# Bisimulation minimisation mostly speeds up probabilistic model checking

Joost-Pieter Katoen[1,2], Tim Kemna[2], Ivan Zapreev[1,2] and David N. Jansen[2,1]

[1] Software Modeling and Verification Group, RWTH Aachen, Germany
[2] Formal Methods and Tools, University of Twente, The Netherlands

**Abstract.** This paper studies the effect of bisimulation minimisation in model checking of monolithic discrete-time and continuous-time Markov chains as well as variants thereof with rewards. Our results show that—as for traditional model checking—enormous state space reductions (up to logarithmic savings) may be obtained. In contrast to traditional model checking, in many cases, the verification time of the original Markov chain exceeds the quotienting time plus the verification time of the quotient. We consider probabilistic bisimulation as well as versions thereof that are tailored to the property to be checked.

## 1 Introduction

Probabilistic model checking enjoys a rapid increase of interest from different communities. Software tools such as PRISM [24] (with about 4,000 downloads), MRMC [29], and LiQuor [5] support the verification of Markov chains or variants thereof that exhibit nondeterminism. They have been applied to case studies from areas such as randomised distributed algorithms, planning and AI, security, communication protocols, biological process modeling, and quantum computing. Probabilistic model checking engines have been integrated in existing tool chains for widely used formalisms such as stochastic Petri nets [10], Statemate [8], and the stochastic process algebra PEPA [23], and are used for a probabilistic extension of Promela [5].

The typical kind of properties that can be checked is time-bounded reachability properties—"Does the probability to reach a certain set of goal states (by avoiding bad states) within a maximal time span exceed $\frac{1}{2}$?"—and long-run averages—"In equilibrium, does the likelihood to leak confidential information remain below $10^{-4}$?" Extensions for cost-based models allow for checking more involved properties that refer to e. g., the expected cumulated cost or the instantaneous cost rate of computations. Intricate combinations of numerical or simulation techniques for Markov chains, optimisation algorithms, and traditional LTL or CTL model-checking algorithms result in simple, yet very efficient verification procedures. Verifying time-bounded reachability properties on models of tens of millions of states usually is a matter of seconds.

Like in the traditional setting, probabilistic model checking suffers from state space explosion: the number of states grows exponentially in the number of

system components and cardinality of data domains. To combat this problem, various techniques have been proposed in the literature. Variants of binary decision diagrams (multi-terminal BDDs) have been (and still are) successfully applied in PRISM [24] to a range of probabilistic models, abstraction-refinement has been applied to reachability problems in MDPs [11], partial-order reduction techniques using Peled's ample-set method have been generalised to MDPs [18], abstract interpretation has been applied to MDPs [36], and various bisimulation equivalences and simulation pre-orders allow model aggregation prior to model checking, e. g., [7, 39]. Recently proposed techniques include abstractions of probabilities by intervals combined with three-valued logics for DTMCs [14, 25, 26], stochastic ordering techniques for CSL model checking [34], abstraction of MDPs by two-player stochastic games [32], and symmetry reduction [31].

The purpose of this paper is to empirically investigate the effect of strong bisimulation minimisation in probabilistic model checking. We hereby focus on fully probabilistic models such as discrete-time and continuous-time Markov chains (DTMCs and CTMCs, for short), and variants thereof with costs. The advantages of probabilistic bisimulation [33] in this setting are manifold. It preserves the validity of PCTL [19] and CSL [2, 4] formulas, variants of CTL for the discrete- and continuous-time probabilistic setting, respectively. It implies ordinary lumpability of Markov chains [9], an aggregation technique for Markov chains that is applied in performance and dependability evaluation since the 1960s. Quotient Markov chains can be obtained in a fully automated way. The time complexity of quotienting is logarithmic in the number of states, and linear in the number of transitions—as for traditional bisimulation minimisation—when using splay trees (a specific kind of balanced tree) for storing partitions [13]. Besides, probabilistic bisimulation can be used for obtaining (coarser) abstractions that are tailored to the properties of interest (as we will see), and enjoys the congruence property for parallel composition allowing compositional minimisation. We consider explicit model checking as the non-trivial interplay between bisimulation and MTBDDs would unnecessarily complicate our study; such symbolic representations mostly grow under bisimulation minimisation [22].

Thanks to extensive studies by Fisler and Vardi [15–17], it is known that bisimulation minimisation for LTL model checking and invariant verification leads to drastic state space reductions (up to logarithmic savings) but at a time penalty: the time to minimise and model check the resulting quotient Kripke structure significantly exceeds the time to verify the original model. This paper considers these issues in probabilistic (i. e., PCTL and CSL) model checking. To that end, bisimulation minimisation algorithms have been realised in the prototypical explicit-state probabilistic model checker MRMC, several case studies have been considered that are widely studied in the literature (and can be considered as benchmark problems), and have been subjected to various experiments. This paper presents our results. As expected, our results show that enormous state space reductions (up to logarithmic savings) may be obtained. In contrast to the results by Fisler and Vardi [15–17], the verification time of the original Markov chain mostly *exceeds* the quotienting time plus the verification

time of the quotient. This effect is stronger for probabilistic bisimulation that is tailored to the property to be checked and for model checking Markov chains with costs (i. e., rewards). This is due to the fact that probabilistic model checking is more time-consuming than traditional model checking, while minimization w. r. t. probabilistic bisimulation is only slightly slower than for traditional bisimulation.

The paper is organised as follows. Section 2 introduces the considered probabilistic models. Section 3 considers probabilistic bisimulation and the algorithms used. Section 4 presents the considered case studies, the obtained results, and analyses these results. Section 5 concludes the paper.

## 2 Preliminaries

*DTMCs.* Let $AP$ be a fixed, finite set of *atomic propositions.* A (labelled) DTMC $\mathcal{D}$ is a tuple $(S, \mathbf{P}, L)$ where $S$ is a finite set of *states,* $\mathbf{P} : S \times S \to [0,1]$ is a *probability matrix* such that $\sum_{s' \in S} \mathbf{P}(s,s') = 1$ for all $s \in S$, and $L : S \to 2^{AP}$ is a *labelling* function which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions that hold in $s$. A path through a DTMC is a sequence[1] of states $\sigma = s_0\, s_1\, s_2 \ldots$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i$. Let $Path^{\mathcal{D}}$ denote the set of all paths in DTMC $\mathcal{D}$. $\sigma[i]$ denotes the $(i{+}1)$th state of $\sigma$, i. e., $\sigma[i] = s_i$.

*The logic PCTL.* Let $a \in AP$, probability $p \in [0,1]$, $k \in \mathbb{N}$ (or $k = \infty$) and $\bowtie$ be either $\leq$ or $\geq$. The syntax of Probabilistic CTL (PCTL) [19] is defined by:

$$\Phi ::= \mathrm{tt} \;\Big|\; a \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg\, \Phi \;\Big|\; \mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{\leq k}\, \Phi).$$

A state $s$ satisfies $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}^{\leq k}\ \Psi)$ if $\{\, \sigma \in Path^{\mathcal{D}}(s) \mid \sigma \models \Phi\ \mathcal{U}^{\leq k}\ \Psi \,\}$ has a probability that satisfies $\bowtie p$. A path $\sigma$ satisfies $\Phi\, \mathcal{U}^{\leq k}\, \Psi$ if within $k$ steps a $\Psi$-state is reached, and all preceding states satisfy $\Phi$. That is, if $\sigma[j] \models \Psi$ for some $j \leq k$, and $\sigma[i] \models \Phi$ for all $i < j$. We define the abbreviation $\Diamond^{\leq k}\Phi := \mathrm{tt}\, \mathcal{U}^{\leq k}\, \Phi$. The unbounded until formula that is standard in temporal logics is obtained by taking $k = \infty$, i. e., $\Phi\, \mathcal{U}\, \Psi = \Phi\, \mathcal{U}^{\leq \infty}\, \Psi$.[2]

Given a set $F$ of PCTL formulas, we denote with $\mathrm{PCTL}_F$ the smallest set of formulas that contains $F$ and is closed under the PCTL operators $\wedge$, $\neg$, and $\mathcal{U}$.

*Verifying hop-constrained probabilistic reachability.* PCTL model checking [19] is carried out in the same way as verifying CTL by recursively computing the set $Sat(\Phi) = \{\, s \in S \mid s \models \Phi \,\}$. The probability of $\{\, \sigma \mid \sigma \models \Phi\ \mathcal{U}^{\leq k}\ \Psi \,\}$ is the least solution of the following linear equation system. Let $S_1 = \{\, s \mid s \models \Psi \,\}$, $S_0 = \{\, s \mid s \models \neg\Phi \wedge \neg\Psi \,\}$, and $S_? = \{\, s \mid s \models \Phi \wedge \neg\Psi \,\} = S \setminus (S_1 \cup S_0)$.

$$Prob^{\mathcal{D}}(s, \Phi\, \mathcal{U}^{\leq k}\, \Psi) = \begin{cases} 1 & \text{if } s \in S_1 \\ \sum\limits_{s' \in S} \mathbf{P}(s,s') \cdot Prob^{\mathcal{D}}(s', \Phi\, \mathcal{U}^{\leq k-1}\, \Psi) & \text{if } s \in S_? \wedge k > 0 \\ 0 & \text{otherwise} \end{cases}$$

[1] In this paper, we do not dwell upon the distinction between finite and infinite paths.
[2] For simplicity, we do not consider the next operator.

One can simplify this system by replacing $S_0$ by $U_0 = S_0 \cup \{ s \in S_? \mid \neg\exists \sigma \in Path^{\mathcal{D}}(s) : \sigma \models \Phi \,\mathcal{U}\, \Psi \}$. If $k = \infty$, one may also replace $S_1$ by $U_1 = S_1 \cup \{ s \in S_? \mid \forall \sigma \in Path^{\mathcal{D}}(s) : \sigma \models \Phi \,\mathcal{U}\, \Psi \}$. The sets $U_0$ and $U_1$ can be found via a simple graph analysis (a depth-first search) in time $O(|S|+|\mathbf{P}|)$.

Alternatively, the probabilities can be calculated by making the states $s \notin S_?$ absorbing as follows. For DTMC $\mathcal{D} = (S, \mathbf{P}, L)$ and $A \subseteq S$, let $\mathcal{D}[A]$ be the DTMC $(S, \mathbf{P}[A], L)$ where the states in $A$ are made absorbing: If $s \in A$, then $\mathbf{P}[A](s,s) = 1$ and $\mathbf{P}[A](s,s') = 0$ for $s' \neq s$. Otherwise, $\mathbf{P}[A](s,s') = \mathbf{P}(s,s')$. Let $\pi^{\mathcal{D}}(s \overset{k}{\rightsquigarrow} s')$ denote the probability of being in state $s'$ after exactly $k$ steps in DTMC $\mathcal{D}$ when starting in $s$. Then:

$$Prob^{\mathcal{D}}(s, \Phi \,\mathcal{U}^{\leq k}\, \Psi) = \sum_{s' \in S_1} \pi^{\mathcal{D}[S_0 \cup S_1]}(s \overset{k}{\rightsquigarrow} s').$$

Calculating $Prob^{\mathcal{D}}(s, \Phi \,\mathcal{U}^{\leq k}\, \Psi)$ thus amounts to computing $(\mathbf{P}[S_0 \cup S_1])^k \cdot \underline{\iota_{S_1}}$, where $\underline{\iota_{S_1}}(s) = 1$ if $s \in S_1$, and 0 otherwise.

*CTMCs.* A (labelled) CTMC $\mathcal{C}$ is a tuple $(S, \mathbf{P}, E, L)$ where $(S, \mathbf{P}, L)$ is a DTMC and $E : S \to \mathbb{R}_{\geq 0}$ provides the *exit rate* for each state. The probability of taking a transition from $s$ within $t$ time units equals $1 - e^{-E(s) \cdot t}$. The probability of taking a transition from state $s$ to state $s'$ within time $t$ is given by: $\mathbf{P}(s,s') \cdot (1 - e^{-E(s) \cdot t})$.

A path through a CTMC is a sequence of states and sojourn times $\sigma = s_0\, t_0\, s_1\, t_1 \ldots$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ and $t_i \in \mathbb{R}_{\geq 0}$ for all $i$. Let $Path^{\mathcal{C}}$ denote the set of all paths in CTMC $\mathcal{C}$.

*Uniformisation.* In a *uniform* CTMC, the exit rate of all states is the same. A non-uniform CTMC can be uniformized by adding self loops as follows: let $\mathcal{C} = (S, \mathbf{P}, E, L)$ be a CTMC and choose $\tilde{E} \geq \max_{s \in S} E(s)$. Then, $Unif_{\tilde{E}}(\mathcal{C}) = (S, \mathbf{P}', E', L)$ where $E'(s) = \tilde{E}$ for all $s$, $\mathbf{P}'(s,s') = E(s)\mathbf{P}(s,s')/\tilde{E}$ if $s \neq s'$ and $\mathbf{P}'(s,s) = 1 - \sum_{s' \neq s} \mathbf{P}'(s,s')$. The probability to be in a given state at a given time in the uniformized CTMC is the same as the one in the original CTMC.

*The logic CSL.* Continuous stochastic logic (CSL, [4]) is similar to PCTL. For $a$, $p$ and $\bowtie$ as before, time bounds $t_1 \in [0, \infty)$ and $t_2 \in [t_1, \infty]$, the syntax is:

$$\Phi ::= \mathrm{tt} \;\Big|\; a \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg\Phi \;\Big|\; \mathcal{P}_{\bowtie p}(\Phi \,\mathcal{U}^{[t_1,t_2]}\, \Phi) \;\Big|\; \mathcal{S}_{\bowtie p}(\Phi)$$

A state $s$ satisfies $\mathcal{P}_{\bowtie p}(\Phi \,\mathcal{U}^{[t_1,t_2]}\, \Psi)$ if the set of timed paths $\{ \sigma \in Path^{\mathcal{C}}(s) \mid \sigma \models \Phi \,\mathcal{U}^{[t_1,t_2]}\, \Psi \}$ has a probability $\bowtie p$. A timed path satisfies $\Phi \,\mathcal{U}^{[t_1,t_2]}\, \Psi$ if within time $t \in [t_1, t_2]$ a $\Psi$-state is reached, and all preceding states satisfy $\Phi$. We will mostly let $t_1 = 0$ and denote this as $\Phi \,\mathcal{U}^{\leq t_2}\, \Psi$. A state $s$ satisfies the formula $\mathcal{S}_{\bowtie p}(\Phi)$ if the steady-state probability to be in a $\Phi$-state (when starting in $s$) satisfies the constraint $\bowtie p$.

CSL model checking [2, 4] can be implemented as follows. The operator $\mathcal{S}$ can be solved by a (standard) calculation of the steady-state probabilities together

with a graph analysis. For the time-bounded until operator, note that, after uniformisation the probability to take $k$ steps within time $t$ does not depend on the actual states visited. This probability is Poisson distributed, and the probability to satisfy the until formula within $k$ steps is calculated using the PCTL algorithm. The total probability is an infinite sum over all $k$, which can be approximated well.

*Rewards.* A discrete-time Markov reward model (DMRM) $\mathcal{D}_r$ is a tuple $(\mathcal{D}, r)$ where $\mathcal{D}$ is a DTMC and $r : S \to \mathbb{R}_{\geq 0}$ is a *reward* assignment function. The quantity $r(s)$ indicates the reward that is earned on leaving state $s$. Rewards could also be attached to edges in a DTMC, but this does not increase expressivity. A path through a DMRM is a path through its DTMC, i. e., sequence of states $\sigma = s_0\, s_1\, s_2 \ldots$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i$.

Let $a$, $p$ and $k$ be as before, and $r \in \mathbb{R}_{\geq 0}$ be a nonnegative reward bound. The two main operators that extend PCTL to Probabilistic Reward CTL (PRCTL) [1] are $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}^{\leq k}_{\leq r}\ \Psi)$ and $\mathcal{E}^{=k}_{\leq r}(\Phi)$. The until-operator is equipped with a bound on the maximum number ($k$) of allowed hops to reach the goal states, and a bound on the maximum allowed cumulated reward ($r$) before reaching these states. Formula $\mathcal{E}^{=k}_{\leq r}(\Phi)$ asserts that the expected cumulated reward in $\Phi$-states until the $k$-th transition is at most $r$. Thus, in order to check the validity of this formula for a given path, all visits to $\Phi$-state are considered in the first $k$ steps and the total reward that is obtained in these states; the rewards earned in other states or earned in $\Phi$-states after the first $k$ steps are not relevant. Whenever the expected value of this quantity over all paths that start in state $s$ is at most $r$, state $s \models \mathcal{E}^{=k}_{\leq r}(\Phi)$.

A continuous-time Markov reward model (CMRM) $\mathcal{C}_r$ is a tuple $(\mathcal{C}, r)$ where $\mathcal{C}$ is a CTMC and $r : S \to \mathbb{R}_{\geq 0}$ is a reward assignment function (as before). The quantity $r(s)$ indicates that if $t$ time units are spent in state $s$, a reward $r(s) \cdot t$ is acquired. A path through a CMRM is a path through its underlying CTMC. Let $\sigma = s_0\, t_0\, s_1\, t_1 \ldots$ be a path. For $t = \sum_{j=0}^{k-1} t_j + t'$ with $t' \leq t_k$ we define $r(\sigma, t) = \sum_{j=0}^{k-1} t_j \cdot r(s_j) + t' \cdot r(s_k)$, the cumulative reward along $\sigma$ up to time $t$.

CSRL [6] is a logic that extends CSL with one operator $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}^{\leq t}_{\leq r}\ \Psi)$ to express time- and reward-bounded properties. Checking this property of a CMRM is difficult. One can either approximate the CMRM by a discretisation of the rewards or compute for each (untimed) path the probability to meet the bound and sum them up. Reward-bounded until properties of a CMRM can be checked via a transformation of rewards into exit rates and checking a corresponding time-bounded until property [6].

## 3 Bisimulation

*Bisimulation.* Let $\mathcal{D} = (S, \mathbf{P}, L)$ be a DTMC and $R$ an equivalence relation on $S$. The quotient of $S$ under $R$ is denoted $S/R$. $R$ is a *strong bisimulation* on $\mathcal{D}$ if for $s_1\ R\ s_2$:

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

$s_1$ and $s_2$ in $\mathcal{D}$ are strongly bisimilar, denoted $s_1 \sim_d s_2$, if there exists a strong bisimulation $R$ on $\mathcal{D}$ with $s_1\ R\ s_2$. Strong bisimulation [9, 23] for CTMCs, that implies ordinary lumpability, is a mild variant of the notion for the discrete-time probabilistic setting: in addition to the above, it is also required that the exit rates of bisimilar states are equal: $E(s_1) = E(s_2)$.

*Measure-driven bisimulation.* Requiring states to be equally labelled with all atomic propositions is rather strong if one is interested in checking formulas that just refer to a (small) subset of propositions, or more generally, sub-formulas. The following notion weakens the labelling requirement in strong bisimulation by requiring equal labellling for a set of PCTL formulas $F$. Let $\mathcal{D} = (S, \mathbf{P}, L)$ be a DTMC and $R$ an equivalence relation on $S$. $R$ is a *$F$-bisimulation* on $\mathcal{D}$ if for $s_1\ R\ s_2$:

$$s_1 \models \Phi \iff s_2 \models \Phi \text{ for all } \Phi \in F \quad \text{and} \quad \mathbf{P}(s_1, C) = \mathbf{P}(s_2, C) \text{ for all } C \in S/R.$$

States $s_1$ and $s_2$ are $F$-bisimilar, denoted $s_1 \sim_F s_2$, if there exists an $F$-bisimulation $R$ on $\mathcal{D}$ with $s_1\ R\ s_2$. $F$-bisimulation on CTMCs (for a set of CSL formulas $F$) is defined analogously [6]. Note that strong bisimilarity is $F$-bisimilarity for $F = AP$.

*Preservation results.* Aziz *et al.* [3] have shown that strong bisimulation is sound and complete with respect to PCTL (and even PCTL$^*$):

**Proposition 1.** *Let $\mathcal{D}$ be a DTMC, $R$ a bisimulation and $s$ an arbitrary state of $\mathcal{D}$. Then, for all PCTL formulas $\Phi$, $s \models_{\mathcal{D}} \Phi \iff [s]_R \models_{\mathcal{D}/R} \Phi$.*

This result can be generalised to $F$-bisimulation in the following way:

**Proposition 2.** *Let $\mathcal{D}$ be a DTMC, $R$ an $F$-bisimulation and $s$ an arbitrary state of $\mathcal{D}$. Then, for all $PCTL_F$ formulas $\Phi$, $s \models_{\mathcal{D}} \Phi \iff [s]_R \models_{\mathcal{D}/R} \Phi$.*

Similar results hold for CSL and bisimulation on CTMCs [4], for PRCTL on DMRM, and for CSRL on CMRM.

*Bisimulation minimisation.* The preservation results suggest that one can verify properties of a Markov chain on a bisimulation quotient. The next issue to consider is how to obtain the quotient. An often used algorithm (called *partition refinement*) is based on *splitting:* Let $\Pi$ be a partition of $S$. A splitter for some block $B \in \Pi$ is a block $Sp \in \Pi$ such that the probability to enter $Sp$ is not the same for each state in $B$. In this case, the algorithm splits $B$ into subblocks such that each subblock consists of states $s$ with identical $\mathbf{P}(s, Sp)$. This step is repeated until a fixpoint is reached. The final partition is the coarsest bisimulation that respects the initial partition. The worst-case time complexity of this algorithm is $O(|\mathbf{P}| \log |S|)$ provided that splay trees are used to store blocks [13]. These data structures are adopted in our implementation.[3]

[3] In practice, an implementation using red-black trees is often slightly faster, although this raises the theoretical complexity to $O(|\mathbf{P}| \log^2 |S|)$, cf. [12, Section 3.4].

*Initial partition.* The choice of initial partition in the partition refinement algorithm determines what kind of bisimulation the result is. If we group states labelled with the same atomic propositions together, the result is the strong bisimulation quotient $S/\sim_d$. If we choose the initial partition according to the satisfaction of formulas in $F$, the resulting partition is the $F$-bisimulation quotient $S/\sim_F$. To get the smallest bisimulation quotient, it is important to start with a coarse initial partition. Instead of only calculating the strong bisimulation quotient, we will also use measure-driven bisimulation for a suitable set $F$.

A naive approach for formula $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}\ \Psi)$ is to choose $F = \{\, \Psi, \Phi \wedge \neg\Psi \,\}$. In fact, $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}\ \Psi)$ is not in $\mathrm{PCTL}_F$, but the equivalent formula $\mathcal{P}_{\bowtie p}(\Phi \wedge \neg\Psi\ \mathcal{U}\ \Psi)$ is. This yields an initial partition consisting of the sets $S_1 = \mathit{Sat}(\Psi)$, $S_? = \mathit{Sat}(\Phi \wedge \neg\Psi)$ and $S_0 = S \setminus (S_1 \cup S_?)$ (cf. Section 2). Note that selecting $F = \{\, \Psi, \Phi \,\}$ would lead to a less efficient initial partition with four blocks instead of three. We improve this initial partition by replacing $S_0$ by $U_0 = \mathit{Sat}(\mathcal{P}_{\leq 0}(\Phi\ \mathcal{U}\ \Psi))$ and $S_1$ by $U_1$, which is essentially[4] $\mathit{Sat}(\mathcal{P}_{\geq 1}(\Phi\ \mathcal{U}\ \Psi))$. (Defining $U_0$ and $U_1$ as satisfaction sets of some formula has the advantage that we can still use Proposition 2.) The sets of states $U_0$ and $U_1$ can be collapsed into single states $u_0$ and $u_1$, respectively. This results in the initial partition $\{\, \{u_0\}, \{u_1\}, S \setminus (U_0 \cup U_1) \,\}$.

For bounded until, one can still use $U_0$, but not $U_1$, since the fact that (almost) all paths satisfy $\Phi\ \mathcal{U}\ \Psi$ does not imply that these paths reach a $\Psi$-state within the step or time bound. Therefore, for this operator the initial partition is $\{\, \{u_0\}, \{s_1\}, S \setminus (U_0 \cup S_1) \,\}$ with $u_0$ as before and $s_1$ the collapsed state for $S_1$.[5] Thus, for bounded until the measure-driven initial partition is finer than for unbounded until. In the experiments reported in the next section, the effect of the granularity of the initial partition will become clear.

## 4 Experiments

To study the effect of bisimulation in model checking, we realised the minimisation algorithms in MRMC and applied them to a variety of case studies, most of which can be obtained from the PRISM webpage.[6] We used PRISM to specify the models and generate the Markov chains. Subsequently, the time and memory requirements have been considered for verifying the chains (by MRMC), and for minimising plus verifying the lumped chain (both by MRMC). All experiments were conducted on a 2.66 GHz Pentium 4 processor with 1 GB RAM running Linux. All reported times are in milliseconds and are obtained by taking the average of running the experiment 10 times.

### 4.1 Discrete time

*Crowds protocol [38].* This protocol uses random routing within a group of nodes (a crowd) to establish a connection path between a sender and a receiver. Rout-

[4] Up to states $s$ where the set $\{\, \sigma \in \mathit{Path}^{\mathcal{D}}(s) \mid \sigma \not\models \Phi\ \mathcal{U}\ \Psi \,\}$ is only almost empty.

[5] For the sake of brevity, we omit the details for the optimal initial partition for time-bounded until-formulas of the form $\mathcal{U}^{[t_1,t_2]}$ with $0 < t_1$.

[6] see `http://www.cs.bham.ac.uk/dxp/prism/index.php`.

ing paths are reconstructed once the crowd changes; the number of such new route establishments is $R$, and is an important parameter that influences the state space. Random routing serves to hide the secret identity of a sender. The table below summarises the results for checking $\mathcal{P}_{\leq p}(\Diamond \mathit{observe})$ where *observe* characterises a situation in which the sender's id is detected. The parameter $N$ in the first column is the number of honest crowd members; our models include $N/5$ dishonest members. The second column shows parameter $R$. The next three columns indicate the size of the state space of the DTMC (i. e., $|S|$), the number of transitions (i. e., the number of non-zero entries in $\mathbf{P}$), and the verification time. The next three columns indicate the number of states in the quotient DTMC, the time needed for obtaining this quotient, and the time to check the validity of the same formula on the quotient. The last two columns indicate the reduction factor for the number of states and total time. Note that we obtain large state space reductions. Interestingly, in terms of time consumption, quotienting obtains a reduction in time of about a factor 4 to 7.

| | | original DTMC | | | lumped DTMC | | | red. factor | |
|---|---|---|---|---|---|---|---|---|---|
| $N$ | $R$ | states | transitions | ver. time | blocks | lump time | ver. time | states | time |
| 5 | 3 | 1198 | 2038 | 3.2 | 53 | 0.6 | 0.3 | 22.6 | 3.7 |
| 5 | 4 | 3515 | 6035 | 11 | 97 | 2.0 | 0.5 | 36.2 | 4.4 |
| 5 | 5 | 8653 | 14953 | 48 | 153 | 6.0 | 0.9 | 56.6 | 6.9 |
| 5 | 6 | 18817 | 32677 | 139 | 209 | 14 | 1.4 | 90.0 | 9.0 |
| 10 | 3 | 6563 | 15143 | 24 | 53 | 4.6 | 0.2 | 124 | 4.9 |
| 10 | 4 | 30070 | 70110 | 190 | 97 | 29 | 0.5 | 310 | 6.4 |
| 10 | 5 | 111294 | 261444 | 780 | 153 | 127 | 0.9 | 727 | 6.1 |
| 10 | 6 | 352535 | 833015 | 2640 | 221 | 400 | 1.4 | 1595 | 6.6 |
| 15 | 3 | 19228 | 55948 | 102 | 53 | 23 | 0.2 | 363 | 4.4 |
| 15 | 4 | 119800 | 352260 | 790 | 97 | 190 | 0.5 | 1235 | 4.1 |
| 15 | 5 | 592060 | 1754860 | 4670 | 153 | 1020 | 0.9 | 3870 | 4.6 |
| 15 | 6 | 2464168 | 7347928 | 20600 | 221 | 4180 | 1.5 | 11150 | 4.9 |

*Leader election [28].* In this protocol, $N$ nodes that are arranged in an unidirectional ring select an identity randomly according to a uniform distribution on $\{1, \ldots, K\}$. By means of synchronous message passing, processes send their identity around the ring. The protocol terminates once a node has selected a unique id (the node with the highest unique id becomes the leader); if no such node exists, the protocol restarts. The property of interest is the probability to elect a leader within a certain number of rounds: $\mathcal{P}_{\leq q}(\Diamond^{\leq (N+1)\cdot 3}\ \mathit{leader\ elected})$. The obtained results are summarised in the table below. For a fixed $N$, the number of blocks is constant. This is due to the fact that the initial state is the only probabilistic state and that almost all states that are equidistant w. r. t. this initial state are bisimilar. For $N = 4$, no gain in computation time is obtained due to the relatively low number of iterations needed in the original DTMC. When $N$ increases, bisimulation minimisation also pays off timewise; in this case a small reduction of the time is obtained (more iterations are needed due to the bound in the until-formula that depends on $N$).

| original DTMC | | | | | lumped DTMC | | | red. factor | |
|---|---|---|---|---|---|---|---|---|---|
| $N$ | $K$ | states | transitions | ver. time | blocks | lump time | ver. time | states | time |
| 4 | 2 | 55 | 70 | 0.02 | 10 | 0.05 | 0.01 | 5.5 | 0.4 |
| 4 | 4 | 782 | 1037 | 0.4 | 10 | 0.5 | 0.01 | 78.2 | 0.8 |
| 4 | 8 | 12302 | 16397 | 7.0 | 10 | 9.0 | 0.01 | 1230 | 0.8 |
| 4 | 16 | 196622 | 262157 | 165.0 | 10 | 175 | 0.01 | 19662 | 0.9 |
| 5 | 2 | 162 | 193 | 0.1 | 12 | 0.1 | 0.02 | 13.5 | 0.9 |
| 5 | 4 | 5122 | 6145 | 2.8 | 12 | 2.9 | 0.02 | 427 | 0.9 |
| 5 | 6 | 38882 | 46657 | 28 | 12 | 26 | 0.02 | 3240 | 1.1 |
| 5 | 8 | 163842 | 196609 | 140 | 12 | 115 | 0.02 | 13653 | 1.2 |

*Cyclic polling server [27].* This standard example in performance analysis considers a set of stations that are allowed to process a job once they possess the token. The single token circulates among the stations. The times for passing a token to a station and for serving a job are all distributed exponentially. We consider the DTMC that is obtained after uniformisation, and check the formula: $\mathcal{P}_{\bowtie p}(\bigwedge_{j\neq 1}^{N} \neg \mathit{serve}_j \; \mathcal{U} \; \mathit{serve}_1)$, i. e. with probability $\bowtie p$ station 1 will be served before any other station, as well as a time-bounded version thereof.[7] Ordinary (strong) bisimulation yields no state-space reduction. The results for measure-driven bisimulation minimisation are summarised below. In checking the bounded until formula, we used the naive initial partition $\{\{s_0\},\{s_1\},S_?\}$. The improved initial partition with $\{u_0\}$ would have led to almost the same number of blocks as the unbounded until, e. g. 46 instead of 151 blocks for $N = 15$. For both formulas, large reductions in state space size as well as computation time are obtained; the effect of $\{u_0\}$ on the number of blocks is also considerable.

| | | | | | time-bounded until | | | | unbounded until | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| original DTMC | | | | | lumped DTMC | | red. factor | | lumped DTMC | | red. factor | |
| $N$ | states | transitions | time $\mathcal{U}^{\leq t}$ | time $\mathcal{U}$ | blocks | time | states | time | blocks | time | states | time |
| 4 | 96 | 368 | 1.4 | 2.1 | 19 | 0.4 | 5.1 | 3.5 | 12 | 0.9 | 8 | 2.3 |
| 6 | 576 | 2784 | 10 | 11 | 34 | 1.2 | 16.9 | 8.3 | 18 | 1.4 | 32 | 7.9 |
| 8 | 3072 | 17920 | 62 | 52 | 53 | 4.0 | 58 | 15.5 | 24 | 2.9 | 128 | 17.9 |
| 12 | 73728 | 577536 | 3050 | 3460 | 103 | 120 | 716 | 25.4 | 36 | 55 | 2048 | 62.9 |
| 15 | 737280 | 6881280 | 39000 | 32100 | 151 | 1590 | 4883 | 24.5 | 45 | 580 | 16384 | 55.3 |

*Randomised mutual exclusion [37].* In this mutual exclusion algorithm, $N$ processes make random choices based on coin tosses to ensure that they can all enter their critical sections eventually, although not simultaneously. The following table summarizes our results for verifying the property that process 1 is the first to enter the critical section, i. e., the PCTL formula $\mathcal{P}_{\leq q}(\bigwedge_{j\neq 1}^{N} \neg \mathit{enter}_j \; \mathcal{U} \; \mathit{enter}_1)$.

| | | | | strong bisimulation | | | | | $F$-bisimulation | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | original DTMC | | | lumped DTMC | | | red. factor | | lumped DTMC | | red. factor | |
| $N$ | states | transitions | ver. time | blocks | lump time | ver. time | states | time | blocks | time | states | time |
| 3 | 2368 | 8272 | 3.0 | 1123 | 8.0 | 1.6 | 2.1 | 0.3 | 233 | 2.9 | 10.2 | 1.0 |
| 4 | 27600 | 123883 | 47.0 | 5224 | 192 | 19 | 5.3 | 0.4 | 785 | 29 | 35.2 | 1.6 |
| 5 | 308800 | 1680086 | 837 | 18501 | 2880 | 120 | 16.7 | 0.3 | 2159 | 507 | 143 | 1.7 |
| 6 | 3377344 | 21514489 | 9589 | – | $> 10^7$ | – | – | – | 5166 | 7106 | 653 | 1.4 |

Due to the relatively high number of transitions, quotienting the DTMC according to $AP$-bisimilarity is computationally expensive, and takes significantly

[7] For the sake of comparison, the unbounded until-formula is checked on the uniformised and not on the embedded DTMC.

more time than verifying the original DTMC. However, measure-driven bisimilarity yields a quotient that is roughly an order of magnitude smaller than the quotient under $AP$-bisimilarity. Due to the coarser initial partition, this quotient is constructed rather fast. In this case, verifying the original model is more time consuming.

### 4.2 Continuous time

*Workstation cluster [21].* This case study considers a system consisting of two clusters of workstations connected via a backbone. Each cluster consists of $N$ workstations, connected in a star topology with a central switch that provides the interface to the backbone. Each component can break down according to a failure distribution. A single repair unit is available to repair the failed components. The number of correctly functioning workstations determines the level of quality of service (QoS). The following two tables summarise the results for checking the probability that:

- In the long run, premium QoS will be delivered in at least 70% of the cases;
- QoS drops below minimum QoS within 40 time-units is at most 0.1;
- QoS goes from minimum to premium between 20 and 40 time units.

The last property involves a sequence of two transient analyses on different CTMCs. The results for the long-run property:

| | original CTMC | | | lumped CTMC | | | red. factor | |
|---|---|---|---|---|---|---|---|---|
| $N$ | states | transitions | ver. time | blocks | lump time | ver. time | states | time |
| 8 | 2772 | 12832 | 3.6 | 1413 | 12 | 130 | 2 | 0.03 |
| 16 | 10132 | 48160 | 21 | 5117 | 64 | 770 | 2 | 0.03 |
| 32 | 38676 | 186400 | 114 | 19437 | 290 | 215 | 2 | 0.2 |
| 64 | 151060 | 733216 | 730 | 75725 | 1360 | 1670 | 2 | 0.2 |
| 128 | 597012 | 2908192 | 6500 | 298893 | 5900 | 14900 | 2 | 0.2 |
| 256 | 2373652 | 11583520 | 103000 | 1187597 | 25400 | 175000 | 2 | 0.2 |

The plain verification time of the quotient is larger than of the original CTMC, despite a state space reduction of a factor two. This is due to the fact that the subdominant eigenvalues of the Gauss-Seidel iteration matrices differ significantly—the closer this value is to one, the slower the convergence rate for the iterative Gauss-Seidel method. For instance for $N = 8$, the values of the original (0.156) and the quotient (0.993) are far apart and the number of iterations needed differ for about two orders of magnitude. The same applies for $N = 16$. These differences are much smaller for larger values of $N$.

The results for time-bounded reachability:

| | | | | | time-bounded until $[0, 40]$ | | | | time-bounded until $[20, 40]$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| original CTMC | | | | | lumped CTMC | | red. factor | | lumped CTMC | | red. factor | |
| $N$ | states | transitions | ver. time $u^{\leq 40}$ | ver. time $u^{[20,40]}$ | blocks | time | states | time | blocks | time | states | time |
| 8 | 2772 | 12832 | 36 | 49 | 239 | 16.3 | 11.6 | 2.2 | 386 | 24.0 | 7.2 | 2.0 |
| 16 | 10132 | 48160 | 360 | 480 | 917 | 70 | 11.0 | 5.1 | 1300 | 96.0 | 7.8 | 5.0 |
| 32 | 38676 | 186400 | 1860 | 2200 | 3599 | 300 | 10.7 | 6.2 | 4742 | 430 | 8.2 | 5.1 |
| 64 | 151060 | 733216 | 7200 | 8500 | 14267 | 1810 | 10.6 | 4.0 | 18082 | 2550 | 8.4 | 3.3 |
| 128 | 597012 | 2908192 | 29700 | 33700 | 56819 | 9300 | 10.5 | 3.2 | 70586 | 12800 | 8.5 | 2.6 |
| 256 | 2373652 | 11583520 | 121000 | 143000 | 226787 | 45700 | 10.5 | 2.6 | 278890 | 60900 | 8.5 | 2.3 |

These results are obtained using a measure-driven bisimulation. In contrast, for an $AP$-bisimulation, we only obtained a 50% state-space reduction. For measure-driven bisimulation another factor 4–5 reduction is obtained. The reduction factors obtained for this case study are not so high, as its formal (stochastic Petri net) specification already exploits some lumping; e. g., workstations are modeled by anonymous tokens.

*IEEE 802.11 group communication protocol [35].* This is a variant of the centralized medium access protocol of the IEEE 802.11 standard for wireless local area networks. The protocol is centralized in the sense that medium access is controlled by a fixed node, the Access Point (AP). The AP polls the wireless stations, and on receipt of a poll, stations may broadcast a message. Stations acknowledge the receipt of a message such that the AP is able to detect whether or not all stations have correctly received the broadcast message. In case of a detected loss, a retransmission by the originator takes place. It is assumed that the number of consecutive losses of the same message is bounded by $OD$, the omission degree. This all refers to time-critical messages; other messages are sent in another phase of the protocol. The property of interest is, as in [35] and other studies of this protocol, the probability that a message originated by the AP is not received by at least one station within the duration of the time-critical phase, i. e., $t = 2.4$ milliseconds, i. e., $\mathcal{P}_{\bowtie p}(\Diamond^{\leq 24000} \mathit{fail})$ where $\mathit{fail}$ identifies all states in which more than $OD$ losses have taken place. The following table reports the results for the verification of this property for different values of $OD$ and the minimization results for a measure-driven bisimulation.

| | original CTMC | | | lumped CTMC | | red. factor | |
|---|---|---|---|---|---|---|---|
| $OD$ | states | transitions | ver. time | blocks | lump + ver. time | states | time |
| 4 | 1125 | 5369 | 121.9 | 71 | 13.5 | 15.9 | 9.00 |
| 12 | 37349 | 236313 | 7180 | 1821 | 642 | 20.5 | 11.2 |
| 20 | 231525 | 1590329 | 50133 | 10627 | 5431 | 21.8 | 9.2 |
| 28 | 804837 | 5750873 | 195086 | 35961 | 24716 | 22.4 | 7.9 |
| 36 | 2076773 | 15187833 | 5103900 | 91391 | 77694 | 22.7 | 6.6 |
| 40 | 3101445 | 22871849 | 7725041 | 135752 | 127489 | 22.9 | 6.1 |

We obtain a state space reduction of about a factor 22, which results in an efficiency improvement of a factor 5 to 10. The reason that the verification times are rather excessive for this model stems from the fact that the time bound (24000) is very large, resulting in many iterations. These verification times can be improved by incorporating an on-the-fly steady-state detection procedure [30], but this is not further considered here.

*Simple P2P protocol [31].* This case study describes a simple peer-to-peer protocol based on BitTorrent—a "torrent" is a small file which contains metadata about the files to be shared and about the host computer that coordinates the file distribution. The model comprises a set of clients trying to download a file that has been partitioned into $K$ blocks. Initially, there is a single client that has already obtained all blocks and $N$ additional clients with no blocks. Each client can download a block (lasting an exponential delay) from any of the others but they can only attempt four concurrent downloads for each block. The following

table summarises our minimisation results using $AP$-bisimilarity in columns 3 through 6. The property of interest is the probability that all blocks are downloaded within 0.5 time units. The last columns list the results for a recently proposed symmetry reduction technique for probabilistic systems [31] that has been realised in PRISM.

| | | | bisimulation minimisation | | | | | symmetry reduction | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| original CTMC | | | lumped CTMC | | | red. factor | | reduced CTMC | | | red. factor | |
| $N$ | states | ver. time | blocks | lump time | ver. time | states | time | states | red. time | ver. time | states | time |
| 2 | 1024 | 5.6 | 56 | 1.4 | 0.3 | 18.3 | 3.3 | 528 | 12 | 2.9 | 1.93 | 0.38 |
| 3 | 32768 | 410 | 252 | 170 | 1.3 | 130 | 2.4 | 5984 | 100 | 59 | 5.48 | 2.58 |
| 4 | 1048576 | 22000 | 792 | 10200 | 4.8 | 1324 | 2.2 | 52360 | 360 | 820 | 20.0 | 18.3 |

We observe that bisimulation minimisation leads to a significantly stronger state-space reduction than symmetry reduction. For $N = 3$ and $N = 4$, bisimulation minimisation leads to a state-space reduction of more than 23 and 66 times, respectively, the reduction of symmetry reduction. Symmetry reduction is—as expected—much faster than bisimulation minimisation, but this is a somewhat unfair comparison as the symmetries are indicated manually. These results suggest that it is affordable to first apply a (fast) symmetry reduction, followed by a bisimulation quotienting on the obtained reduced system. Unfortunately, the available tools did not allow us to test this idea.

### 4.3 Rewards

This section reports on the results for bisimulation minimisation for Markov reward models. Note that the initial partitions need to be adapted such that only states with equal reward are grouped. We have equipped two DTMCs and one CTMC with a reward assignment function $r$:

- Crowds protocol (DMRM): the reward indicates the number of messages sent;
- Randomised mutual exclusion protocol (DMRM): the reward indicates the number of attempts that have been undertaken to acquire access to the critical section;
- Workstation cluster (CMRM): the reward is used to measure the repair time.

Recall that for DMRMs, $r(s)$ indicates the reward that is earned on leaving a state, while for CMRMs, $r(s) \cdot t$ is the earned reward when staying $t$ time-units in $s$. The experiments are focused on verifying time- and reward-bounded until-formulas. For DMRMs, these formulas are checked using a path graph generation algorithm as proposed in [1] which has a time complexity in $O(k \cdot r \cdot |S|^3)$, where $k$ and $r$ are the time-bound and reward-bound, respectively. For CMRMs, we employed the discretization approach by Tijms and Veldman as proposed in [20] which runs in time $O(t \cdot r \cdot |S|^3 \cdot d^{-2})$ where $d$ is the step size of the discretisation. In our experiments, the default setting is $d = \frac{1}{32}$.

For the Crowds protocol (for $R = 3$), we checked the probability that the sender's id is discovered within 100 steps and maximally two messages, i. e.,

$\mathcal{P}_{\leq p}(\Diamond^{\leq 100}_{\leq 2}\, observe)$. In case of the randomised mutual exclusion protocol, we checked $\mathcal{P}_{\leq q}(\bigwedge_{j\neq 1}^{N} \neg enter_j\ \mathcal{U}^{\leq 50}_{\leq 10}\ enter_1)$, i. e., maximally 10 attempts are allowed to enter the critical section. Finally, for the workstation cluster, we checked the change of providing minimum QoS to premium QoS within maximally 5 time units of repair (and 10 time units). All results are listed in the following table.

Due to the prohibitive (practical) time-complexity, manageable state space sizes are (much) smaller than for the case without rewards. Another consequence of these large verification times, bisimulation minimisation is relatively cheap, and results in possibly drastic time savings, as for the Crowds protocol.

| *Crowds protocol with rewards* | | | | | | | |
|---|---|---|---|---|---|---|---|
| | original DTMC | | | lumped DTMC | | red. factor | |
| $N$ | states | transitions | ver. time | blocks | lump + ver. time | states | time |
| 5 | 1198 | 2038 | 2928 | 93 | 44.6 | 12.88 | 65.67 |
| 10 | 6563 | 15143 | 80394 | 103 | 73.5 | 63.72 | 1094.49 |
| 15 | 19228 | 55948 | 1004981 | 103 | 98.7 | 186.68 | 10182.13 |
| 20 | 42318 | 148578 | 5174951 | 103 | 161 | 410.85 | 32002.61 |
| *Randomised mutual exclusion protocol with rewards* | | | | | | | |
| 2 | 188 | 455 | 735 | 151 | 616 | 1.25 | 1.19 |
| 3 | 2368 | 8272 | 60389 | 1123 | 19010 | 2.11 | 3.18 |
| 4 | 27600 | 123883 | 5446685 | 5224 | 298038 | 5.28 | 18.28 |
| 5 | 308800 | 1680086 | $> 10^7$ | 18501 | 3664530 | 16.69 | – |
| *Workstation cluster with rewards* | | | | | | | |
| 2 | 276 | 1120 | 278708 | 147 | 55448 | 1.88 | 5.03 |
| 3 | 512 | 2192 | 849864 | 268 | 151211 | 1.91 | 5.62 |
| 4 | 820 | 3616 | 2110095 | 425 | 347324 | 1.93 | 6.08 |
| 5 | 1200 | 5392 | $> 10^7$ | 618 | 2086575 | 1.94 | – |
| 6 | 1652 | 7520 | $> 10^7$ | 847 | 3657682 | 1.95 | – |

## 5 Concluding remarks

Our experiments confirm that significant (up to logarithmic) state space reductions can be obtained using bisimulation minimisation. The appealing feature of this abstraction technique is that it is fully automated. For several case studies, also substantial reductions in time have been obtained (up to a factor 25). This contrasts results for traditional model checking where bisimulation minimisation typically outweighs verifying the original system. Time reduction strongly depends on the number of transitions in the Markov chain, its structure, as well as on the convergence rate of numerical computations. The P2P protocol experiment shows encouraging results compared with symmetry reduction [31] (where symmetries are detected manually). For measure-driven bisimulation for models without rewards, this speedup comes with no memory penalty: the peak memory use is typically unchanged; for ordinary bisimulation some experiments showed an increase of peak memory up to 50 %. In our case studies with rewards, we experienced a 20–40 % reduction in peak memory use.

We plan to further investigate combinations of symmetry reduction with bisimulation minimisation, and to extend our experimental work towards MDPs and simulation preorders.

*Acknowledgement.* This research has been performed as part of the MC=MC project that is financed by the Netherlands Organization for Scientific Research (NWO), and the project VOSS2 that is financed by NWO and the German Research Council (DFG).

## References

1. Suzana Andova, H. Hermanns, and Joost-Pieter Katoen. Discrete-Time Rewards Model-Checked. In K.G. Larsen and P. Niebert, editors, *Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 2791, pages 88–104. LNCS, Springer, 2003.
2. Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.
3. Adnan Aziz, Kumud Sanwal, Vigyan Singhal, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli. It Usually Works: The Temporal Logic of Stochastic Systems. In P. Wolper, editor, *Computer Aided Verification (CAV)*, volume 939 of *LNCS*, pages 155–165. Springer, 1995.
4. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-Checking Algorithms for Continuous-Time Markov Chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.
5. Christel Baier, Frank Ciesinski, and Marcus Größer. ProbMela and verification of Markov decision processes. *ACM SIGMETRICS Performance Evaluation Review*, 32(4):22–27, 2005.
6. Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. On the Logical Characterisation of Performability Properties. In Ugo Montanari, Jos D. P. Rolim, and Emo Welzl, editors, *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1853 of *LNCS*, pages 780–792. Springer, 2000.
7. Christel Baier, Joost-Pieter Katoen, Holger Hermanns, and Verena Wolf. Comparative branching-time semantics for Markov chains. *Information and Computation*, 200(2):149–214, 2005.
8. Eckard Bode, Marc Herbstritt, Holger Hermanns, Sven Johr, Thomas Peikenkamp, Reza Pulungan, Ralf Wimmer, and Bernd Becker. Compositional Performability Evaluation for STATEMATE. In *Quantitative Evaluation of Systems (QEST)*, pages 167–178. IEEE Computer Society, 2006.
9. P. Buchholz. Exact and ordinary lumpability in finite Markov chains. *Journal of Applied Probability*, 31:59–75, 1994.
10. Davide D'Aprile, Susanna Donatelli, and Jeremy Sproston. CSL Model Checking for the GreatSPN Tool. In Cevdet Aykanat, Tugrul Dayar, and Ibrahim Korpeoglu, editors, *Computer and Information Sciences*, volume 3280 of *LNCS*, pages 543–553. Springer, 2004.
11. Pedro R. D'Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen, and Kim Guldstrand Larsen. Reachability Analysis of Probabilistic Systems by Successive Refinements. In Luca de Alfaro and Stephen Gilmore, editors, *Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM/PROBMIV)*, volume 2165 of *LNCS*, pages 39–56. Springer, 2001.
12. S. Derisavi. *Solution of Large Markov Models Using Lumping Techniques and Symbolic Data Structures.* PhD thesis, University of Illinois at Urbana-Champaign, 2005.

13. Salem Derisavi, Holger Hermanns, and William H. Sanders. Optimal State-Space Lumping in Markov Chains. *Information Processing Letters*, 87(6):309–315, 2003.
14. Harald Fecher, Martin Leucker, and Verena Wolf. *Don't Know* in Probabilistic Systems. In Antti Valmari, editor, *Model Checking of Software (SPIN)*, volume 3925 of *LNCS*, pages 71–88. Springer, 2006.
15. K. Fisler and M. Y. Vardi. Bisimulation Minimization in an Automata-Theoretic Verification Framework. In Ganesh Gopalakrishnan and Phillip J. Windley, editors, *Formal Methods in Computer-Aided Design (FMCAD)*, volume 1522 of *LNCS*, pages 115–132. Springer, 1998.
16. K. Fisler and M. Y. Vardi. Bisimulation and Model Checking. In Laurence Pierre and Thomas Kropf, editors, *Correct Hardware Design and Verification Methods (CHARME)*, volume 1703 of *LNCS*, pages 338–342. Springer, 1999.
17. K. Fisler and M. Y. Vardi. Bisimulation Minimization and Symbolic Model Checking. In *Formal Methods in System Design*, volume 21, pages 39–78. Kluwer Academic Publishers, 2002.
18. Marcus Größer and Christel Baier. Partial Order Reduction for Markov Decision Processes: A Survey. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever, editors, *Formal Methods for Components and Objects (FMCO)*, volume 4111 of *LNCS*, pages 408–427. Springer, 2005.
19. N. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
20. B. Haverkort, L. Cloth, H. Hermanns, J.-P. Katoen, and C. Baier. Model Checking Performability Properties. In *Dependable Systems and Networks (DSN)*, pages 103–112. IEEE Computer Society, 2002.
21. B. Haverkort, H. Hermanns, and J.-P. Katoen. On the Use of Model Checking Techniques for Dependability Evaluation. In *Symposium on Reliable Distributed Systems (SRDS)*, pages 228–237. IEEE Computer Society, 2000.
22. Holger Hermanns, Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Markus Siegle. On the use of MTBDDs for performability analysis and verification of stochastic systems. *Journal of Logic and Algebraic Programming*, 56(1-2):23–67, 2003.
23. Jane Hillston. *A Compositional Approach to Performance Modelling*. Distinguished Dissertations Series. Cambridge University Press, New York, NY, USA, 1996.
24. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In H. Hermanns and J. Palsberg, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.
25. Michael Huth. An Abstraction Framework for Mixed Non-deterministic and Probabilistic Systems. In Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, Joost-Pieter Katoen, and Markus Siegle, editors, *Validation of Stochastic Systems*, volume 2925 of *LNCS*, pages 419–444. Springer, 2004.
26. Michael Huth. On finite-state approximants for probabilistic computation tree logic. *Theoretical Computer Science*, 346(1):113–134, 2005.
27. Oliver C. Ibe and Kishor S. Trivedi. Stochastic Petri Net Models of Polling Systems. *Selected Areas in Communications*, 8(9):1649–1657, 1990.
28. Alon Itai and Michael Rodeh. Symmetry breaking in distributed networks. *Information and Computation*, 88(1):60–87, 1990.
29. Joost-Pieter Katoen, Maneesh Khattri, and Ivan S. Zapreev. A Markov Reward Model Checker. In *Quantitative Evaluation of Systems (QEST)*, pages 243–244. IEEE Computer Society, 2005.

30. Joost-Pieter Katoen and Ivan S. Zapreev. Safe On-The-Fly Steady-State Detection for Time-Bounded Reachability. In *Quantitative Evaluation of Systems (QEST)*, pages 301–310. IEEE Computer Society, 2006.
31. M. Kwiatkowska, G. Norman, and D. Parker. Symmetry Reduction for Probabilistic Model Checking. In T. Ball and R. Jones, editors, *Computer Aided Verification (CAV)*, volume 4114 of *LNCS*, pages 234–248. Springer, 2006.
32. Marta Kwiatkowska, Gethin Norman, and David Parker. Game-based Abstraction for Markov Decision Processes. In *Quantitative Evaluation of Systems (QEST)*, pages 157–166. IEEE Computer Society, 2006.
33. Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.
34. Mouad Ben Mamoun, Nihal Pekergin, and Sana Younes. Model Checking of Continuous-Time Markov Chains by Closed-Form Bounding Distributions. In *Quantitative Evaluation of Systems (QEST)*, pages 189–198. IEEE Computer Society, 2006.
35. Mieke Massink, Joost-Pieter Katoen, and Diego Latella. Model Checking Dependability Attributes of Wireless Group Communication. In *Dependable Systems and Networks (DSN)*, pages 711–720. IEEE Computer Society, 2004.
36. David Monniaux. Abstract interpretation of programs as Markov decision processes. *Science of Computer Programming*, 58(1-2):179–205, 2005.
37. A. Pnueli and L. Zuck. Verification of Multiprocess Probabilistic Protocols. *Distributed Computing*, 1(1):53–72, 1986.
38. M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. In *ACM Transactions on Information and System Security*, volume 1, pages 66–92. ACM Press, 1998.
39. Jeremy Sproston and Susanna Donatelli. Backward Bisimulation in Markov Chain Model Checking. *IEEE Transactions on Software Engineering*, 32(8):531–546, 2006.