal{C} \models \mathcal{A}$) are identical to [3] (that contains results up to N=7); our verification times are, however, three orders of magnitude faster. If lumping is not applied, then most of the time is spent on the transient analysis. Lumping can save approximately $\frac{2}{3}$ of the state space ($\frac{\text{\#blocks}}{\text{\#product states}}$), however, it has a major impact on the verification times.

#queues	#CTMC	No lumping			With lumping			
N	states	#product states	time(s)	%transient	#blocks	time(s)	%transient	%lumping
2	16	51	0	0%	21	0	0%	0%
3	48	143	0.01	0%	52	0.02	0%	60%
4	128	363	0.03	60%	126	0.08	13%	65%
5	320	875	0.13	84%	298	0.37	16%	79%
6	768	2043	0.57	88%	690	1.75	15%	82%
7	1792	4667	2.6	90%	1570	8.58	14%	84%
8	4096	10491	14.8	94%	3522	41.92	13%	85%
9	9216	23291	120	98%	7810	230	22%	77%
10	20480	51195	636	98%	17154	1381	25%	75%

Table 1. Experimental results for polling server system (no parallelization)

4.2 Robot navigation

A robot moves on a grid with $N \times N$ cells, see Fig. 5. It can move up, down, left and right with rate λ. The black squares from the grid represent walls, i.e., the robot is prohibited to pass through them. The robot is allowed to stay in consecutive C-cells for at most T_1 units of time, and in the D-cells for at most T_2 units of time. There is no time constraint on the residence times in the A-cells. The task is to compute the probability to reach the B-cell from the A-cell labeled with $\nearrow$. No time constraint is imposed on reaching the target. The DTA objective is shown in Fig. 6. Intuitively, q_A, q_B, q_C and

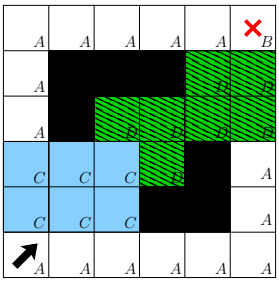


Fig. 5. Robot on a grid ($\lambda = 1$)

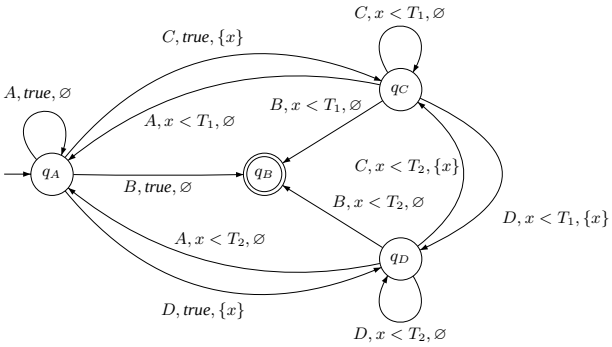


Fig. 6. DTA for robot case study ($T_1 = 3$, $T_2 = 5$)

q_D represent the states that the robot is in the respective cells. From q_A, q_C, and q_D, it is possible to go to the final state q_B. The outgoing edges from q_C and q_D have the guard $x < T_1$ or $x < T_2$; while their incoming edges reset the clock x.

Table 2 presents the results. Lumping is attractive as it reduces the state space by a factor two, and speeds-up the verification. As opposed to the polling system case, most time is spent on building the product and solving the linear equation system. The gray rows in Table 1 and 2 refer to similar product size whereas the verification times differ by two orders of magnitude (14.88 vs. 2433.31). This is due to the fact that there are two and three subgraphs, respectively. The resulting linear equation system has 2 and 3 variables accordingly and this influences the verification times. The number of blocks is not monotonically increasing, as the robot grid (how the walls and regions C and D are located) is randomly generated. The structure of the grid, e.g., whether it is symmetric or not, has a major influence on the lumping time and quotient size.

N	#CTMC states	No lumping			With lumping			
		#product states	time(s)	%transient	#blocks	time(s)	%transient	%lumping
10	100	148	0.09	59%	78	0.09	43%	32%
20	400	702	6.7	18%	380	7.1	14%	7%
30	900	1248	32	17%	619	26	14%	6%
40	1600	2672	119	13%	1296	93	10%	5%
50	2500	4174	135	17%	2015	138	12%	7%
60	3600	4232	309	16%	1525	261	12%	7%
70	4900	8661	904	12%	4212	1130	7%	3%
80	6400	9529	1753	12%	4339	1429	14%	4%
90	8100	9812	2433	8%	2613	1922	6%	5%

Table 2. Experimental results for the robot example (without parallelization)

4.3 Systems biology

The last case study stems from a real example [15] in systems biology. The goal is to generate activated messengers. M ligands can bind with a number of receptors, cf. Fig. 7). Initially each ligand binds with a free receptor R with rate k_{+1} and it forms a ligand-receptor (LR) B_0. The LR then undergoes a sequence of N modifications with a constant rate k_p and becomes $B_1, \ldots, B_N$. From every LR B_i ($0 \leqslant i \leqslant N$) the ligand can separate from the receptor with rate k_{-1}. The LR B_N can link with an inactive messenger with rate k_{+x} and then forms a new component ligand-receptor-messenger (LRM) (the last B_N in Fig. 7). The LRM can decompose into three separate components in their initial forms with rate k_{-1}, or the messenger can separate from LRM into an inactive (resp. active) messenger with rate k_{-x} (resp. k_{cat}).

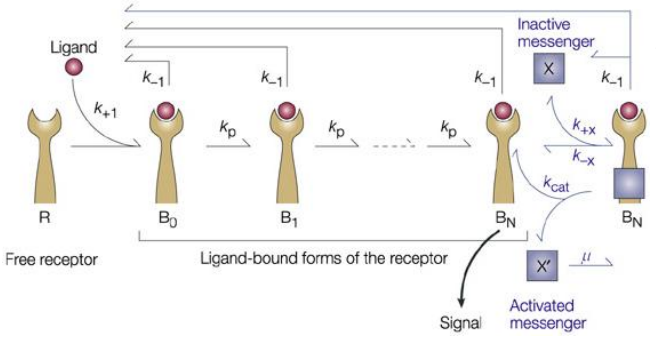


Fig. 7. Kinetik proof reading with a messenger [15] (#$R = 900$, #Inact. msg $= 10000$, $N = 6$, $k_{+1} = 6,7 \times 10^{-3}$, $k_{-1} = 0,5$, $k_p = 0,25$, $k_{+x} = 1,2 \times 10^{-3}$, $k_{-x} = 0,01$, $k_{cat} = 100$)

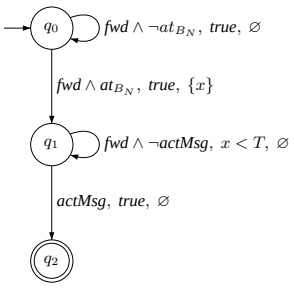


Fig. 8. DTA for the biology case study ($T = 1$)

The 1-clock DTA objective is given in Fig. 8. Intuitively, it requires a transformation from R to LR B_N directly without jumping back to R inbetween and manage to activate a messenger within T time units after reaching the LR B_N. In the DTA, *fwd* means that the last transformation is moving forward, i.e., not jumping back to R; *at* $_{B_N}$ means that the process reaches LR B_N and *actMsg* means the active messenger is generated. In q_0, the process is on the way to reach LR B_N. When it reaches LR B_N, the DTA goes from q_0 to q_1 and resets the clock x. In q_1, there is no active messenger generated yet. It will go to q_2 when an active messenger is generated. Note that the time constraint $x < T$ is

	#CTMC	No lumping			With lumping			
M	states	#product states	time(s)	%transient	#blocks	time(s)	%transient	%lumping
1	18	31	0	0%	13	0	0%	0%
2	150	203	0.06	93%	56	0.05	58%	39%
3	774	837	1.36	94%	187	0.84	64%	30%
4	3024	2731	17.29	97%	512	9.19	73%	24%
5	9756	7579	152.54	97%	1213	73.4	76%	21%
6	27312	18643	1547.45	98%	2579	457.35	78%	20%
7	68496	41743	11426.46	99%	5038	3185.6	85%	14%
8	157299	86656	23356.5	99%	9200	11950.8	81%	18%
9	336049	169024	71079.15	99%	15906	38637.28	76%	22%
10	675817	312882	205552.36	99%	26256	116314.41	71%	26%

Table 3. Experimental results for the biology example (no parallelization)

checked on the self-loop on q_1. As Table 3 indicates, lumping works very well on this example: it reduces both time and space by almost one order of magnitude.

4.4 Parallelization

Model checking a CTMC against a 1-clock DTA can be parallelized in a natural manner. In this section, we present the results when parallelization is applied on the above three case studies. We experimented on distributing the tasks on 1 machine with 4 cores as well as 5 machines with 4 cores each (20 cores in total). We focused on parallelizing the transient analysis; as lumping is not parallelized, we determine the speedup without lumping. For each CTMC $\mathcal{C}_i$, we need to compute for each state its transient probability vector which corresponds to a column in the transient probability matrix. We distribute the computation of different columns to different cores (which might be on different machines). To do so, we launch N different processes and send them the rate matrix and a list of initial states, and each process returns the transient probability vector for each initial state. The speedup factor is computed by $\frac{\text{time without para.}}{\text{time with para.}}$. From Table 4, it follows that parallelization mostly works well for larger models. For small models, it usually does not pay off to distribute the computation tasks, due to overhead. For the polling server system, as the number of stations N increases, the value of $\frac{\text{speedup for 20 cores}}{\text{speedup for 4 cores}}$ approximates 20/4=5. The same applies to the biology example. The parallelization does not work well on the robot example. The performance on 20 cores might even be worse than on 4 cores. This is due to the fact that only the transient analysis is parallelized. From Table 1 and 2, we can see that most of the computation time is spent on transient analysis for the polling and biology examples. This explains why parallelization works well here. In the robot example, however, the transient analysis does not dominate the computation time. This yields moderate speedups. An interesting future work is to apply parallel lumping as in [10] to our setting.

5 Multi-Clock DTA Objectives

The graph decomposition approach for 1-clock DTA fails in the *multi*-clock setting as in case of a reset edge, it cannot be determined to which time point it will jump (unlike $x = 0$ in the 1-clock case). These time points, however, can be approximated by discretization as shown below. W.l.o.g. assume there are two clocks x, y. The maximal

N	4 Cores		20 Cores	
	time(s)	speedup	time(s)	speedup
4	0.03	1.21	0.18	0.18
5	0.08	1.70	0.22	0.59
6	0.32	1.77	1.54	0.37
7	1.04	2.58	2.08	1.29
8	7.35	2.02	4.04	3.68
9	40.28	2.98	13.76	8.73
10	186.02	3.42	54.97	11.58
11	863.3	3.35	233.99	12.35
12	3940.42	3.65	1089	13.22

N	4 Cores		20 Cores	
	time(s)	speedup	time(s)	speedup
30	23.59	1.37	27.18	1.19
40	84.05	1.42	81.64	1.47
50	122.01	1.11	117.17	1.16
60	266.67	1.16	265.48	1.17
70	793.48	1.14	778.69	1.16
80	1474.88	1.19	1441.99	1.22
90	2498.34	0.97	1917.1	1.27
100	1667.78	1.14	1342.26	1.41
110	4614.92	1.32	5165.7	1.18

N	4 Cores		20 Cores	
	time(s)	speedup	time(s)	speedup
3	0.45	3.03	0.42	3.22
4	5.3	3.26	3.44	5.02
5	44.73	3.41	15.87	9.61
6	620.16	2.50	160.58	9.64
7	4142.19	2.76	949.32	12.04
8	8168.62	2.86	1722.63	13.56
9	23865.17	2.98	5457.01	13.03
10	70623.46	2.91	16699.22	12.31

Table 4. Parallel verification of polling (left), robot (mid), and biology example (right)

constants c_x, c_y to which x and y are compared in the DTA are discretized by $h = \frac{1}{N}$ for some a priori fixed $N \in \mathbb{N}_{>0}$. As a result, there are $c_x c_y / h^2 = N^2 c_x c_y$ areas (or grids). The behavior of all the points in one grid can be regarded as approximately the same. For grid (i, j), it can either delay to $(i{+}1, j{+}1)$, or jump to $(0, j)$ (resp. $(i, 0)$) by resetting clock x (resp. y) or to $(0, 0)$ by resetting both clocks. The following definition originates from [12].

Definition 7 (Product of CTMC and DTA). *Let $\mathcal{C} = (S, \mathrm{AP}, L, s_0, \mathbf{P}, E)$ be a* CTMC *and $\mathcal{A} = (2^{\mathrm{AP}}, \mathcal{X}, Q, q_0, Q_F, \rightarrow)$ be a* DTA. *Let $\mathcal{C} \otimes \mathcal{A} = (Loc, \mathcal{X}, \ell_0, Loc_F, E, \rightsquigarrow)$, where $Loc = S \times Q$; $\ell_0 = (s_0, q_0)$; $E(s, q) = E(s)$; $Loc_F := S \times Q_F$, and $\rightsquigarrow$ is the smallest relation defined by:*

$$\frac{\mathbf{P}(s, s') > 0 \ \wedge \ q \xrightarrow{L(s), g, X} q'}{s, q \overset{g,X}{\rightsquigarrow} \pi} \text{ such that } \pi(s', q') = \mathbf{P}(s, s').$$

Note that $\pi \in Distr(S \times Q)$ is a probability distribution. The symbolic edge $\ell \overset{g,X}{\rightsquigarrow} \pi$ with distribution π induces transitions of the form $\ell \overset{g,X}{\underset{p}{\longmapsto}} \ell'$ with $p = \pi(\ell')$. $\mathcal{C} \otimes \mathcal{A}$ is a stochastic process that can be equipped with a probability measure on infinite paths, cf. [12]. We approximate the stochastic behavior by a discrete-time Markov chain (DTMC). This is done by discretizing clock values in equidistant step-sizes of size $h = 1/N$.

Definition 8 (Discretization). *Let $\mathcal{C} \otimes \mathcal{A} = (Loc, \mathcal{X}, \ell_0, Loc_F, E, \rightsquigarrow)$, $k = |\mathcal{X}|$, C be the largest constant in $\mathcal{A}$, $\boldsymbol{C} = C^k$ be a vector with all elements equal to C, $h = 1/N$ for some $N \in \mathbb{N}_{>0}$ and $\boldsymbol{h} = h^k$. The (unlabeled) DTMC $\mathcal{D}_h = (S, s_0, \mathbf{P})$ is given by $S = Loc \times (\{i \cdot h \mid 0 \leqslant i \leqslant C{\cdot}N{+}1\})^k$, $s_0 = (\ell_0, \mathbf{0})$ and $\mathbf{P}((\ell, \eta), (\ell', \eta'))$ is given by:*

- $p \cdot \left(1 - \frac{1 - e^{-E(\ell) \cdot h}}{h \cdot E(\ell)}\right)$ iff $\ell \overset{g,X}{\underset{p}{\longmapsto}} \ell' \ \wedge \ \eta \models g \wedge \ \eta' = \eta[X := 0] \wedge \ \eta \neq \boldsymbol{C}{+}\boldsymbol{h}$;
- $p \cdot \left(\frac{1}{hE(\ell)} - (1 + \frac{1}{hE(\ell)}) e^{-E(\ell)h}\right)$ iff $\ell \overset{g,X}{\underset{p}{\longmapsto}} \ell' \ \wedge \eta \models g \wedge \eta' = (\eta + h)[X := 0] \wedge \eta \neq \boldsymbol{C}{+}\boldsymbol{h}$;
- $e^{-E(\ell) \cdot h}$ iff $\ell = \ell' \wedge \eta' = \eta + h \wedge \ \eta \neq \boldsymbol{C}{+}\boldsymbol{h}$;
- p iff $\ell \overset{g,X}{\underset{p}{\longmapsto}} \ell' \ \wedge \eta \models g \wedge \ \eta' = \eta[X := 0] \wedge \ \eta = \boldsymbol{C}{+}\boldsymbol{h}$;
- 0 *otherwise.*

The number of states in the derived DTMC is $|S|\cdot|Q|\cdot(C\cdot N+1)^k$. The following result states that for h approaching zero, *Prob*($\mathcal{C} \models \mathcal{A}$) equals the reachability probability in the DTMC $\mathcal{D}_h$ of a state of the form $(\ell,\cdot)$ with $\ell \in Loc_F$.

Theorem 4. *Let $P_h^{\mathcal{D}}(s_0, \Diamond F)$ be the reachability probability in the* DTMC $\mathcal{D}_h$ *to reach a state $(\ell,\cdot)$ with $\ell \in Loc_F$. Then* $\lim_{h\to 0} P_h^{\mathcal{D}}(s_0, \Diamond F) =$ *Prob*($\mathcal{C} \models \mathcal{A}$).

To illustrate the performance of the discretization technique, we add a clock y to the robot example, where y is never reset. The time constraint $y < T_y$ is added to all incoming edges of q_B. The results are shown in Table 5 with $h{=}5 \cdot 10^{-2}$. Although the resulting DTMC size is quite large, the computation times are still acceptable, as computing reachability probabilities in a DTMC is rather fast. For the sake of comparison, we applied the discretization technique to the polling server system with a 1-clock DTA objective. We obtain the same probabilities as before; results are given in Table 6 with precision 0.001.

N	CTMC size	DMTA size	DTMC size	time (s)
10	100	105	865305	79
20	400	475	3914475	412
30	900	1003	8265723	868
40	1600	1669	13754229	1605
50	2500	2356	19415796	2416
60	3600	3411	28110051	3559
70	4900	4850	39968850	22427

Table 5. Experimental results for the robot example with 2-clock DTA ($T_y = 3$)

N	CTMC size	DMTA size	DTMC size	time (s)
2	16	31	31031	15.89
3	48	89	89089	52.66
4	128	229	229229	152.2
5	320	557	557557	407.4
6	768	1309	1310309	1042
7	1792	3005	3008005	2577
8	4096	6781	6787781	6736
9	9216	15101	15116101	30865

Table 6. Polling example with 1-clock DTA using discretization

6 Conclusion

We have presented a practical approach to verifying CTMCs against DTA objectives. First, we showed that a standard region construction on DTA suffices. For 1-clock DTA, we showed that the graph decomposition approach of [12] offers rather straightforward possibilities for optimizations, viz. bisimulation minimization and parallelization. Several experiments substantiate this claim. The main result of this paper is that it shows that 1-clock DTA objectives can be handled by completely standard means: region construction of timed automata, transient analysis of CTMCs, graph analysis, and solving linear equation systems. Our approach clearly outperforms alternative techniques for CSL$^{\text{TA}}$ [3, 14], and allows for the verification of objectives that cannot be treated with other CTMC model checkers. Our prototype is available at http://moves.rwth-aachen.de/CoDeMoC. Finally, we remark that although we only considered finite acceptance conditions in this paper, our approach can easily be extended to DTA with Rabin acceptance conditions.

Acknowledgement We thank Verena Wolf (Saarland University) for providing us with the biology case study. This research is funded by the DFG research training group 1295 AlgoSyn, the SRO DSN project of CTIT, University of Twente, the EU FP7 project QUASIMODO and the DFG-NWO ROCKS project.

References

1. PRISM website. http://www.prismmodelchecker.org.
2. R. Alur and D. L. Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.
3. E. G. Amparore and S. Donatelli. Model checking CSL^{TA} with deterministic and stochastic Petri Nets. In *Dependable Systems and Networks (DSN)*, pages 605–614, 2010.
4. C. Baier, N. Bertrand, P. Bouyer, T. Brihaye, and M. Grösser. Almost-sure model checking of infinite paths in one-clock timed automata. In *LICS*, pages 217–226, 2008.
5. C. Baier, L. Cloth, B. R. Haverkort, M. Kuntz, and M. Siegle. Model checking Markov chains with actions and state labels. *IEEE TSE*, 33(4):209–224, 2007.
6. C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE TSE*, 29(6):524–541, 2003.
7. C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Performance evaluation and model checking join forces. *Commun. of the ACM*, 53(9):74–85, 2010.
8. B. Bérard, A. Petit, V. Diekert, and P. Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fund. Inf.*, 36(2-3):145–182, 1998.
9. N. Bertrand, P. Bouyer, T. Brihaye, and N. Markey. Quantitative model-checking of one-clock timed automata under probabilistic semantics. In *QEST*, pages 55–64, 2008.
10. S. Blom, B. R. Haverkort, M. Kuntz, and J. van de Pol. Distributed Markovian bisimulation reduction aimed at CSL model checking. *ENTCS*, 220(2):35–50, 2008.
11. T. Brazdil, J. Kreal, J. Kretinsky, A. Kucera, and V. Rehak. Stochastic real-time games with qualitative timed automata objectives. In *CONCUR*, LNCS, pages 207–221, 2010.
12. T. Chen, T. Han, J.-P. Katoen, and A. Mereacre. Quantitative model checking of continuous-time Markov chains against timed automata specification. In *LICS*, pages 309–318, 2009.
13. S. Derisavi, H. Hermanns, and W. H. Sanders. Optimal state-space lumping in Markov chains. *Inf. Process. Lett.*, 87(6):309–315, 2003.
14. S. Donatelli, S. Haddad, and J. Sproston. Model checking timed and stochastic properties with CSL^{TA}. *IEEE TSE*, 35(2):224–240, 2009.
15. B. Goldstein, J. R. Faeder, and W. S. Hlavacek. Mathematical and computational models of immune-receptor signalling. *Nat. Reviews Immunology*, 4:445–456, 2004.
16. B. R. Haverkort. Performance evaluation of polling-based communication systems using SPNs. In *Appl. of Petri Nets to Comm. Networks*, pages 176–209, 1999.
17. J.-P. Katoen, E. M. Hahn, H. Hermanns, D. N. Jansen, and I. Zapreev. The ins and outs of the probabilistic model checker MRMC. In *QEST*, pages 167–176, 2009.
18. J.-P. Katoen, T. Kemna, I. Zapreev, and D. Jansen. Bisimulation minimisation mostly speeds up probabilistic model checking. In *TACAS*, LNCS 4424, pages 87–101, 2007.
19. A. Valmari and G. Franceschinis. Simple $O(m \log n)$ time Markov chain lumping. In *TACAS*, LNCS 6015, pages 38–52, 2010.
20. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338, 1985.