Math. Struct. in Comp. Science (2013), *vol.* 23, *pp.* 751–795. © Cambridge University Press 2013
doi:10.1017/S0960129512000254

Model checking for performability

C. BAIER[†], E. M. HAHN[‡], B. R. HAVERKORT[§], H. HERMANNS[‡] and J.-P. KATOEN[¶]

[†] *Technische Universität Dresden, Dresden, Germany*
Email: baier@tcs.inf.tu-dresden.de
[‡] *Saarland University, Saarbrücken, Germany*
Email: {hermanns;emh}@cs.uni-saarland.de
[§] *Embedded Systems Institute, Eindhoven, The Netherlands*
and
University of Twente, Enschede, The Netherlands
Email: brh@cs.utwente.nl
[¶] *University of Twente, Enschede, The Netherlands*
and
RWTH Aachen University, Aachen, Germany
Email: katoen@cs.rwth-aachen.de

Received 18 July 2011; revised 14 October 2011

This paper gives a bird's-eye view of the various ingredients that make up a modern, model-checking-based approach to performability evaluation: Markov reward models, temporal logics and continuous stochastic logic, model-checking algorithms, bisimulation and the handling of non-determinism. A short historical account as well as a large case study complete this picture. In this way, we show convincingly that the smart combination of performability evaluation with stochastic model-checking techniques, developed over the last decade, provides a powerful and unified method of performability evaluation, thereby combining the advantages of earlier approaches.

1. Introduction

Since the mid-1970's the notion of *performability* has been developed and applied (see Section 2) successfully in the joint assessment of the performance and reliability of computer and communication systems. A crucial element in doing this has been the construction and analysis of so-called Markov reward models (see Section 3).

In this paper we demonstrate that modern-day temporal logics, like CSL and CSRL, allow a very flexible way of presenting all kinds of performability measures of interest, as shown in Section 4. Combined with stochastic model-checking techniques (see Section 5), they form an excellent vehicle for actually carrying out performability evaluation studies, as demonstrated in Section 8. Moreover, recent results for achieving state-space reductions using the notion of bisimulation, as developed in the realm of formal verification, can be applied here also, as we show in Section 6. Furthermore, the issue of non-determinism, which has traditionally not been addressed at all in the field of stochastic models for performability evaluation, also turns out to be a valuable technique when used in combination with stochastic models, leading to upper

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

and lower bounds for performability measures when parts of the model are underspecified – see Section 7. We present a case study in Section 8 addressing a survivability assessment of an abstract model of the Google file system demonstrating the power of our approach. A concise overview of the symbols used throughout this paper is given in an appendix.

To summarise, this paper provides background details on the various ingredients that make up a modern, model-checking-based approach to performability evaluation. It shows that the combination of performability evaluation with stochastic model checking provides us with a powerful and unified evaluation tool exploiting the advantages of earlier approaches.

2. A short history of performability

2.1. *Before it all started (up to 1975)*

Up until the mid 1970's, the notion of performability as we know it now did not exist. Although the term 'performability' can be found in pre-1975 publications (for example, using Google Scholar), it was just used then to mean '*perform-ability*': for example, in the context of biological and chemical reactions, the ability to make a step or to take place.

Before the mid-1970's, the desirable properties of performance (throughput, delays) and reliability (mean-time to failure, availability, reliability) for computer systems were typically treated independently, effectively assuming that the system could not fail when assessing performance, or, when addressing reliability or availability, that the performance, as such, did not matter, and that only the sheer operation was of interest.

In the 1960's, the field of performance evaluation in itself started to develop, but focussed on relatively simple queueing models and queueing networks (*cf.* the well-known books Kleinrock (1975) and Kleinrock (1976)), with applications in the (classical) telecommunications sector and multiprogrammed computer systems (mainframes).

The field of reliability engineering was already flourishing, largely in the context of mechanical and electrical engineering, thereby focussed on small, and often simplified, models of system components, and using simple reliability block diagrams and fault-trees. Dealing with complex component dependencies, repair units and larger system models was not done in practice.

The use of simple Markovian models (continuous-time Markov chains), both in the context of reliability and performance evaluation, started to grow, though practical application was hampered by the limited numerical means (algorithms and readily available computers) of that time. An example of this is provided by the work in Arnold (1973) on a number of simple Markovian models, which showed the importance of the notion of system coverage, that is, the probability of proper reconfiguration after a system fault has taken place. His model, and most other Markov chain related studies, focussed on relatively simple and structured models, for which closed-form solutions for the measures of interest could be obtained.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

2.2. *Performability: birth and childhood (1975–1985)*

Throughout the 1970's, the idea started to develop that for many applications (or systems), a separate treatment of performance and reliability was not always useful. Example cases that emerged were large telecommunication switches in which individual modules could fail, thereby affecting the delivered system performance, but not directly causing the system as a whole to fail. Other fields where such *fault-tolerant systems* started to emerge include the aerospace and aircraft industries, and in large-scale computing facilities in which multiple fairly autonomous systems were interconnected through new networking technology, such as Ethernet or IBM token ring. It is interesting to note that the first annual IEEE Conference on Fault-Tolerant Computing Systems (FTCS) took place in 1971: it still runs anually, though following a merger with the IFIP conference series on Dependable Computing for Critical Applications (DCCA) and the IEEE Performance and Dependability Symposium (PDS), since 2000 it has been held under the name Dependable Systems and Networks (DSN)[†].

The first contributions on performability, that is, the integrated analysis of system performance and reliability, were published in the IEEE FTCS series in the 1970's. Notable amongst this work is Beaudry (1978) which developed the notion of 'the mean *computation* before failure' in a Markovian model as a generalisation of the classical mean *time* to failure, thereby taking into account the different computational capacity in various degraded systems states, giving them a different value (or reward), before reaching an overall system failure state. Another important paper was Gay and Ketelsen (1979), which studied the performance of gracefully degradable systems.

The true seminal papers on performability are due to John F. Meyer, now Professor Emeritus of the University of Michigan. In a series of papers (Meyer 1976; Meyer 1980; Meyer 1982), he carefully defined the notion of performability as a precise probabilistic measure of levels of performance that can be attained in systems that change their structure over time, for example, due to the occurrence of faults and repair actions. Although the framework and definitions he introduced are very general, they are most often 'implemented' in a Markovian context, and this is often taken to be *the* performability framework (as we will do in the rest of the paper). Meyer and his students also focussed on the Markovian case (see, for example, Movaghar and Meyer (1984), Sanders and Meyer (1987) and Sanders and Meyer (1991)), and also developed a number of algorithms to evaluate simpler models, for example, models with an acyclic structure (Furchtgott and Meyer 1984).

In the early 1980's, a number of developments took place that influenced the growth of the performability field:

— Small computer systems (VAX, PDP 11 and later PCs) became available that brought the numerical evaluation of Markovian models within the reach of many researchers. Towards 1985, software tools also started to emerge, allowing Markovian models with several thousands of states to be constructed and analysed efficiently.

[†] See http://www.dsn.org/.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

— Clearly, such large Markovian models could not be input manually, but were the result of 'high-level models' with an underlying Markov chain semantics. Instrumental in this has been the work reported in Molloy (1982) on stochastic Petri nets (extending Petri nets with stochastic timing), and later Ajmone Marsan *et al.* (1984) in developing generalised stochastic Petri nets (GSPNs) and Movaghar and Meyer (1984) on stochastic activity networks (SANs). The Petri net approaches were initially developed purely with a view to the application of performance evaluation, while SANs were developed with the performability application in mind from the outset.
— In 1983, Gross and Miller introduced an efficient method for analysing the time-dependent behaviour of continuous-time Markov chains (CTMCs) (Gross and Miller 1984). This method, now known as uniformisation, played an enormously important role in the development of efficient performability algorithms in the years to come. However, it took some years (at least 5) for it to be taken up. In fact, it turned out that the method was a specialised case of a method that had been proposed earlier in Jensen (1953).

2.3. *Performability: adolescence (1985–1995)*

A characteristic of the adolescent phase is an enormous sudden growth. The (re)invention of uniformisation in the early 1980's resulted in a huge growth in the application potential for performability evaluation since it allowed the evaluation of the transient state-probabilities in larger Markov chains in an efficient and stable way, without the need to solve differential equations numerically. This new technique, combined with the quickly growing computational and memory capabilities of commodity computers, made performability evaluation applicable in practice. The number of publications in the field on new uniformisation-based techniques, supporting tools and applications grew sharply. More attention was paid to high-level tool mechanisms, thus also making the specification of large models much easier. During this period, there was also a growth in research capacity around the theme of 'performability', and new software tools were developed, along with the publication of new algorithms for both general and special cases.

In 1991, the first performability workshop took place at the University of Twente[†]. This brought together some forty researchers in the field (the number being limited in part by travel bans due to the outbreak of the Gulf war a few days before the workshop). The combined evaluation of performance and reliability (or dependability, as it was now often called more generically – see Avizienis *et al.* (2004)) had become more mainstream.

To conclude this section, the work of many others could be mentioned, for example:

— for tools, see Donatiello and Iyer (1987a), Donatiello and Iyer (1987b), Geist and Trivedi (1990), Goyal *et al.* (1987) and Haverkort and Niemegeers (1996);
— for specialised recursive algorithms to evaluate performability distributions for specially structured, typically acyclic, Markovian models, see Goyal and Tantawi (1987), Goyal and Tantawi (1988) and Grassi *et al.* (1988);

[†] See http://www.pmccs.net/ and the special issue van Dijk *et al.* (1992).

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

— for some more general numerical techniques, see Pattipati *et al.* (1993), Qureshi and Sanders (1994a), Reibman and Trivedi (1988), Reibman and Trivedi (1989), Smith *et al.* (1988) and Souza e Silva and Gail (1992).

More on all these can be found in several surveys, such as Meyer (1992), Meyer (1995) and Haverkort *et al.* (2001).

2.4. *Performability: young adult (1995–2005)*

Throughout the 1990's and the early 21st century, performability evaluation grew more mature in terms of the refinement of algorithms and the incorporation of the algorithms in more tools. Markov reward models (Howard 1971a; Howard 1971b), that is, the combination of CTMCs with state and impulse reward structures remained the most prominent model. Model enrichment towards semi-Markovian models (Markov models with more generally distributed state residence times) were addressed, but were not taken up widely.

The use of symbolic state space representation methods, such as multi-terminal decision diagrams (Hermanns *et al.* 2003) and matrix diagrams (Ciardo *et al.* 2007), adapted from the field of formal verification and the use of binary decision diagrams, made even larger state spaces possible (Markov chains with hundreds of millions of states). The bottleneck in practical use of such large models now lay in the representation of the non-sparse real probability vectors.

A second important development was the connection made to the field of model checking (Baier and Katoen 2008) through the development of temporal logics, now enhanced with (stochastic) information on timing and reward, and leading to model-checking procedures for CSL (continuous stochastic logic) (Aziz *et al.* 2000; Baier *et al.* 2003) and CSRL (continuous stochastic reward logic) (Baier *et al.* 2000; Baier *et al.* 2010b). This connection also led to the incorporation of performability-like evaluation techniques into model-checking tools like MRMC (Katoen *et al.* 2011), PRISM (Kwiatkowska *et al.* 2009), and (extended) stochastic Petri net tools like GreatSPN (Chiola 1985) and SMART (Ciardo *et al.* 2003).

2.5. *Performability: maturity and offspring (2005 onwards)*

Some 30 years after its invention, the notion of performability evaluation, specialised to Markov reward models, is now fully developed. Although refinements in techniques and tools still take place, no new breakthroughs in the performability field *per se* can be reported (nor are they expected in the future). However, the combination of performability with state-of-the-art model-checking techniques, as presented in this paper, does form a major technical development, as will become clear in the rest of this paper.

As with most well-developed techniques and technology, the novelty often lies in new application fields. The notion of survivability, that is, the ability of a system to recover to agreed-upon levels of quality of service after the occurrence of disasters of some form, can be cast in the performability framework. The case study in this paper, which is based on the earlier work Cloth and Haverkort (2005), is an example of this. Also, some more

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

recent approaches to assessing system security in a quantitative way have built on the notion of performability evaluation (Haverkort 2006; Sanders 2010).

Another application lies in the field of energy consumption for electronic devices, especially when these are battery-powered. Instead of looking at reward as something one gains from a system, reward can also be seen as a system's cost, for example, in terms of energy usage. System performance models can then be combined with battery models, leading to integrated analyses (and trade-offs) between system performance and system lifetime. However, this leads to the use of Markov reward models that are time-inhomogeneous with respect to their transition rates and rewards, thus making numerical analysis much more difficult and challenging in practice, as shown in Cloth *et al.* (2007).

Finally, the general notion of performability evaluation as originally defined by Meyer can also be used in combination with other underlying mathematical structures, that is, models other than Markov reward models. For instance, the base model could be a timed automaton, with or without probabilities or stochastic timing, and with or without non-determinism. The underlying mathematical structure is then no longer a plain Markov reward model, but a discrete-continuous hybrid system describing systems of coupled non-linear partial differential equations, which leads to new challenges for numerical solution – see, for instance, Abate *et al.* (2008), Abate *et al.* (2011), Berendsen *et al.* (2006) and Larsen and Rasmussen (2008).

3. Markov reward models

This section introduces some basics concerning Markov reward models (MRMs). We consider continuous-time models, which means that the core behaviour of an MRM is given as a continuous-time Markov chain (CTMC). Basically, a CTMC is a finite-state automaton where transitions are labelled by (the rates of) negative exponential distributions. Recall that a non-negative continuous random variable X is exponentially distributed with rate $\lambda \in \mathbb{R}_{>0}$ if the probability of X being at most t (where t is a time parameter) is given by

$$F_X(t) \;=\; \Pr(X \leqslant t) = 1 - e^{-\lambda \cdot t}$$

for $t \geqslant 0$, and has mean $1/\lambda$.

Definition 3.1 (CTMC). A labelled continuous-time Markov chain (CTMC) $\mathscr{C}$ is a tuple $(S, \mathbf{R}, \mathbf{L})$ where:

— S is a finite set of *states*,
— $\mathbf{R} : S \times S \to \mathbb{R}_{>0}$ is the *rate matrix*; and
— $L : S \to 2^{AP}$ is the *labelling* function, which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions $a \in AP$ that are valid in s.

A state s is said to be *absorbing* if and only if $\mathbf{R}(s, s') = 0$ for all states s'.

The state residence time of a state s in a CTMC is determined by its outgoing rates $\mathbf{R}(s, \cdot)$. More precisely, the residence time of s is exponentially distributed with rate

$$E(s) = \sum_{s' \in S} \mathbf{R}(s, s').$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Accordingly, the probability to exit state s within t time units is given by

$$\int_0^t E(s) \cdot e^{-E(s)\tau} d\tau.$$

On leaving state s, the probability that the next state is s' equals

$$\frac{\mathbf{R}(s, s')}{E(s)},$$

or $\mathbf{P}(s, s')$ for short. Combining these two ingredients, we get that the probability of making the transition $s \to s'$ within t time units is

$$\mathbf{P}(s, s') \cdot \int_0^t E(s) e^{-E(s)\cdot\tau} d\tau.$$

As a running example, we will model the failure behaviour of the Hubble space telescope (HST), which is a well-known orbiting astronomical observatory. In particular, we focus on the steering unit, which contains six gyroscopes. They report any small movements of the spacecraft to the HST pointing and control system. The computers then command the spinning reaction wheels to keep the spacecraft stable or moving at the desired rate in order to avoid the telescope pointing device from staggering. This is of particular importance in preventing pictures taken by the telescope from being blurred. The system works by comparing the HST motion relative to the axes of the spinning masses inside the gyroscopes. Due to the possibility of failure, the gyroscopes are arranged in such a way that any group of three gyroscopes can keep the telescope operating with full accuracy. With fewer than three gyroscopes, the telescope turns into sleep mode and a Space Shuttle mission must be undertaken to repair it. Without operational gyroscopes, the telescope runs the risk of crashing. Hubble is the only telescope designed to be serviced in space by astronauts. Four servicing missions were performed between 1993 to 2002. Servicing Mission 3A (1999) was initiated after three of the six on-board gyroscopes had failed (a fourth failed a few weeks before the mission, rendering the telescope incapable of performing observations).

Example 3.2. We model the HST and the failing behaviour of its gyroscopes as a CTMC. We make the following (not necessarily realistic) assumptions about the timing behaviour of the telescope: each gyroscope has an average lifetime of 10 years, the average preparation time of a repair mission is two months, and to turn the telescope into sleep mode takes 1/100 years (about 3.5 days) on average. Assuming a base time scale of a single year, the real-time probabilistic behaviour of the failure and repair of the gyroscopes is now modelled by the CTMC of Figure 1. This model can be understood as follows. The mean residence time of a state is the reciprocal of the sum of its outgoing transition rates. In state 6, for instance, one out of 6 gyroscopes may fail. As these failures are stochastically independent and as each gyroscope fails with rate 1/10, this state has outgoing rate 6/10. If fewer operational gyroscopes are available, these rates decrease proportionally, and state residence times become larger. In state 2, there are two possibilities: either one of the remaining two gyroscopes fails (with probability 1000/1002), or the telescope is turned into sleep mode (with probability 2/1002). The mean residence time of state 2 is 10/1002.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

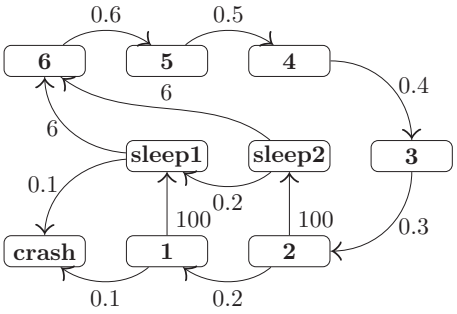


Fig. 1. A simplified CTMC model of the Hubble space telescope.

In practice, we can assume that during a repair, all failed gyroscopes will be replaced, and the system restarts as new; since the dominant delay is the mission preparation time, the rates of going from any of the sleep states to the initial state are equal.

An MRM is now given as a CTMC enriched with the notion of a cost, or, dually, a gain. This is done in two ways. The costs associated with transitions, which are called *impulse rewards*, are constant non-negative real values that are incurred on performing a transition. Thus, on making a transition between state s and s' with impulse reward $\iota(s, s')$, a reward of $\iota(s, s')$ is earned. Similarly, a cost rate is associated with states, and is called a *state reward*. The intuition is that residing t time units in a state with cost rate $\rho(s)$ leads to a reward of $\rho(s) \cdot t$ being earned.

Definition 3.3 (Markov reward model). A labelled Markov reward model (MRM) $\mathscr{M}$ is a triple $(\mathscr{C}, \iota, \rho)$ where $\mathscr{C}$ is a labelled CTMC, $\iota : S \times S \to \mathbb{R}_{\geqslant 0}$ assigns *impulse rewards* to pairs of states and $\rho : S \to \mathbb{R}_{\geqslant 0}$ assigns a *reward rate* to each state $s \in S$.

At this point we should explain why MRMs are considered as the central model in this paper, and indeed in performability analysis in general. The incorporation of stochastic timing is motivated by the fact that failures and repairs, which are the key events of interest in performability, do indeed occur in a stochastic manner since their occurrence is random. Given that, the negative exponential distribution is a specific, though rather reasonable, choice, especially for failures. It is well known that the exponential distribution maximises the entropy (if only the mean failure rate is known, the most appropriate stochastic approximation is by means of an exponential distribution with that mean). For repairs, one might also consider other distributions, such as a uniform distribution or combinations of uniform and deterministic distributions. However, these distributions can be matched arbitrarily closely by phase-type distributions (at the cost of state-space increase), which are defined as the time until absorption in a CTMC. Finally, MRMs turn out to provide a good balance between expressivity on the one hand and the analysis possibilities on the other – with other stochastic assumptions, the analysis would become much more involved.

Example 3.4. One of the main tasks of the HST is to take bright pictures of astronomical targets. A key instrument is the High Speed Photometer (HSP), which was designed to

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

measure the brightness and polarity of rapidly varying celestial objects. It can observe in the ultraviolet, visible and near infra-red regions of the spectrum at a rate of one measurement per 10 μsec (which amounts to about $32{\cdot}10^{10}$ measurements per year). Observations are still possible with fewer gyroscopes (up to two), but the area that can be viewed is then more restricted, and observations requiring very accurate pointing are more difficult. We can model this effect using state rewards by assuming that with 6 gyroscopes, a 100% coverage is possible, which gradually decays to 20% for two gyroscopes. Accordingly:

$$\begin{aligned}\rho(6) &= 32{\cdot}10^{10}\\ \rho(5) &= \frac{4}{5}{\cdot}\rho(6)\\ \rho(4) &= \frac{3}{5}{\cdot}\rho(6)\\ \rho(3) &= \frac{2}{5}{\cdot}\rho(6)\\ \rho(2) &= \frac{1}{5}{\cdot}\rho(6).\end{aligned}$$

All other states have state reward zero: in particular, we assume that no observations are possible in sleep mode.

Alternatively, one can consider the angular measurement of one of the other key instruments of the HST, the Wide Field Camera (WFC3). It is the HST's most recent and technologically advanced instrument (installed in 2009) for taking images in the visible spectrum. Its optical channel covers 164 by 164 arcsec (arcsec is the unit of angular measurement and amounts to 1/60 of one degree), which is about 8.5% of the diameter of the full moon as seen from the earth. In a similar way to the above, we assume that with 6 gyroscopes, 100% coverage is possible, and this gradually decays to 20% for two gyroscopes. Using $\widehat{\rho}(6) = 164^2$, we define $\widehat{\rho}(\cdot)$ as above. However, as the WFC3 can still make measurements when the gyroscopes are in sleep mode, we have

$$\widehat{\rho}(sleep2) = \widehat{\rho}(2) = 0.2{\cdot}\widehat{\rho}(6),$$

and let

$$\widehat{\rho}(sleep1) = \widehat{\rho}(1) = 0.$$

Finally,

$$\widehat{\rho}(crash) = 0.$$

As an example of impulse rewards, we consider the switching and repair costs between the different operational modes of the HST. For instance, switching from an operational mode to a sleep mode requires physical changes of the HST, and repair costs are typically huge since a space mission will need to be prepared and undertaken. A hypothetical

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

reward structure is

$$\iota(2, sleep2) = \iota(1, sleep1) = 1$$
$$\iota(sleep1, 6) = \iota(sleep2, 6) = 10^8$$
$$\iota(sleep1, crash) = \iota(1, crash) = 10^{12}.$$

Here, we assume for simplicity that a crash is 10,000 times more expensive than a repair. All other transitions have impulse reward zero.

Runs of MRMs are formalised as *maximal paths*, that is, paths that are not extensible: an infinite path σ through an MRM is a sequence $s_0, t_0, s_1, t_1, s_2, t_2, \ldots$ where for $i \in \mathbb{N}$, we have $s_i \in S$, $t_i \in \mathbb{R}_{>0}$ and $\mathbf{R}(s_i, s_{i+1}) > 0$. A finite path is a sequence $s_0, t_0, s_1, t_1, s_2, t_2, \ldots, s_n, t_n$ where for $0 \leqslant i \leqslant n-1$, the requirements are the same as for an infinite path, but in addition we also have $E(s_n) = 0$ and $t_n = \infty$. Intuitively, states s_i are visited along the path σ, and the residence time in state s_i equals t_i. For $i \in \mathbb{N}$, we let $\sigma[i] = s_i$ be the $(i+1)$th state of σ if it is defined. For $t \in \mathbb{R}_{\geqslant 0}$ and i the smallest index with $t < \sum_{j=0}^{i} t_j$, let $\sigma@t = \sigma[i]$, which is the state occupied in path σ at time t. The set of all maximal paths starting in state s, that is, $s_0 = s$, is denoted by $Path(s)$. The next definition formalises the reward that is accumulated along a path σ up to some time instant t.

Definition 3.5 (cumulative reward). Let $\sigma = s_0, t_0, s_1, t_1, s_2, t_2, \ldots$ be a path of the MRM $\mathscr{M}$ and $t = \sum_{j=0}^{k-1} t_j + t'$ with $t' < t_k$. The *cumulative* reward of σ up to time t is defined by

$$y(\sigma, t) \;=\; \sum_{j=0}^{k-1} t_j \cdot \rho(s_j) + t' \cdot \rho(s_k) + \sum_{j=0}^{k-1} \iota(s_j, s_{j+1}).$$

Example 3.6. Consider the following behaviour of our Hubble telescope model:

$$\sigma = \mathbf{6}, 3, \mathbf{5}, 3, \mathbf{4}, 1, \mathbf{3}, \frac{1}{2}, \mathbf{2}, \frac{1}{1000}, \mathbf{sleep2}, 3, \mathbf{sleep1}, \frac{1}{2}, \mathbf{6} \ldots$$

where the boldface elements denote states and the other numbers denote the state residence times. So, for example, $\sigma[3] = 3$, $\sigma@9 = sleep2$. Consider the reward function ρ defined earlier, that is, the number of measurements by the HSP per unit of time. In particular, all impulse rewards in this case are zero. Hence we get

$$y(\sigma, 9) = 3{\cdot}\rho(6) + 3{\cdot}\frac{4}{5}{\cdot}\rho(6) + 1{\cdot}\frac{3}{5}{\cdot}\rho(6) + \frac{1}{2}{\cdot}\frac{2}{5}{\cdot}\rho(6) + \frac{1}{1000}{\cdot}\frac{1}{5}{\cdot}\rho(6) + \frac{449}{1000}{\cdot}0,$$

which means that in total about $193{\cdot}10^{10}$ observations have been made in the first 9 operational years of the telescope.

We can now define a probability space on measurable sets of maximal paths of an MRM using cylinder sets. We will not dwell here on the technical details of this definition (see Baier *et al.* (2003) for more information), but just use Pr to denote the probability measure on the induced sigma-algebra. Let the transient probability

$$\pi(s, s', t) \;=\; \Pr_s\{\sigma \in Path(s) \mid \sigma@t = s'\}$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

denote the probability of being in state s' at time t given initial state s. Here, we use $\Pr_s$ to denote the probability measure on measurable sets of maximal paths with state s as starting state. The steady-state probability of being in state s', given that the MRM starts in state s, is given by

$$\pi(s, s') = \lim_{t \to \infty} \pi(s, s', t).$$

This limit always exists for finite MRMs. If the steady-state distribution does not depend on the starting state s, we simply write $\pi(s')$ instead of $\pi(s, s')$. For $S' \subseteq S$, we use

$$\pi(s, S') = \sum_{s' \in S'} \pi(s, s')$$

to denote the steady-state probability for set S'. In a similar way, $\pi(s, S', t)$ is defined for time t.

Definition 3.7 (expected state rewards). The *expected long-run reward rate* while residing in state s' having started in s is defined by

$$\rho(s, s') = \left(\rho(s') \; + \; \sum_{u \in S} \mathbf{P}(u, s') \cdot \iota(u, s') \right) \cdot \pi(s, s'). \tag{1}$$

The *expected instantaneous reward rate at time t* while residing in s' having started in s is given by

$$\rho(s, s', t) = \left(\rho(s') \; + \; \sum_{u \in S} \mathbf{P}(u, s') \cdot \iota(u, s') \right) \cdot \pi(s, s', t). \tag{2}$$

The *expected accumulated reward at time t* while residing in state s' having started in s is defined by

$$\mathrm{EY}(s, s', t) \; = \; \int_0^t \rho(s, s', x)\, dx. \tag{3}$$

To explain the definition of the expected long-run reward rate for state s', intuitively, the term $\rho(s') \cdot \pi(s, s')$ gives the expected state reward rate at s' for an infinite time horizon. In order to take the impulse rewards into account, we consider the average reward to reach s' from its predecessors and scale this with $\pi(s, s')$, the frequency of visiting s' in the long run. The expected instantaneous reward rate at time t is defined analogously by replacing the steady-state probability $\pi(s, s')$ by the transient-state probability $\pi(s, s', t)$.

The above notions can be generalised to sets of states in the following straightforward manner. For $S' \subseteq S$, let

$$\rho(s, S') = \sum_{s' \in S'} \rho(s, s'),$$

and, similarly,

$$\rho(s, S', t) = \sum_{s' \in S'} \rho(s, s', t)$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

and

$$\mathrm{EY}(s, S', t) = \sum_{s, \in S'} \mathrm{EY}(s, s', t).$$

In addition, for time interval I, let

$$\mathrm{EY}(s, S', I) = \int_I \rho(s, S', x)\, dx.$$

Example 3.8. We consider some expected reward measures for the Hubble telescope. For this example, it does not make sense to combine any of the impulse and state reward structures, so we will just focus on one of them.

First, we consider the expected long-run average angular measurement (the measure of Equation (1)) of the telescope using reward function $\widehat{\rho}$. This number turns out to be zero for all states. This is because we will always finally enter the state '*crash*' and never leave it again. Thus, in the long run, the MRM will almost surely be in this state. Because in this state no measurements are possible, the final value is indeed zero.

Next we consider the average rate of measurements at time t (the measure of Equation (2)), using reward structure ρ. We obtain

$$\begin{aligned}
\rho(6,1,2) &\approx 0.0 \cdot 0.0000004 &&\approx 0\\
\rho(6,2,2) &\approx 6.4 \cdot 0.0001938 &&\approx 0.00124032\\
\rho(6,3,2) &\approx 12.8 \cdot 0.0654052 &&\approx 0.83718656\\
\rho(6,4,2) &\approx 19.2 \cdot 0.2217154 &&\approx 4.25693568\\
\rho(6,5,2) &\approx 25.6 \cdot 0.4016865 &&\approx 10.2831744\\
\rho(6,6,2) &\approx 32.0 \cdot 0.3082933 &&\approx 9.8653856\\
\rho(6,\mathit{sleep2},2) &\approx 0.0 \cdot 0.0026254 &&\approx 0\\
\rho(6,\mathit{sleep1},2) &\approx 0.0 \cdot 0.0000761 &&\approx 0\\
\rho(6,\mathit{crash},2) &\approx 0.0 \cdot 0.0000039 &&\approx 0
\end{aligned}$$

If we sum up all these values to $\rho(6, S, 2) = \sum_{s \in S} \rho(6, s, 2)$, we find that the average rate of measurement at time 2 having started in state 6 is about 25.24392256.

Similarly, we can consider the expected total operation cost until time 2 (measure of Equation (3)) when using the instantaneous reward ι. It turns out that

$$\mathrm{EY}(s, S, 2) \approx 4812316.14144378.$$

This arguably high cost is due to the high cost of a crash. Figure 2 plots the average number of measurements (using the reward structure ρ and the measure of Equation (2)) and the expected cost (using the reward structure ι and the measure of Equation (3)) for time bounds 0 through 7.

4. Specifying performability

4.1. *A logic for performability guarantees*

The first step in our approach is to enable the specification of measures of interest, in particular, a broad range of performability measures, on Markov reward models. In order

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

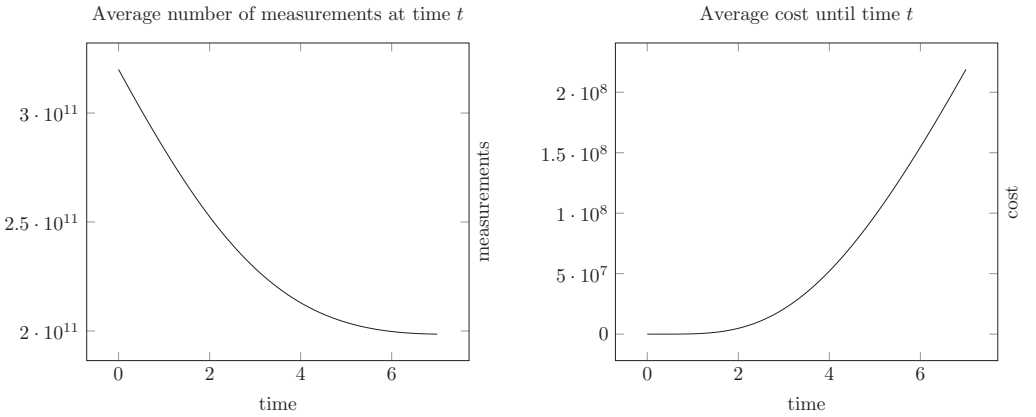


Fig. 2. Plots for the expected reward at or until time t for the Hubble telescope.

to do this, we adopt temporal logic, in particular, branching temporal logics. As a basis, we use CTL (Computation Tree Logic (Clarke *et al.* 1986)), which is an extension of propositional logic that allows us to express properties that refer to the relative order of events. Statements can be made about either states or paths, that is, the sequences of states that model the system evolution. While CTL alllows us to state properties such as 'all paths only visit legal states and eventually end up in a goal state', CSRL (*Continuous Stochastic Reward Logic*) additionally allows us to specify:

(1) the *likelihood* with which certain behaviours occur;

(2) the *time frame* in which certain events should happen;

(3) the *costs* (or rewards) that are allowed to be made.

State formulae (ranged over by capital Greek letters) are formulae in standard propositional logic with tt (true) and atomic propositions being the base cases. Recall that atomic propositions are used as state labels in our MRMs. This corresponds to the first three clauses in Definition 4.1. Each state formula Φ induces a set of states

$$\mathsf{Sat}^{\mathscr{M}}(\Phi) = \{\, s \in S \mid s \models \Phi \,\}$$

satisfying the formula. In order to address long-run probabilities, expected reward rates and expected accumulated reward, we use $\mathbb{L}$, $\mathbb{E}$ and $\mathbb{C}$ as operators in our logic. They take as basis a set of states characterised by the formula Φ, and an interval. For instance, $\mathbb{L}_K(\Phi)$ holds if the probability in the long run of being in a state in $\mathsf{Sat}^{\mathscr{M}}(\Phi)$ lies in the interval $K \subseteq [0,1]$, and $\mathbb{E}^t_J(\Phi)$ holds if the expected reward rate in $\mathsf{Sat}^{\mathscr{M}}(\Phi)$ at time t lies in J. Finally, the standard next and until operators are used as building blocks for our path formulae, except that both are enriched with two intervals: one constraining the elapsed time along a path and the other constraining its accumulated reward. The state formula $\mathbb{P}_K(\varphi)$ then holds in state s whenever the probability of all paths starting in state s that fulfil path formula φ lies in K. Pulling this all together, we get the following syntax.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Definition 4.1 (CSRL syntax). Let $I, J, K \subseteq \mathbb{R}_{\geqslant 0}$ be non-empty intervals (with rational bounds allowing intervals of the form $[c, \infty)$) with $K \subseteq [0,1]$ and $t \in \mathbb{R}_{\geqslant 0}$. The syntax of CSRL-formulae over the set of atomic propositions AP is defined inductively as follows:

— tt is a state formula.
— Each atomic proposition $a \in AP$ is a state formula.
— If Φ and Ψ are state formulae, then so is $\Phi \wedge \Psi$.
— If Φ is a state formula, then so are $\neg\Phi$, $\mathbb{L}_K(\Phi), \mathbb{E}_J(\Phi), \mathbb{E}^t_J(\Phi)$ and $\mathbb{C}^I_J(\Phi)$.
— If φ is a path formula, then $\mathbb{P}_K(\varphi)$ is a state formula.
— If Φ and Ψ are state formulae, then $X^I_J\,\Phi$ and $\Phi\,\mathscr{U}^I_J\,\Psi$ are path formulae.

We refer to the sublogic of CSRL that does not contain $\mathbb{E}$ and $\mathbb{C}$ as CSRL$^-$.

Definition 4.2 (CSRL semantics). The relation $\models$ for CSRL state formulae is defined by

$$
\begin{aligned}
&s \models \mathrm{tt} \\
&s \models a && \text{iff } a \in L(s) \\
&s \models \neg\Phi && \text{iff } s \not\models \Phi \\
&s \models \Phi \wedge \Psi && \text{iff } s \models \Phi \wedge s \models \Psi \\
&s \models \mathbb{L}_K(\Phi) && \text{iff } \pi(s, \mathsf{Sat}^{\mathscr{M}}(\Phi)) \in K \\
&s \models \mathbb{E}_J(\Phi) && \text{iff } \rho(s, \mathsf{Sat}^{\mathscr{M}}(\Phi)) \in J \\
&s \models \mathbb{E}^t_J(\Phi) && \text{iff } \rho(s, \mathsf{Sat}^{\mathscr{M}}(\Phi), t) \in J \\
&s \models \mathbb{C}^I_J(\Phi) && \text{iff } \mathrm{EY}(s, \mathsf{Sat}^{\mathscr{M}}(\Phi), I) \in J \\
&s \models \mathbb{P}_K(\varphi) && \text{iff } \Pr(s \models \varphi) \in K \text{ where } \Pr(s \models \varphi) = \Pr_s\{\, \sigma \in \mathit{Path}(s) \mid \sigma \models \varphi \,\}.
\end{aligned}
$$

For path formulae and path $\sigma = s_0, t_0, s_1, t_1, s_2, t_2, \ldots$, the relation $\models$ is defined by

$$
\begin{aligned}
&\sigma \models X^I_J\,\Phi && \text{iff } \sigma[1] \text{ is defined and } \sigma[1] \models \Phi \text{ and } t_0 \in I \text{ and } y(\sigma, t_0) \in J \\
&\sigma \models \Phi\,\mathscr{U}^I_J\,\Psi && \text{iff } \exists t \in I.\, (\sigma@t \models \Psi \wedge (\forall t' \in [0,t).\, \sigma@t' \models \Phi) \wedge y(\sigma, t) \in J).
\end{aligned}
$$

Here, $\Pr_s$ denotes the probability measures of the sigma-algebra on the set of maximal paths in an MRM starting in state s – the definition is fairly standard and can be found in Baier *et al.* (2003). Stated in words, the path $\sigma = s_0, t_0, s_1, t_1, \ldots$ satisfies the formula $X^I_J\,\Phi$ whenever s_1 satisfies Φ, t_0 lies in I, and the earned reward $\rho(s_0)\cdot t_0 + \iota(s_0, s_1)$ lies in J. Path σ fulfils $\Phi\,\mathscr{U}^I_J\,\Psi$ whenever after t time units (with $t \in I$), a Ψ-state is reached, and prior to that, only Φ-states are visited. Recall that $\sigma@t$ denotes the current state in σ at time instant t, and that $y(\sigma, t)$ denotes the accumulated reward along the prefix of σ up to time t.

We will use two convenient abbreviations as follows. Let $\Diamond^I_J\,\Phi$ denote $\mathrm{tt}\,\mathscr{U}^I_J\,\Phi$. Given that tt holds in any state, this formula requires a state satisfying Φ that is eventually reached at some time point $t \in I$ such that the accumulated reward up to t lies in the interval J. The logical dual of $\Diamond$, denoted $\Box$, is defined by

$$\mathbb{P}_{\geqslant p}\big(\Box^I_J\Phi\big) \;\equiv\; \mathbb{P}_{\leqslant p}\big(\Diamond^I_J\neg\Phi\big).$$

We adopt the notational convention that intervals of the form $[0, \infty)$ are omitted from the modalities next, until, $\Diamond$ and $\Box$. Also, as above, we abbreviate probability intervals, for example, $[0, p]$ is written as $\leqslant p$ and $[0, p)$ as $< p$. For conciseness, we write '$= t$' for point

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

intervals of the form $[t, t]$ and '$\leqslant t$' for intervals of the form $[0, t]$. Similar conventions are adopted for reward intervals.

Example 4.3. We can specify some measures of interest for our Hubble telescope, where we concentrate on the reward function $\widehat{\rho}$ denoting the number of observations per time unit. For instance,

$$\mathbb{P}_{\geqslant 0.99999}(\Box^{\leqslant 32} \neg crash)$$

requires with at least 'five nines' likelihood that the telescope does not crash within the next 32 years. Alternatively,

$$\mathbb{P}_{\geqslant 0.9}(\Diamond^{=32} \neg(crash \vee sleep1 \vee sleep2 \vee 1))$$

holds when after 32 years (a transient probability), the telescope has at least two operational gyroscopes (and thus can make observations) in at least 9 out of 10 cases. The fact that in such an observational mode, the average coverage of the WFC3 exceeds 130×130 arcsec is expressed by the formula

$$\mathbb{E}^{32}_{>130^2}(\neg(crash \vee sleep1 \vee sleep2 \vee 1)).$$

This should not be confused with the formula

$$\mathbb{C}^{\leqslant 32}_{>130^2}(\neg(crash \vee sleep1 \vee sleep2 \vee 1)),$$

which expresses the fact that the *total* coverage of the WFC3 in the first 32 years exceeds 130×130 arcsec. Finally, we consider the impulse reward function ι, which denotes the repair and switching costs of the telescope. The formula

$$\mathbb{P}_{\leqslant 0.000001}\left(\Diamond^{\leqslant 50}_{>2\cdot 10^8} \neg(crash \vee sleep1 \vee sleep2 \vee 1)\right)$$

asserts that with likelihood at most 10^{-6} within the first 50 years, observations can be made such that the total costs exceed $2{\cdot}10^8$. This effectively means that with extremely low probability, two repair missions are needed within the first 50 years.

4.2. *Specifying performability measures*

Given the logical framework of CSRL, we are led to ask which performability measures can be expressed. First, however, it is important to note that we use the logic to express *constraints* on the measures-of-interest rather than the measures themselves. This is because the boolean interpretation of the logic – a formula either holds or it does not. For instance, we can express the fact that the likelihood that the accumulated reward in a time interval is below or above a threshold, but cannot state the value of the cumulative reward. That is to say, CSRL provides ample means for expressing *performability guarantees* for MRMs. Indeed, CSRL allows us to express guarantees over almost all commonly known performability measures, our point of reference being the list of performability measures given in the seminal papers Smith *et al.* (1988) and Meyer and Sanders (2001), which have been widely adopted within the dependability community.

Table 1 lists the performability measures of Smith *et al.* (1988), together with a specification in CSRL that asserts a guarantee on this measure. For simplicity, we

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Table 1. *Important performability base cases and their specification in CSRL.*

	Performability measure		CSRL formula
(a)	$\pi(s,t) \in K$	transient probability of being in state s at time t	$\mathbb{P}_K(\Diamond^{=t}\ at_s)$
(b)	$\pi(s) \in K$	steady-state probability of being in state s	$\mathbb{L}_K(at_s)$
(c)	$A(t) \in K$	transient availability	$\mathbb{P}_K(\Diamond^{=t}\ \neg F)$
(d)	$\lim_{t\to\infty} A(t) \in K$	steady-state availability	$\mathbb{L}_K(\neg F)$
(e)	$\Pr\{\rho(X(t')) \geqslant 1, \forall t' \geqslant t\} \in K$	reliability	$\mathbb{P}_K(\Box^{\geqslant t}\ \neg F)$
(f)	$E[\rho(X(t))] \leqslant r$	expected reward rate at time t	$\mathbb{E}^{t}_{\leqslant r}(\mathrm{tt})$
(g)	$E[Y(t)] \leqslant r$	expected cumulative reward until time t	$\mathbb{C}^{\leqslant t}_{\leqslant r}(\mathrm{tt})$
(h)	$\Pr\{Y(t) \leqslant r\} \in K$	distribution of cumulative reward until t	$\mathbb{P}_K(\Diamond^{=t}_{\leqslant r}\ \mathrm{tt})$
(i)	$\Pr\{Y(\infty) \leqslant r\} \in K$	distribution of cumulative reward	$\mathbb{P}_K(\neg F\ \mathscr{W}_{\leqslant r}\ F)$
(j)	$\Pr\{A(t) \leqslant r\} \in K$	interval availability	(see case (m))
(k)	$E[W(t)] \leqslant r$	expected time-average cum. reward until t	$\mathbb{C}^{\leqslant t}_{\leqslant r\cdot t}(\mathrm{tt})$
(l)	$E[W(\infty)] \leqslant r$	expected time-average cum. reward	$\mathbb{E}_{\leqslant r}(\mathrm{tt})$
(m)	$\Pr\{W(t) \leqslant r\} \in K$	dist. of time-average cum. reward up to t	$\mathbb{P}_K(\Diamond^{=t}_{\leqslant r\cdot t}\ \mathrm{tt})$
(n)	$\Pr\{W(\infty) \leqslant r\} \in K$	distribution of time-average cum. reward	–

assume a system that can either be operational or failed, like in the Hubble telescope. Here, it is assumed that failed states are indicated by F (like the proposition *crash* for the HST) and operational states by $\neg F$. For measures (i) and (l) to be specified in CSL, we have to require states of F to be absorbing. Although failed and operational states are usually modelled by binary rewards (for example, zero and one for the modes failed and operational, respectively (Smith *et al.* 1988)), we use the proposition F since it provides us with more specification flexibility. The second column of Table 1 describes the constraint on performability measures while referring to the random variables $X(t)$ and $Y(t)$ that describes the state of the MRM at time t and the accumulated reward at time t, respectively. The random variable $W(t)$ describes the time-averaged accumulated reward and is defined as $Y(t)/t$. The random variable $A(t)$ indicates the availability of the system and is equal to one if the system is operational at time t, and zero otherwise. In this case, reward one is assigned to the operational states by the state-reward structure ρ, while the non-operational states are assigned reward zero. The third column provides the formal characterisations of the performability measures, and is identical (modulo adaptations to our notation) to those in Smith *et al.* (1988).

For state s, the atomic proposition at_s uniquely characterises state s, that is, it only holds in state s and not in any other state. Formulae (a) and (b) are guarantees on transient-state and steady-state probabilities. The transient availability at a certain time instant t (measure (c)) expresses (a bound on) the probability of not being in a failed state at time t. Using the $\mathbb{L}$-operator, this can also be generalised to infinite time horizons – see measure (d). The reliability measure (e) expresses a probability that the system is up, that is, $\rho(X(t)) \geqslant 1$ from a certain time instant t on. This time constraint is represented by the time bound of the always (that is, $\Box$) operator, while being reliable is (as before) indicated by $\neg F$. Note that the measures (a) through (e) can also be expressed in CSL (Baier *et al.* 2003).

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Measures (f) and (g) are straightforward applications of the $\mathbb{E}$ and $\mathbb{C}$ operators, respectively. As there is no need to select a certain set of states, the state subformula simply equals true. Measure (h) expresses the simultaneous distribution of the accumulated reward against time, that is, it expresses the probability for the reward accumulated at time t to be at most r. This measure is also known as *Meyer's performability distribution* (Meyer 1980). As there is no restriction imposed on the type of state reached at time t, the subformula true is used. For an infinite time horizon, the accumulated reward until failure is typically considered. This is expressed by measure (i).

Measure (j) is a special case of measure (m) in which only failed and operational states are distinguished (typically by binary rewards). The CSRL-formula for measure (m) can thus also be applied to (j) without modification. The CSRL-formula for guarantees on the measures (k) and (m) follow directly from the fact that $W(t) = Y(t)/t$ (for finite t). Note that the reward bound is $r{\cdot}t$ since an accumulated reward $r{\cdot}t$ over the interval $[0, t]$ yields a time-averaged accumulated reward r. Measure (n) cannot be specified in CSRL since it is only possible to refer to the *expected* time-averaged cumulative reward (which is identical to the expected steady-state reward), and not to the *probability* that the time-averaged cumulative reward is below a given value.

To conclude, we emphasise that CSRL allows us to specify much more complex performability measures than those listed in Table 1. For instance, for cases (f), (g) and (h), we may select a subset of states, for example, those in which the system is guaranteed to offer a certain quality-of-service, that are of interest at time instant t (rather than considering any state). Moreover, the general syntax of the logic means that nesting of measures is supported naturally. This allows us to specify non-trivial properties such as the transient probability at time t to be in a state s, say, that guarantees that almost surely the accumulated reward (when starting in s) within a given deadline d is at most r exceeds 0.99 can be expressed by

$$\mathbb{P}_{>0.99}\left(\Diamond^{=t}\ \mathbb{P}_{=1}\left(\Diamond^{\leqslant d}_{\leqslant r}\ \mathrm{tt}\right)\right)$$

4.3. *Duality*

Inspired by an observation in Beaudry (1978), time and reward constraints are dual in the sense that they can be swapped, provided the MRM is 'rescaled' at the same time. To discuss this effect in more detail, we will ignore impulse rewards, that is, we consider MRMs for which $\iota(s, s') = 0$ for every pair of states s, s'. For simplicity, we will also omit the component ι from an MRM in this section. Given an MRM $\mathscr{M} = (\mathscr{C}, \rho)$ with $\rho(s) > 0$ for all states s, we consider the dual MRM $\mathscr{M}^*$ that results from $\mathscr{M}$ by adapting the exit rates and reward function such that the reward units in state s in $\mathscr{M}$ correspond to the time units in state s in $\mathscr{M}^*$, and *vice versa*.

Definition 4.4 (dual MRM). Let MRM $\mathscr{M} = (S, \mathbf{R}, L, \rho)$ with $\rho(s) > 0$ for all states $s \in S$. The *dual* MRM is

$$\mathscr{M}^* = (S, \mathbf{R}^*, L, \rho^*)$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

where

$$\mathbf{R}^*(s,s') = \frac{\mathbf{R}(s,s')}{\rho(s)}$$
$$\rho^*(s) = \frac{1}{\rho(s)}.$$

Intuitively, the transformation of $\mathscr{M}$ into $\mathscr{M}^*$ stretches the residence time in state s by a factor that is proportional to the reciprocal of its reward $\rho(s)$ if $0 < \rho(s) < 1$. The reward function is changed similarly. Thus, all states s for which $\rho(s) < 1$ are accelerated, while all states s with $\rho(s) > 1$ are slowed down. The residence of t time units in MRM $\mathscr{M}^*$ might be interpreted as the earning of t reward in state s in $\mathscr{M}$, or (conversely) an earning of a reward r in state s in MRM $\mathscr{M}$ corresponds to a residence of r time units in $\mathscr{M}^*$. The proof of the following theorem can be found in Baier *et al.* (2010a).

Theorem 4.5 (duality theorem). For MRM $\mathscr{M} = (S, \mathbf{R}, L, \rho)$ with $\rho(s) > 0$ for all $s \in S$ and CSRL$^-$ state formula Φ,

$$\mathsf{Sat}^{\mathscr{M}}(\Phi) \quad = \quad \mathsf{Sat}^{\mathscr{M}^*}(\Phi^*),$$

where Φ^* is obtained from Φ by swapping I and J in every subformula in Φ of the form X_J^I or $\mathscr{U}_J^I$.

This duality result turns out to be of practical importance for checking formulae of the form $\mathbb{P}_K(\Phi_1\, \mathscr{U}_J^I\, \Phi_2)$ where $I = [0, \infty)$. In such formulae, there is no time constraint, just a constraint on the accumulated (state) reward. Thanks to the duality theorem, the 'dual' formula $\mathbb{P}_K(\Phi_1\, \mathscr{U}_I^J\, \Phi_2)$ can be checked on the dual MRM, and efficient procedures exist for such time-bounded formulae. A major restriction, however, is that all state rewards must be strictly positive: this duality result does not hold if $\mathscr{M}$ contains states equipped with a zero reward since the reverse of earning a zero reward in $\mathscr{M}$ when considering Φ should correspond to a residence of 0 time units in $\mathscr{M}^*$ for Φ^*, which, since the advance of time in a state cannot be halted, is in general impossible. However, the result of Theorem 4.5 applies to some other practical cases: for example, when for each subformula of the form $\Phi_1\, \mathscr{U}_J^I\, \Phi_2$, we have

$$J = [0, \infty)$$

or

$$\mathsf{Sat}^{\mathscr{M}}(\Phi_1) \subseteq \{\, s \in S \mid \rho(s) > 0 \,\},$$

that is, all Φ_1-states are positively rewarded. The intuition is that either the reward constraint (that is, time constraint) is trivial in Φ (in Φ^*) or zero-rewarded states are not involved in checking the reward constraint. We define $\mathscr{M}^*$ here by setting

$$\mathbf{R}^*(s,s') = \mathbf{R}(s,s')$$

and

$$\rho^*(s) = 0$$

when $\rho(s) = 0$, otherwise it is defined as above.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Suppose now that the given MRM $\mathscr{M}$ is strongly connected. Then so is $\mathscr{M}^*$, and the steady-state probabilities in $\mathscr{M}$ and $\mathscr{M}^*$ do not depend on the starting state. Hence, the CSRL formula $\mathbb{L}_K(\Phi)$ either holds for all states in $\mathscr{M}$ or for none of them. The same holds for $\mathscr{M}^*$ and CSRL formulae of the form $\mathbb{E}_J(\Phi)$.

Let $\pi(s)$ and $\pi^*(s)$ denote the steady-state probability for state s in $\mathscr{M}$ and $\mathscr{M}^*$, respectively. Then $\big(\pi(s)\big)_{s\in S}$ is the unique vector such that

$$\sum_{s\in S} \pi(s) = 1$$

$$\sum_{u\in S} \pi(u)\cdot \mathbf{R}(u,s) = \pi(s)\cdot \mathrm{E}(s).$$

Recall that

$$\mathrm{E}(s) = \sum_{u\in S} \mathbf{R}(s,u)$$

is the exit rate of state s.

Similarly, $\big(\pi^*(s)\big)_{s\in S}$ is the unique vector such that

$$\sum_{s\in S} \pi^*(s) = 1$$

$$\sum_{u\in S} \pi^*(u)\cdot \mathbf{R}^*(u,s) = \pi^*(s)\cdot \mathrm{E}^*(s)$$

where

$$\mathrm{E}^*(s) = \sum_{v\in S} \mathbf{R}^*(s,v).$$

Since

$$\mathbf{R}(s,v) = \rho(s)\cdot \mathbf{R}^*(s,v),$$

we get

$$\mathrm{E}(s) = \rho(s)\cdot \mathrm{E}^*(s).$$

We now define

$$q = \sum_{u\in S} \pi(u)\cdot\rho(u)$$

$$\chi(u) = \frac{1}{q}\cdot \pi(u)\cdot\rho(u) \quad \text{for all states } u\in S.$$

So,

$$\sum_{u\in S}\chi(u) \;=\; \frac{1}{q}\cdot \underbrace{\sum_{u\in S}\pi(u)\cdot\rho(u)}_{=q} \;=\; 1,$$

and

$$\chi(u)\cdot\mathbf{R}^*(u,s) = \frac{1}{q}\cdot\pi(u)\cdot\rho(u)\cdot\frac{\mathbf{R}(u,s)}{\rho(u)}$$

$$= \frac{1}{q}\cdot\pi(u)\cdot\mathbf{R}(u,s).$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Hence:

$$\begin{aligned}\sum_{u\in S}\chi(u)\cdot \mathbf{R}^*(u,s) &= \frac{1}{q}\cdot\sum_{u\in S}\pi(u)\cdot\mathbf{R}(u,s)\\ &= \frac{1}{q}\cdot\pi(s)\cdot \mathrm{E}(s)\\ &= \frac{1}{q}\cdot\pi(s)\cdot\rho(s)\cdot \mathrm{E}^*(s)\\ &= \chi(s)\cdot \mathrm{E}^*(s).\end{aligned}$$

We therefore conclude that $\chi(s) = \pi^*(s)$ for all states s. As a consequence, we get the duality of the long-run average operator $\mathbb{L}_K(\cdot)$ and the expected long-run reward operator $\mathbb{E}_J(\cdot)$ in the following sense.

Theorem 4.6 (duality of long-run average and expected long-run reward). If MRM $\mathscr{M}$ is strongly connected, we have

$$\mathsf{Sat}^{\mathscr{M}}(\mathbb{L}_K(\Phi)) \;=\; \mathsf{Sat}^{\mathscr{M}^*}(\mathbb{E}_{K^*}(\Phi^*)),$$

where $K^* = \{\, x \mid x\cdot q \in K \,\}$ and q is defined as above.

Proof. The calculation above shows that

$$\pi^*(u) = \chi(u) = \frac{1}{q}\cdot\rho(u)\cdot\pi(u)$$

for all states $u \in S$. This yields

$$\begin{aligned} s \in \mathsf{Sat}^{\mathscr{M}}(\mathbb{L}_K(\Phi)) \quad &\text{iff} \quad \sum_{u\in Sat^{\mathscr{M}}(\Phi)}\pi(u)\in K\\ &\text{iff} \quad \sum_{u\in Sat^{\mathscr{M}}(\Phi)}\frac{1}{q}\cdot\pi(u)\in K^*\\ &\text{iff} \quad \sum_{u\in Sat^{\mathscr{M}^*}(\Phi^*)}\pi^*(u)\cdot\frac{1}{\rho(u)}\in K^*\\ &\text{iff} \quad \sum_{u\in Sat^{\mathscr{M}^*}(\Phi^*)}\pi^*(u)\cdot\rho^*(u)\in K^*\\ &\text{iff} \quad s\in \mathsf{Sat}^{\mathscr{M}^*}(\mathbb{E}_{K^*}(\Phi^*)).\end{aligned}$$

□

The practical relevance of this theorem is as follows. Efficient algorithms exist to check whether $s \models \mathbb{L}_K(\Phi)$. In fact, for strongly connected MRMs, this boils down to computing steady-state probabilities, which can be done by solving a system of linear equations in the size of the state space. The above theorem shows that in order to check whether $s \models \mathbb{E}_K(\Phi)$, the same procedure can be followed on the dual MRM.

5. Model checking

Suppose we are confronted with a (finite) Markov reward model originating from some high-level formalism such as a stochastic reward net, a stochastic activity network or

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

a Markovian queueing network, and a performability guarantee formulated in the logic described above. So how do we compute the set of states satisfying this guarantee? The basic computational procedure is a simple *recursive descent* over the logical formula. This means, basically, that the formula is broken down into its subformulae, that the computation starts with the simplest subformulae, and once this step is completed, then considers the formulae that combine subformulae using a single operator, and so on, until the entire formula is captured. Considering the parse tree of the formula, this computation is just a bottom-up traversal over the parse tree, where at each node (representing a subformula) a single algorithm is invoked. In this way, formulae of arbitrary complexity can be treated in a uniform manner. This recursive descent mechanism is adopted from *model-checking* algorithms (Baier and Katoen 2008; Clarke *et al.* 1986). The main difference compared with traditional model-checking algorithms, where all computations involve graph algorithms, fixed-point computations and the like, is that in our setting, *numerical* algorithms are needed to reason about the probabilities and reward aspects. To achieve this, well-known techniques for solving systems of linear equations, determining long-run probabilities, and transient probabilities (such as uniformisation (Gross and Miller 1984)) are embedded in the tree traversal as subroutines.

We will explain the computational procedure in a bit more detail by means of an example. Consider the formula

$$\mathbb{P}_{\geqslant 0.99999}\left(\Box_{\leqslant 10}^{\leqslant 500}\, \mathbb{E}_{[90,100]}(\mathit{operational} \wedge \neg \mathit{idle})\right).$$

We are thus interested in computing the states in an MRM from which, with at least 'five-nine' dependability (that is, probability at least 0.99999), the computation will only visit certain states in the next 500 time units while consuming in total at most 10 units of reward. The states in which the system has to reside for the next 500 time units should guarantee that starting from there, the expected costs to keep the system, when in equilibrium, functioning in operational mode (that is, non-idling) are between 90 and 100 reward units. Note that we assume that *operational* and *idle* are propositions of the MRM under study. Before continuing with the explanation of our computational procedure, it is worth spending a few moments thinking about how to determine the required states – it is not easy.

We will start by explaining the model-checking algorithm. We begin by determining the subformulae of the guarantee at hand. The above formula has the following subformulae:

— $\mathit{operational}$
— idle
— $\neg \mathit{idle}$
— $\mathit{operational} \wedge \neg \mathit{idle}$
— $\mathbb{E}_{[90,100]}(\mathit{operational} \wedge \neg \mathit{idle})$
— and the entire formula.

The computation starts with the simplest subformulae, that is, *operational* and *idle*. Since these are the most elementary formulae of our logical framework, we will assume that their validity in any state of the model can be determined directly. In the case of a stochastic reward net, for instance, the *operational* and *idle* states could be states with a

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

certain number of tokens in a certain place, whereas in a Markovian queueing network, they could refer to states with a certain queue occupancy. The computation of the set of states satisfying *idle*, denoted by $\mathsf{Sat}(idle)$, is therefore straightforward. The same applies to computing $\mathsf{Sat}(operational)$.

The parse tree traversal proceeds by considering the next-to-simplest subformulae, that is, $\neg idle$. This set is simply obtained by complementing $\mathsf{Sat}(idle)$, that is,

$$\mathsf{Sat}^{\mathcal{M}}(\neg idle) = S - \mathsf{Sat}^{\mathcal{M}}(idle),$$

where S is the entire set of states in the MRM under consideration. We thus see that the logical operator negation is interpreted using its set-theoretical analogue: complementation. The same holds for conjunction (and the other propositional logical operators). Accordingly,

$$\mathsf{Sat}^{\mathcal{M}}(operational \wedge \neg idle) = \mathsf{Sat}^{\mathcal{M}}(operational) \cap \mathsf{Sat}^{\mathcal{M}}(\neg idle).$$

For brevity, we will let $U = \mathsf{Sat}^{\mathcal{M}}(operational \wedge \neg idle)$. The next step in the procedure is to compute the states satisfying $\mathbb{E}_{[90,100]}(\mathbf{1}_U)$, where $\mathbf{1}_U$ is the characteristic function of the set U, that is, $\mathbf{1}_U(s) = 1$ if and only if $s \in U$. That is to say, we have to determine the set of states from which the system can be started and that guarantee an expected long-run reward in states in U to be in the interval $[90, 100]$. Since we are following a recursive descent procedure, the set $\mathbf{1}_U$ has already been computed. Following Definition 3.7, we proceed by determining

$$\rho(s, s') = \left(\rho(s') + \sum_{u \in S} \mathbf{P}(u, s') \cdot \iota(u, s') \right) \cdot \pi(s, s')$$

where $\pi(s, s')$ denotes the steady-state probability of state s' when starting in state s. We then have

$$s \in \mathsf{Sat}^{\mathcal{M}}\left(\mathbb{E}_{[90,100]}(operational \wedge \neg idle)\right) \quad \text{iff} \quad \sum_{u \in U} \rho(s, u) \in [90, 100].$$

In the general case, the steady-state probabilities depend on the initial state – certain states may not even be reachable, depending on where we start. Using a graph analysis, which is basically a depth-first traversal through the graph underlying the MRM, we then determine the strongly connected components (SCCs) that are *terminal*. A terminal SCC is a subgraph in which each state can reach any other state within the subgraph, but no other states. Hence, once we reach such a terminal SCC, we cannot leave it again; we can only cycle through that component, and can never escape it. Furthermore, the steady-state probabilities of each state in a terminal SCC are independent of which state in the terminal SCC we start from. For each such component, the steady-state probabilities are determined by solving a system of linear equations, which can be done using standard means. For terminal SCC T, we let $\pi_T(s')$ be the steady-state probability for being in state s' in T in equilibrium under the condition that we start in some state of T. In order to determine $\pi(s, s')$, we will now determine the probability

$$x_{s,T} = \Pr_s\{s \models \Diamond T\}$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

of reaching the terminal SCC T from state s. This is done for all terminal SCCs, and can be computed by solving the following system of linear equations for all states $s \in S$ from which T is reachable:

$$x_{s,T} = \begin{cases} 1 & \text{if } s \in T \\ \sum_{s' \in S} \mathbf{P}(s,s') \cdot x_{s',T} & \text{otherwise.} \end{cases}$$

If T is not reachable from s, the $x_{s,T}$ is set to zero. The above equation can be solved using standard means (for example, Gauss–Seidel).

As a final step in the verification process, we will now determine the set of states satisfying the formula

$$\mathbb{P}_{\geqslant 0.99999}\left(\Box_{\leqslant 10}^{\leqslant 500} \mathbf{1}_G\right)$$

where

$$G = \mathsf{Sat}^{\mathscr{M}}\left(\mathbb{E}_{[90,100]}(\mathit{operational} \wedge \neg \mathit{idle})\right)$$

is as computed before. Note first that this formula is equivalent to

$$\mathbb{P}_{\leqslant 0.00001}\left(\Diamond_{\leqslant 10}^{\leqslant 500} \mathbf{1}_{\overline{G}}\right)$$

where $\overline{G}$ is a shorthand for $S \setminus G$, that is, the complement of G with respect to the total set of states S. We now consider a path $s_0\, s_1\, s_2 \ldots$ through the MRM. (For simplicity, we will omit the state residence times.) Once a state in $\overline{G}$ has been reached, knowing which states will be visited afterwards is completely irrelevant. That is, when $s_k \in \overline{G}$, the fact of whether later states s_j (for $j > k$) are in G or not has no effect on whether the path satisfies $\Diamond \mathbf{1}_{\overline{G}}$. This means we can treat $s_k \in \overline{G}$ as an *absorbing* state. This applies to all states in $\overline{G}$. So, prior to doing any computational step, the MRM $\mathscr{M}$ is changed into $\mathscr{M}'$ by making all states in $\overline{G}$ absorbing and turning their reward into zero. The number of reachable states in $\mathscr{M}'$ is never larger than that in $\mathscr{M}$. It is not difficult to see that it now suffices to check the formula

$$\mathbb{P}_{\leqslant 0.00001}(\Diamond_{\leqslant 10}^{=500} \mathbf{1}_{\overline{G}})$$

in the newly obtained MRM $\mathscr{M}'$. This formula should be compared with the formula for the distribution of the cumulative reward – see performability measure (h) in Table 1. The shapes are very similar, and, indeed, we have transformed the verification of our original formula into a (standard) performability measure calculation on another MRM. Thus we are left to determine the transient reward probability to be in a state in $\overline{G}$ at time 500 when starting from an arbitrary state in the (transformed) MRM. Using Meyer's original terminology, where $X(t)$ denotes the state of the MRM at time t and $Y(t)$ denotes the random variable for the accumulated reward until time t, we get

$$s \models \mathbb{P}_{\leqslant 0.00001}(\Diamond_{\leqslant 10}^{=500} \mathbf{1}_{\overline{G}}) \quad \text{iff} \quad \Pr\{Y(500) \leqslant 10 \wedge X(500) \in \overline{G} \mid X(0) = s\} \leqslant 0.00001.$$

A discussion of the numerical techniques required to obtain this transient reward measure is beyond the scope of this paper – see Baier *et al.* (2010a) for a detailed discussion of appropriate numerical procedures; here we will just mention that the techniques described in Qureshi and Sanders (1994b), Sericola (2000) and Tijms and Veldman (2000) are all potential candidates.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

6. Bisimulation

Bisimulation and simulation relations play a central role in the design of hierarchical and compositional systems, but are also used for abstraction purposes. Bisimulation and simulation relations provide formal criteria of when two systems (or two states of a system) have the same observable branching behaviour (bisimulation), or when the observable branching behaviour of one system is covered by the observable branching behaviour of another one (simulation). The underlying notion of 'observability' can refer to action names for transitions and a classification of the actions into visible and invisible ones, or it can refer to atomic propositions that serve as observables of the states. In addition, several quantitative parameters (for example, timing constraints or reward functions) can be taken into account to ensure that the values of the relevant performance measures are the same for bisimilar systems.

Inspired by Milner's seminal work (Milner 1971; Milner 1980) on (bi)simulation relations for non-probabilistic systems, Larsen and Skou (1991) introduced probabilistic bisimulation for discrete-time probabilistic systems. Roughly speaking, their notion of bisimulation equivalence rephrased for (discrete-time) Markov chains with state labels requires that bisimilar states satisfy the same atomic propositions, and that the probabilities of moving within one step to each of the bisimulation equivalence classes agree. An analogous definition of bisimulation equivalence for CTMCs (rephrased for the state-labelled approach) is that bisimilar states cannot be distinguished by the state labels and move with the same total rates to each of the bisimulation equivalence classes. This notion of bisimulation equivalence for CTMCs fits in nicely with the notion of lumpability (Howard 1971a), and can be seen as a conservative extension of bisimulation equivalence of the embedded discrete-time Markov chain in the following sense. Two states of a CTMC are bisimilar if and only if they are bisimilar in the embedded discrete-time Markov chain (DTMC) and have the same total exit rate.

In the literature, several variants of bisimulation equivalence have been proposed for discrete- and continuous-time Markov chains and extensions to these that include non-determinism (Hermanns 2002) or rewards (Bernardo and Bravetti 2001; Aldini and Bernardo 2007). These variants include abstracting from internal (invisible) steps (Segala and Lynch 1995; Baier and Hermanns 1997) and the dropping of symmetry requirements, leading to formal notions of simulation and refinement relations (Jonsson and Larsen 1991; Baier *et al.* 2005b; Caillaud *et al.* 2010).

We will focus here on the basic concept of bisimulation equivalence for the model of MRMs, that is, a state-labelled CTMC with state and impulse rewards. The formal definition of bisimulation equivalence follows the standard principle of coinduction, where conditions are first provided to define the notion of a bisimulation relation, and bisimulation equivalence is then defined as the coarsest bisimulation relation.

For state s and a set of states U, we use $\mathbf{R}(s, U)$ to denote the total rate for moving from state s to some state in U, that is,

$$\mathbf{R}(s, U) = \sum_{u \in U} \mathbf{R}(s, u).$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Thus,

$$\mathbf{R}(s, S) = E(s)$$

is the total rate for state s. Similarly, $\mathbf{P}(s, U)$ stands for the probability to move from s within one step to U, that is,

$$\mathbf{P}(s, U) = \frac{\mathbf{R}(s, U)}{E(s)}.$$

Definition 6.1 (bisimulation relation and bisimulation equivalence). Let $\mathscr{M} = (\mathscr{C}, \iota, \rho)$ be an MRM where $\mathscr{C} = (S, \mathbf{R}, L)$. A *bisimulation relation* on $\mathscr{M}$ is an equivalence relation $\mathscr{R}$ on the state space S of $\mathscr{M}$ such that for all pairs $(s_1, s_2) \in \mathscr{R}$ and all equivalence classes U of $\mathscr{R}$ the following conditions hold:

$$L(s_1) = L(s_2) \tag{1}$$

$$\mathbf{R}(s_1, U) = \mathbf{R}(s_2, U) \tag{2}$$

$$\rho(s_1) = \rho(s_2) \tag{3}$$

$$\sum_{u \in U} \mathbf{P}(s_1, u) \cdot \iota(s_1, u) = \sum_{u \in U} \mathbf{P}(s_2, u) \cdot \iota(s_2, u). \tag{4}$$

The states s_1 and s_2 of $\mathscr{M}$ are said to be bisimulation equivalent (or just *bisimilar* for short), denoted $s_1 \sim s_2$, if there exists a bisimulation relation $\mathscr{R}$ with $(s_1, s_2) \in \mathscr{R}$.

It can be shown that the relation $\sim$ is an equivalence relation that satisfies conditions (1)–(4). Hence, $\sim$ is the *coarsest* bisimulation relation.

The intuitive meaning of conditions (1)–(4) is as follows:

(1) This is a standard condition for (bi)simulation relations for state-labelled models where the 'observability' of a state s is considered to be given by the atomic propositions that hold for s. So (1) simply requires that bisimilar states are equally observable.

(2) This is a formalisation of the above-stated condition that bisimilar states of a CTMC have the same cumulative rate for moving within one step to some bisimulation equivalence class U. It is equivalent to stating that

$$\mathbf{P}(s_1, U) = \mathbf{P}(s_2, U)$$

and

$$E(s_1) = E(s_2).$$

(3) This requires that bisimilar states have the same state reward.

(4) This condition requires that the expected impulse reward earned by taking a transition to some bisimulation equivalence class U coincides for bisimilar states.

The following theorem asserts that bisimilar states yield the same performance measures when they are expressible in CSRL.

Theorem 6.2 (preservation of performance measures). If s_1, s_2 are bisimilar states of an MRM $\mathscr{M}$, then for all CSRL-formulae Φ,

$$s_1 \models \Phi \text{ iff } s_2 \models \Phi.$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

The proof of Theorem 6.2 is by structural induction and follows the proof techniques provided in, for example, Baier *et al.* (2005b) for the preservation of CSL-definable properties under bisimulation equivalence for CTMCs.

An important application of Theorem 6.2 is that it allows us to use bisimulation equivalence as a reduction technique. In order to verify a CSRL property Φ for a given MRM $\mathscr{M}$, we build the quotient $\mathscr{M}/\sim$, where the states are the bisimulation equivalence classes of the states in $\mathscr{M}$. The rate matrix $\mathbf{R}_\sim$ of $\mathscr{M}/\sim$ is given by

$$\mathbf{R}_\sim([s], U) = \mathbf{R}(s, U)$$

for each state s and bisimulation equivalence class U where $[s] = \{s' \in S : s \sim s'\}$ denotes the bisimulation equivalence class of state s. For each state s of $\mathscr{M}$, we have the labelling of $[s]$ is $L(s)$, the reward rate of $[s]$ is $\rho(s)$ and the impulse reward for the pair $([s], U)$ is given by

$$\sum_{u \in U} \mathbf{P}(s, u) \cdot \imath(s, u).$$

Obviously, conditions (1)–(4) in Definition 6.1 ensure that $\mathscr{M}/\sim$ is well defined. Furthermore, each state s of $\mathscr{M}$ is bisimilar to its bisimulation equivalence class $[s]$ in the combined MRM that results from taking the disjoint union of $\mathscr{M}$ and $\mathscr{M}/\sim$. By Theorem 6.2, s and $[s]$ satisfy the same CSRL formulae. This observation allows us to switch from $\mathscr{M}$ to its quotient $\mathscr{M}/\sim$ and to apply the model-checking techniques sketched in the previous section to $\mathscr{M}/\sim$ rather than $\mathscr{M}$. Building the quotient is often called *lumping*.

We expect that the proof techniques presented in Desharnais and Panangaden (2003) for CSL are also applicable here to show that bisimulation equivalence for MRMs is the coarsest equivalence that preserves the truth values of all CSRL formulae. This means that whenever two states satisfy the same CSRL formulae, they are bisimilar. Together with Theorem 6.2, this means that the bisimulation quotient $\mathscr{M}/\sim$ is the smallest MRM that satisfies precisely the same CSRL formulae.

Example 6.3. Consider the model of Figure 3. It is a variant of the Hubble telescope model already given previously in Figure 1. However, in contrast to the previous model, here we distinguish between the six individual gyroscopes, which may be working or not working – in the previous model, we just counted the number of gyroscopes working. However, if we are in fact only interested in properties that depend on the number of working gyroscopes, and are never interested in whether a particular gyroscope is functional, we could choose a labelling that assigns the same label to all the states of Figure 3 that have the same number of working gyroscopes, the same sleep mode and the same rewards. Consequently, the bisimulation quotient of the model in Figure 3 is the same as that for Figure 1.

Notice that the model of Figure 1 only has 9 states, whereas the model of Figure 3 has

$$\binom{6}{6} + \binom{6}{5} + \binom{6}{4} + \binom{6}{3} + 2 \cdot \binom{6}{2} + 2 \cdot \binom{6}{1} + \binom{6}{0} = 85$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

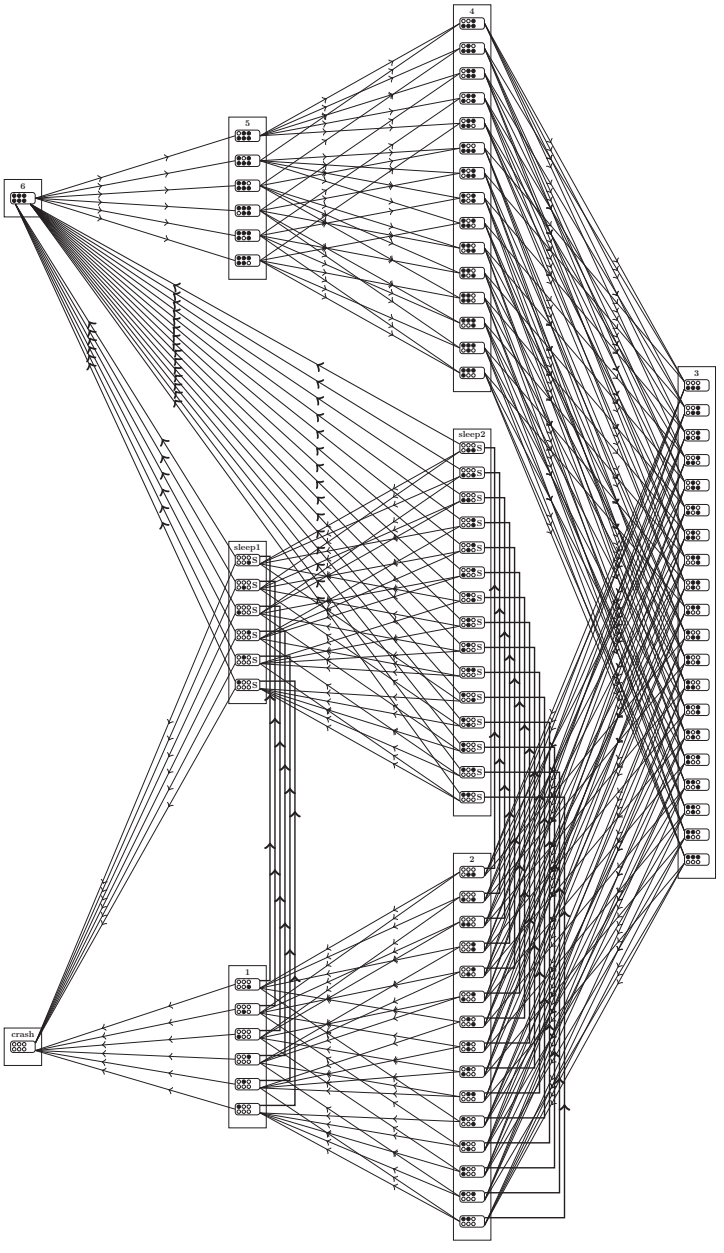


Fig. 3. Unlumped version of the Hubble telescope from Figure 1. In each state, the six gyroscopes are denoted by circles. A black circle denotes a working gyroscope, whereas a white one means that it is defective. Sleep mode is marked by '**S**'. Thin lines between states correspond to a rate of 0.1 (failure of a gyroscope), medium ones to 6 (servicing mission), and thick ones to 100 (go to sleep). Boxes containing several states and a labelling denote which state of Figure 1 the included states are lumped into.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

states. For this model, bisimulation thus yields a reduction in the number of states by a factor of ≈ 9. If we generalise the model so that it has n telescopes, the size of the state space would be $O(2^n)$ for the unlumped version and $O(n)$ for the lumped one.

7. Non-determinism

7.1. *Continuous-time Markov reward decision processes*

A continuous-time Markov decision process (CTMDP) extends CTMCs by adding non-deterministic choices. As with CTMCs, the model consists of states, and the timed behaviour is governed by exponential distributions. But unlike the case for CTMCs, each state may have a number of non-deterministic decisions of next-step distributions. The class of CTMDPs is of interest because it can be viewed as a common semantic model for various performance and dependability modelling formalisms, including generalised stochastic Petri nets (Ajmone Marsan *et al.* 1984), Markovian stochastic activity networks (Sanders and Meyer 1987) and interactive Markov chains (Hermanns and Katoen 2009).

Non-deterministic decisions are decisions that we cannot actually associate a particular probability distribution with, since it is unknown or inapplicable. Such decisions may result from the interleaved execution of concurrent systems, from underspecification of the model or from leaving out probabilities we do not have enough information about, such as user actions or certain environmental influences. Labels are usually used to distinguish the non-deterministic alternatives. Here instead, we support models where there is internal non-determinism between equally labelled next steps. In summary, a CTMDP specification consists of state transitions, corresponding distributions and a labelling function that maps transitions to labels.

Definition 7.1 (CTMDP). A *continuous-time Markov decision process* (CTMDP) $\mathscr{C}$ is a tuple $(S, Act, \mathbf{R}, L)$ with:

— S a countable set of *states*;
— Act a set of *actions*;
— $\mathbf{R} : S \times Act \times S \to \mathbb{R}_{>0}$ the rate function; and
— $L : S \to 2^{AP}$ a labelling function.

The set of actions that are enabled in state s is denoted by

$$Act(s) = \{\, \alpha \in Act \mid \exists s'.\, \mathbf{R}(s, \alpha, s') > 0 \,\}.$$

A CTMC is a CTMDP in which for each state s, $Act(s)$ is a singleton or empty. The operational behaviour of a CTMDP is similar to that of a CTMC, except that on entering state s, an action α, say, in $Act(s)$ is selected non-deterministically (unless the state is absorbing).

As we have non-deterministic decisions, we cannot talk about the probability of a model satisfying a property. Instead, probabilistic behaviour results after applying an entity that resolves the non-deterministic decisions. This entity is called a scheduler, policy or adversary. Intuitively, a scheduler acts as follows: whenever it is given a current state, it picks an enabled transition. It may do this using (or not using) probabilities, and it may

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

base the decision on the history of the process since its initialisation. This history can be considered in more or less detail, and may, for instance, consist of the sequence of states (path) visited thus far, with or without time stamps of the state changes. The scheduler may also use the information about the time it has spent in the current state in order to reconsider the decision.

The potential of such schedulers forms a natural hierarchy, with *memoryless* schedulers (which do not use any history or time information) being the smallest class, and *arbitrary* schedulers (using all the above-mentioned concepts) being the largest class. An interesting intermediate class of schedulers are *time-abstract* schedulers, which use arbitrary history information, apart from time, and arise naturally when dealing with CTMDPs resulting from abstractions of CTMCs (Katoen *et al.* 2007).

Given a CTMDP and a particular class of schedulers, the basic approach in associating a semantics with a logical state formula Φ from a logic like CSL is to demand that the formula Φ be satisfied regardless of the choice of scheduler used to turn the decision process into a stochastic process. This is closely related to the question of which scheduler maximises or minimises a given CSL path formula.

These questions have induced a considerable amount of recent work in the context of CTMDPs. The base problem considered is that of computing maximal time-bounded reachability ($\Diamond^{\leqslant t}$) probabilities. The existence of optimal schedulers has been shown for both, the arbitrary (Rabe and Schewe 2011) and the time-abstract (Brázdil *et al.* 2009) scheduler setting, the latter coming with a decision algorithm.

Of practical relevance are approximative model-checking procedures, which are related to those discussed in Section 5. An efficient approach for time-abstract schedulers has been devised that is tailored to *uniform* CTMDPs. In this model class, there is a unique rate E such that for each state and each non-deterministic alternative, the total outgoing rate of this alternative is E (Baier *et al.* 2005a). The overhead over the CTMC algorithm is linear in the maximal non-deterministic fanout. This algorithm has been generalised to locally uniform CTMCs, which are models where there is a uniform exit rate *per state* (Neuhäußer and Zhang 2010). This, in turn, is the basis for a model-checking algorithm for IMCs, *viz.* interactive Markov chains (Zhang and Neuhäußer 2010). If we relinquish the uniformity restriction entirely, but stay in the time-abstract setting, the only known algorithm has exponential complexity (Brázdil *et al.* 2009).

Approximate model checking with respect to the most general class, *viz.* arbitrary schedulers, has been a challenge until recently. A first step was a discretisation procedure for time-bounded reachability (Neuhäußer and Zhang 2010). It is only recently that an efficient algorithm has been proposed[†]. To handle properties depending on time (instantaneous rewards, time-bounded until, and so on), it uses an initial *gain vector*, the exact value of which depends on the property to be checked. Starting at the time bound (or time point) t, this value is propagated back in time along the model states, until time 0 is reached. While doing this, the time interval $[0, t]$ is divided into smaller intervals, for which the (almost) optimal decision for each state is constant. The correctness follows from a

[†] In fact, it is in a setting with state rewards and for a full CSL- (and CSRL-) like logic (Buchholz *et al.* 2011)

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

result in Miller (1968) implying that an optimal policy exists, and only a finite number of switches of the actions is needed to describe it. The algorithm returns a scheduler that maximises or minimises a reward measure over a finite or infinite time horizon.

If reward values are *zero*, and we have the appropriate *initial value* for the gain vector g_t, the problem can be exploited to arrive at a uniformisation-based approach to the computation of time bounded reachability probabilities within time t. It is easy to generalise this to the maximal reachability for a finite interval $[t', t]$, which is the key element in checking the probabilistic operator in CSL. Moreover, by computing the gain vector between $[t', t]$ with $t' > 0$, followed by a probabilistic reachability analysis for the interval $[0, t']$, we are able to compute the minimum/maximum gain vector for $[t', t]$: this then gives us a complete model-checking algorithm for CTMDPs. Experimental evidence shows that the efficiency of the new numerical approach provides a dramatic improvement over the state-of-the-art. The situation here resembles the milestones in approximate CTMC model-checking research, which initially resorted to discretisation (Baier *et al.* 1999), but only got effective and mainstream technology through the use of uniformisation (Baier *et al.* 2003).

It is not yet clear whether the CSRL-specific time-and-reward bounded operator can also be checked in this way. However, the duality result presented in Theorem 4.5 extends to (state-rewarded) CTMDPs (Baier *et al.* 2008). This means we can also check the reward-bounded but time-unbounded formulae. A lot of work is currently going on in this area, which also covers continuous-time Markov games, thereby extending the CTMDP setting to a second type of non-determinism. In the discrete-time setting, such games have been shown to be useful for obtaining under- and over-approximations of concrete models using abstraction (Kattenbelt *et al.* 2009; Wachter *et al.* 2007). We anticipate similar applications in the continuous-time setting in the near future.

8. Case study: the Google file system

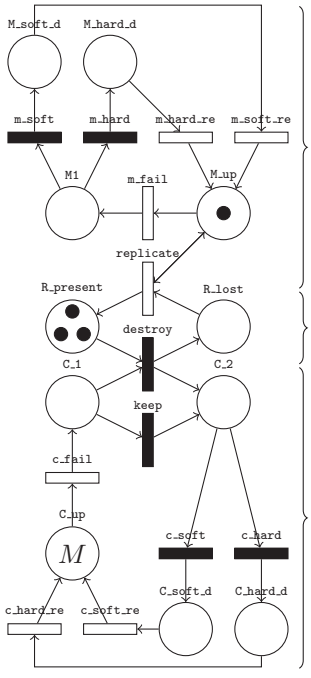
In this section, we demonstrate the application of performability in practice on a case study developed in an earlier publication (Cloth and Haverkort 2005). The model we consider addresses a replicated file system as used as part of the Google search engine (Ghemawat *et al.* 2003).

The high-level description is given as a generalised stochastic Petri net (GSPN) (Ajmone Marsan *et al.* 1984). GSPNs are Petri nets in which transitions are either immediate or stochastic, the latter being decorated with rates. A GSPN can be transformed into an *underlying CTMC*. The states of this CTMC consist of an assignment of the number of tokens to (a subset of) the places.

In the file system model we consider, files are divided into *chunks* of equal size. Several copies of each chunk reside at several *chunk servers*. There is a single *master* server, which knows the location of the chunk copies. If a user of the file system wants to access a certain chunk of a file, it asks the master for the location. Data transfer then takes place directly between a chunk server and the user.

The GSPN describing the model is shown in Figure 4, and is identical to the one given in the original paper (Cloth and Haverkort 2005). Timed transitions are given as white

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press



transition		rate / prob.
`m_fail`		0.0005
`m_soft_re`		#`M_soft_d`·12
`m_hard_re`		6.0
`c_fail`		0.05
`c_soft_re`		#`C_soft_d`·12
`c_hard_re`		1.0
`replicate`	20.0	if #`R_present`>0
	2.0	if #`R_present`=0
`m_soft`		0.95
`m_hard`		0.05
`c_soft`		0.95
`c_hard`		0.05
`destroy`		$\frac{\#\texttt{R_present}}{\#\texttt{C_up}}$
`keep`		$1 - \frac{\#\texttt{R_present}}{\#\texttt{C_up}}$

Condition for `replicate`: $(\#\texttt{C_up} > \#\texttt{R_present}) \wedge (\#\texttt{C_up} \cdot S \geqslant (\#\texttt{R_present}+1) \cdot N)$

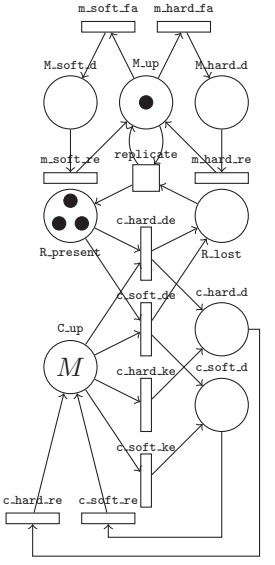
Fig. 4. GSPN of the Google file system as described in (Cloth and Haverkort 2005).

rectangles and immediate transitions as black ones. Conditions and effects are specified as usual for Petri nets by arcs connecting transitions and places. Additional conditions, together with rates and probabilities, are displayed in the right-hand side of the figure.

The GSPN describes the life cycle of a single chunk, but takes into account the load caused by the other chunks. The upper part describes the master. It may be: up and running (token at `M_up`); failed, but the type of failure not yet decided (`M1`); failed because of a software failure (`M_soft_d`); or failed because of a hardware failure (`M_hard_d`). The middle part describes the number of copies of available (`R_present`), as well as the number of lost (`R_lost`) copies of the chunk under consideration. The lower part of the GSPN describes the behaviour of the chunk servers. It contains places denoting the numbers of running servers (`C_up`) and servers with software (`C_soft_d`) and hardware (`C_hard_d`) failures. If a server crashes, it either stores the chunk under consideration, and thus a copy of it is lost (`destroy`), or it only stores chunks that we do not consider explicitly (`keep`), so no copies are lost.

We transformed the GSPN into an equivalent model in which there are no immediate transitions, as shown in Figure 5. GSPNs without immediate transitions can be easily transformed into the PRISM (Kwiatkowska *et al.* 2009) modelling language, which is a stochastic variant of Dijkstra's guarded command language. We used PRISM , partially as a model checker and partially to transform the models (CTMCs in this case) to the (sparse matrix) format of our model checker MRMC (Katoen *et al.* 2011). All experiments were conducted on an Intel Core 2 Duo P9600 with 2.66 GHz clock frequency and 4 GB of main memory running Linux.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press



transition	rate
m_soft_fa	$0.0005 \cdot 0.95$
m_hard_fa	$0.0005 \cdot 0.05$
m_soft_re	#M_soft_d·12
m_hard_re	6.0
c_soft_de	$0.05 \cdot 0.95 \cdot \frac{\#\text{R_present}}{\#\text{C_up}}$
c_hard_de	$0.05 \cdot 0.05 \cdot \frac{\#\text{R_present}}{\#\text{C_up}}$
c_soft_ke	$0.05 \cdot 0.95 \cdot (1 - \frac{\#\text{R_present}}{\#\text{C_up}})$
c_hard_ke	$0.05 \cdot 0.05 \cdot (1 - \frac{\#\text{R_present}}{\#\text{C_up}})$
c_soft_re	#C_soft_d·12
c_hard_re	1.0
replicate	20.0 if #R_present>0
	2.0 if #R_present=0

Condition for replicate: (#C_up>#R_present)∧ (#C_up·S⩾(#R_present+1)·N)

Fig. 5. Transformed GSPN of the Google file system without intermediate transitions. The model is equivalent to the one of Cloth and Haverkort (2005).

Table 2. *Number of states and transitions in the underlying CTMC of the GSPN of Figure 5, depending on the number of chunk servers.*

M	#states $= 6(M^2 + 1)$	#transitions
20	2406	15323
40	9606	63614
60	21606	145335
80	38406	260885
100	60006	410435
120	86406	593985

The model has three parameters:

— M is the number of chunk servers;
— S is the number of chunks a chunk server may store;
— N is the total number of chunks.

We fix $S = 5000$ and $N = 100000$. Table 2 shows the number of states and transitions for different values of M.

We first consider a survivability property expressed in CSL (as described in Section 2.5):

$$\Phi = \mathit{severe_hardware_disaster} \Rightarrow \mathbb{P}_{\geqslant x}(\Diamond^{\leqslant T}\, \mathit{service_level_3}),$$

where

$$\mathit{severe_hardware_disaster} = (\texttt{M_hard_d} = 1) \wedge (\texttt{C_hard_d} > 0.75 \cdot M) \wedge (\texttt{C_soft_d} = 0),$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Table 3. *Performance figures for bounded until property* Φ *in the underlying CTMC of the GSPN under consideration. The table shows the minimal probability ('Prob.') over all states in which severe_hardware_disaster is fulfilled to reach a state in which service_level_3 holds within t time units. In addition, we give the time needed for the computations ('Time') in minutes ('m') and seconds ('s').*

M		$T = 20$	$T = 40$	$T = 60$	$T = 80$	$T = 100$	$T = 120$	$T = 140$
20	Time	0s	0s	0s	0s	0s	0s	0s
	Prob.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
40	Time	0s	1s	1s	1s	1s	2s	2s
	Prob.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
60	Time	7s	12s	19s	25s	31s	36s	43s
	Prob.	0.000000	0.001766	0.507116	0.990604	0.999992	1.000000	1.000000
80	Time	16s	32s	47s	1m 4s	1m 24s	1m 48s	2m 11s
	Prob.	0.000000	0.001767	0.507119	0.990604	0.999992	1.000000	1.000000
100	Time	43s	1m 34s	2m 28s	3m 19s	4m 18s	4m 36s	5m 36s
	Prob.	0.000000	0.001767	0.507119	0.990604	0.999992	1.000000	1.000000
120	Time	1m 30s	2m 54s	4m 20s	5m 44s	7m 15s	8m 13s	9m 20s
	Prob.	0.000000	0.001767	0.507119	0.990604	0.999992	1.000000	1.000000

and

$$service_level_3 = (\texttt{M_up} = 1) \wedge (\texttt{R_present} \geqslant 3).$$

The formula Φ states that in all states in which severe hardware problems have occurred (master server is down and more than three quarters of the chunk servers are down), a state in which a guaranteed quality-of-service level (all three chunk copies are present and the master server is available) holds will be reached within time t with a probability of at least x. We give the performance figures in Table 3 for different time bounds t and different numbers of chunk servers M. We used MRMC for this analysis; we could have used PRISM as well, but we chose MRMC so that we would be able to compare performance measures in a reasonable way with the results of the next set of experiments. We set the algorithms to use a precision of 10^{-6}.[†]

The analyses were performed using a standard algorithm for the time-bounded until operator in CTMCs (Baier *et al.* 2003). Instead of giving the truth value of Φ for the individual states, we give the minimal probability that *service_level_3* is reached within time t from state s, over all s in which *severe_hardware_disaster* holds. Cloth and Haverkort (2005) described a number of similar properties, and the time complexity for all these is the same.

It can be seen that the time needed for the analyses grows approximately linearly with the time bound. The same holds with respect to the model size. For instance, for $M = 100$, the state space is about three times larger than for $M = 60$ (see Table 2), and so is the

[†] Some results reported in Table 3 differ slightly from those reported in Cloth and Haverkort (2005) – this appears to be rooted in numerical instabilities of the prototype implementation used then.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

time needed for the analysis. These observations are in accordance with the complexity results for the algorithm used (Baier *et al.* 2003). As expected, the more time we allow for recovery to the required quality of service, the more likely it is that we recover before the bound is reached.

As seen in the table, for values of M below 60, the reachability probability is always zero. This seemingly strange result is in accordance with the theory. Consider the enabling condition for replicate: '$(\#\texttt{C_up} \cdot S \geqslant (\#\texttt{R_present} + 1) \cdot N)$'. Given that we have $M = 60$, $S = 5000$ and $N = 100000$, if all chunk servers are up, it is $\#\texttt{C_up} \cdot S = 300000$. If there are two existing replicas, it is $(\#\texttt{R_present} + 1) \cdot N) = 300000$. For service level three, we need to obtain three replicas in the end. For $M = 59$ or below, this will never be possible, because then $\#\texttt{C_up} \cdot S < 300000$ and the condition to generate the third replica can never be fulfilled since we can never have enough chunk servers up and running. Because of this, service level three will never be reached.

In the previous analyses, we assumed that we know the probabilities and rates appearing in the model exactly. We now consider a model variant in which we assume that we do not know the probabilities of whether a hardware or a software failure occurs in the chunk server part, but that the other probabilities and rates are known – this case was not addressed in Cloth and Haverkort (2005). We thus have a non-deterministic choice between `c_soft` and `c_hard` (of Figure 4). Accordingly, the underlying model is no longer a Markov chain, but a Markov decision process. Using an algorithm integrated into an experimental version of MRMC (Buchholz *et al.* 2011), we can handle time-bounded until formulae for this model class. We give performance measures in Table 4. The non-determinism can be resolved in different ways, leading to different underlying stochastic models. The probabilities given in the table correspond to the resolution of the non-determinism such that reachability probabilities are either minimal or maximal (over the general class of schedulers). Because the non-deterministic choice abstracts from the concrete probabilities in the purely stochastic model, the probabilities of Table 3 lie between the minimal and maximal probabilities of those computed for Table 4. In Figure 6, we fix $M = 60$ and plot the probabilities in the CTMC (following Table 3), as well as the lower and upper bounds obtained from the CTMDP as a function of the time bound (following Table 4). The probabilities obtained from the CTMC model are close to the upper bound for the CTMDP model. This is because in the CTMC we have a probability of 0.95 for a software failure, which can be repaired much more quickly than a hardware failure. The algorithm for analysing models involving non-determinism is more complex because we have to consider the worst-case probabilities over all possible choices. Because of this, the analyses took much longer, but we were still able to complete all of the analyses we completed for the original model.

We now consider the variant of the model without non-determinism again, and assign a reward of 1 to all transitions corresponding to the failure of a component (`m_soft_fa`, `m_hard_fa`, `c_soft_de`, `c_soft_ke`, `c_hard_de`, `c_hard_ke`). This means that we need to add a transition reward structure to the underlying CTMC. The property

$$\Psi_1 = \mathbb{E}_{<x}(\mathrm{tt})$$

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Table 4. *Performance figures for bounded until property* Φ *in the CTMDP variant of the GSPN under consideration. The table shows the minimal probability ('Prob.') over all states in which severe_hardware_disaster is fulfilled to reach a state in which service_level_3 holds within t time bounds. In addition, we give the time needed for the computations ('Time'), in minutes ('m') and seconds ('s'). In lines marked by 'min' we minimise over the non-deterministic choice whereas for lines marked with 'max' we maximise.*

M			$T = 20$	$T = 40$	$T = 60$	$T = 80$	$T = 100$	$T = 120$	$T = 140$
20	min	Time	4s	8s	14s	18s	23s	29s	34s
		Prob.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	max	Time	5s	11s	17s	23s	29s	36s	43s
		Prob.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
40	min	Time	22s	45s	1m 9s	1m 35s	2m 2s	2m 27s	2m 53s
		Prob.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	max	Time	31s	1m 4s	1m 37s	2m 12s	2m 46s	3m 22s	4m 0s
		Prob.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
60	min	Time	1m 39s	3m 15s	4m 55s	6m 34s	8m 7s	9m 48s	11m 10s
		Prob.	0.000000	0.000899	0.372063	0.968060	0.999877	1.000000	1.000000
	max	Time	1m 39s	3m 18s	4m 59s	6m 38s	8m 8s	9m 37s	11m 9s
		Prob.	0.000000	0.001830	0.514567	0.991275	0.999994	1.000000	1.000000
80	min	Time	4m 10s	8m 6s	12m 14s	16m 30s	20m 5s	23m 29s	25m 24s
		Prob.	0.000000	0.000899	0.372069	0.968061	0.999877	1.000000	1.000000
	max	Time	4m 21s	8m 23s	12m 42s	16m 46s	19m 52s	23m 29s	25m 26s
		Prob.	0.000000	0.001830	0.514569	0.991275	0.999994	1.000000	1.000000
100	min	Time	8m 4s	16m 2s	24m 3s	32m 7s	39m 9s	45m 31s	50m 35s
		Prob.	0.000000	0.000899	0.372069	0.968061	0.999877	1.000000	1.000000
	max	Time	8m 18s	16m 22s	24m 26s	32m 12s	37m 56s	45m 15s	50m 48s
		Prob.	0.000000	0.001830	0.514569	0.991275	0.999994	1.000000	1.000000
120	min	Time	15m 15s	29m 44s	43m 40s	59m 18s	67m 7s	76m 56s	84m 48s
		Prob.	0.000000	0.000899	0.372069	0.968061	0.999877	1.000000	1.000000
	max	Time	14m 59s	30m 20s	43m 39s	58m 54s	66m 42s	76m 22s	83m 25s
		Prob.	0.000000	0.001830	0.514569	0.991275	0.999994	1.000000	1.000000

corresponds to the average number of failures per time unit, while

$$\Psi_2 = \mathbf{C}_{<x}^{[0,T]}(\mathrm{tt})$$

describes the total number of failures until time t.

As MRMC does not yet support properties of the form of Ψ_1 and Ψ_2 (but instead focuses on the reward-bounded until), we used PRISM to carry out these analyses. We

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

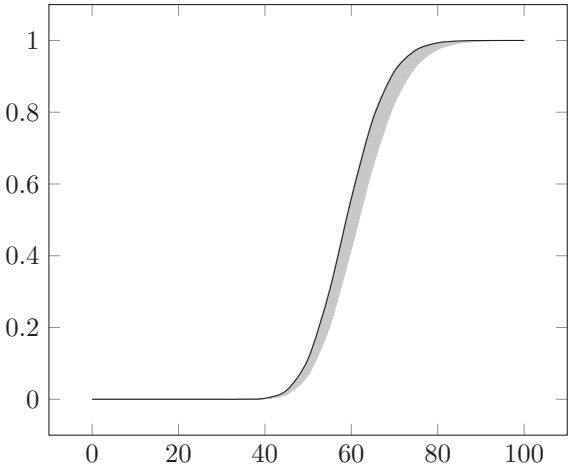


Fig. 6. Reachability probabilities for $M = 60$ as a function of the time bound. The black plot corresponds to the numbers obtained from the CTMC model (see Table 3), while the enclosed grey area is given by lower and upper bounds in the CTMDP model (see Table 4.)

Table 5. *Performance figures for reward-based property* Ψ_1. *The table shows the long-run average number of failures per time unit ('Reward') as well as the time needed to perform the analysis ('Time') in minutes ('m') and seconds ('s').*

M		
20	Time	0s
	Reward	0.050500
40	Time	1s
	Reward	0.050500
60	Time	1s
	Reward	0.050500
80	Time	2s
	Reward	0.050500
100	Time	3s
	Reward	0.050500
120	Time	6s
	Reward	0.050500
140	Time	9s
	Reward	0.050500

used the PRISM engine based on sparse matrices and a precision of 10^{-6}. In Table 5, we give results for Ψ_1, and in Table 6 we provide those for Ψ_2. Instead of the truth values, we provide the rewards obtained when starting in the initial state of the model. There is no visible influence of M on the average number of failures per time unit. In theory, the choice of M should, however, affect this number: a failure can only occur if the place `C_up` is not empty, and this is more likely for large M. However, for the analyses we

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Table 6. *Performance figures for reward-based property* Ψ_2. *The table shows the average number of failures that have occurred up until time t ('Reward'), as well as the time needed for the analysis ('Time') in minutes ('m') and seconds ('s').*

M		$T = 20$	$T = 40$	$T = 60$	$T = 80$	$T = 100$	$T = 120$	$T = 140$
20	Time	0s	0s	0s	0s	0s	0s	0s
	Reward	1.010000	2.019999	3.029999	4.039998	5.049998	6.059997	7.069997
40	Time	1s	1s	1s	1s	1s	1s	1s
	Reward	1.010000	2.019999	3.029999	4.039998	5.049998	6.059997	7.069997
60	Time	3s	3s	3s	3s	3s	3s	3s
	Reward	1.010000	2.019999	3.029999	4.039998	5.049998	6.059997	7.069997
80	Time	6s	6s	6s	6s	6s	6s	6s
	Reward	1.010000	2.019999	3.029999	4.039998	5.049998	6.059997	7.069997
100	Time	11s	11s	11s	18s	18s	18s	16s
	Reward	1.010000	2.019999	3.029999	4.039998	5.049998	6.059997	7.069997
120	Time	18s	18s	18s	19s	23s	31s	31s
	Reward	1.010000	2.019999	3.029999	4.039998	5.049998	6.059997	7.069997

have carried out, the effect of the choice of M is below the analysis precision. This can be explained by the fact that repair rates are much higher than the failure rates, so it is already unlikely that `C_up` is empty for the smaller M considered.

The time needed for the analysis is again approximately linear in the number of states. However, it is much lower than for the time-bounded reachability analyses carried out for property Φ (we also checked this for PRISM). In addition, for Ψ_2, the dependence on the time bound t is much weaker than for the computations of the reachability probability. While in the worst case the algorithm used is still linear in the time bound, it features what is known as a 'steady-state detection' mechanism. For this model, this allows us to terminate the iterative algorithm at an early stage, while still guaranteeing the precision requested.

9. Epilogue

In this paper we have shown that model checking and performability analysis, combined with logics and performability specifications, form 'dream teams'. We have illustrated this through a detailed treatment of various core performability measures on an important model in performabilty analysis – continuous-time Markov reward models. The flexibility provided by using stochastic temporal logics like CSRL as a specification vehicle means we can give succinct representations of many standard, and new, performability measures that have practical relevance. In addition, model checking provides a *unified algorithmic approach* for analysing a broad variety of performability measures. That is to say, a single algorithm suffices to treat all measures that can be expressed in the logical framework provided. There is no need to come up with a new algorithm for a new formula, that is, a new measure. It is our firm belief that this is one of the major strengths of the approach advocated in this paper.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Apart from giving an overview of the key ingredients in a model-checking-based performability evaluation, we have also provided a few new results. We extended an existing duality result for (constrained) reachability properties by showing the duality of long-run averages and expected long-run rewards. We have also defined notions of bisimulation for MRMs with impulse rewards, illustrated the potential impact of considering bisimulation quotients on our running example and considered the notion of expected reward measures in the presence of state and impulse rewards. Finally, a comprehensive case study (the Google file system) demonstrates the power of the currently available model-checking technology. In particular, we demonstrated the analysis in the presence of non-determinism, that is, we presented some experimental results on model checking continuous-time Markov reward *decision* processes.

As future work, it is important to improve the efficiency of some of the verification algorithms: in particular, for time- and reward-bounded reachability probabilities. Given the close intertwining of the elapse of time and reward, this is a challenge. Furthermore, it is fair to say that the analysis of continuous-time Markov reward decision processes is in its infancy, and much progress is required there. On the modelling side, reward extensions of stochastic hybrid systems seem to be of interest. Finally, we believe that in order to increase scalability, we will need aggressive abstraction techniques that go far beyond the state space reductions that can be obtained by bisimulation quotienting.

Appendix A. Table of symbols

The following table gives an overview of symbols used in the paper.

Symbol	Meaning	Page
$\Pr(ev)$	probability of event ev	756
$Path(s)$	set of maximal paths starting in state s	760
$\Pr$	measure on paths	760
$\Pr_s$	probability measure on paths starting in state s	761
t	used for time duration and time points	756
$\mathscr{C}$	used for continuous-time Markov chains (CTMCs)	756
$\mathscr{C}$	used for continuous-time Markov decision processes (CTMDPs)	778
$\mathscr{M}$	used for Markov reward models (MRM)	758
λ	used for rates	756
X	random variable	756
$X(t)$	state of a Markov model at time t	766
$Y(t)$	accumulated reward	766
$W(t)$	time-averaged accumulated reward	766
$A(t)$	availability at time t	766
$F_X(t)$	cumulative probability distribution of X	756
S	state space of a stochastic model	756

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Symbol	Meaning	Page
s, u	used for states	756
$\mathbf{R}(s, s')$	rate from state s to state s'	756
AP	set of atomic propositions	756
a	used for atomic propositions	756
$L(s)$	labelling of state s	756
$E(s)$	sum of outgoing rates of state s	756
$\mathbf{P}(s, s')$	probability to finally jump from state s to state s'	757
$\mathbf{P}(s, S')$	probability to finally jump from state s to set S' of states	757
$s \to s'$	transition from state s to state s'	757
$\iota(s, s')$	impulse reward from state s to state s'	758
$\rho(s)$	reward rate of state s	758
$\rho,\ \widehat{\rho},\ \tilde{\rho}$	reward structures	758
σ	used for paths	760
s_i	used for $(i+1)$th state of a path	760
t_i	used for $(i+1)$th time duration of a path	760
$\sigma[i]$	$(i+1)$th state of path σ	760
$\sigma@t$	state occupied at time t on path σ	760
$\pi(s, s', t)$	probability of being in state s' at time t if started in state s	760
$\pi(s, s')$	steady-state probability of being in state s' if started in state s	761
$\pi(s')$	steady-state probability of being in state s' regardless of starting state	761
$\pi(s, S', t)$	probability of being in a set S' of states at time t if started in state s	761
$\pi(s, S')$	steady-state probability of being in a set S' of states if started in state s	761
$y(\sigma, t)$	cumulative reward on path σ up to time t	760
$\rho(s, s')$	expected long-run reward rate for state s' if started in state s	761
$\rho(s, s', t)$	expected instantaneous reward rate for state s' at time t if started in state s	761
$\rho(s, S')$	expected long-run reward rate for a set S' of states if started in state s	761
$\rho(s, S', t)$	expected instantaneous reward rate for a set S' of states at time t if started in state s	761
$\mathrm{EY}(s, s', t)$	expected accumulated reward in state s' at time t if started in state s	761
$\mathrm{EY}(s, S', t)$	expected accumulated reward at t in set S' of states if started in state s	762

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Symbol	Meaning	Page
$\mathrm{EY}(s, S', I)$	expected reward accumulated in state s' in time interval I if started in state s	765
F	marks a failure state	766
Φ	used for CSRL state formulae	764
Ψ	used for CSRL state formulae	763
$s \models \Phi$	state s fulfills CSRL formula Φ	763
φ	used for CSRL path formulae	763
$\mathsf{Sat}^{\mathscr{M}}(\Phi)$	states fulfilling Φ	762
I	used for time intervals	763
J	used for reward intervals	763
K	used for probability intervals	763
tt	state formula: the boolean value 'true'	764
$\wedge$	state formula: logical 'and'	764
$\neg$	state formula: negation	764
$\mathbb{L}_K(\Phi)$	state formula: long-run probability	764
$\mathbb{E}_J(\Phi)$	state formula: long-run average reward	764
$\mathbb{E}^t_J(\Phi)$	state formula: instantaneous reward	764
$\mathbb{C}^I_J(\Phi)$	state formula: accumulated reward	764
$\mathbb{P}_K(\varphi)$	state formula: probability measure	764
$X^I_J\,\Phi$	path formula: next	764
$\Phi\,\mathscr{U}^I_J\,\Psi$	path formula: until	764
$\Diamond^I_J$	state formula: finally	764
$\Box^I_J$	state formula: always	764
$\cdot^*$	dual model, reward structure, and so on	767
q	defined in part about duality	769
$\chi(u)$	defined in part about duality	769
G	used for states fulfilling a certain formula	773
$\mathbf{1}_U$	characteristic function of the set U of states	772
T	used for strongly connected components (SCCs)	772
g_t	initial gain vector	780
$x_{s,T}$	probability of reaching SCC T from state s	773
$\overline{G}$	complement set of states of G	773
$[s]$	equivalence class containing state s	776
$\mathscr{R}$	a bisimulation relation	775
$\sim$	coarsest bisimulation relation	775
$\mathscr{M}/\!\sim$	bisimulation quotient of model $\mathscr{M}$	776
Act	actions of a CTMDP	778
$Act(s)$	actions enabled in state s	778
$\mathbf{R}(s, \alpha, s')$	rate to state s' from state s choosing action α	778

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

References

Abate, A., Katoen, J.-P. and Mereacre, A. (2011) Quantitative automata model checking of autonomous stochastic hybrid systems. In: *Hybrid Systems: Computation and Control (HSCC)*, ACM 83–92.

Abate, A., Prandini, M., Lygeros, J. and Sastry, S. (2008) Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica* **44** (11) 2724–2734.

Ajmone Marsan, M., Conte, G. and Balbo, G. (1984) A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems* **2** 93–122.

Aldini, A. and Bernardo, M. (2007) Mixing logics and rewards for the component-oriented specification of performance measures. *Theoretical Computer Science* **382** 3–23.

Arnold, T. (1973) The concept of coverage and its effect on the reliability model of a repairable system. *IEEE Transactions on Computers* **22** 251–254.

Avizienis, A., Laprie, J.-C., Randell, B. and Landwehr, C. E. (2004) Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* **1** (1) 11–33.

Aziz, A., Sanwal, K., Singhal, V. and Brayton, R. K. (2000) Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic* **1** (1) 162–170.

Baier, C., Cloth, L., Haverkort, B. R., Hermanns, H. and Katoen, J.-P. (2010a) Performability assessment by model checking of Markov reward models. *Formal Methods in System Design* **36** (1) 1–36.

Baier, C., Haverkort, B. R., Hermanns, H. and Katoen, J.-P. (2000) On the logical characterisation of performability properties. In: International Colloquium on Automata, Languages, and Programming (ICALP). *Springer-Verlag Lecture Notes in Computer Science* **1853** 780–792.

Baier, C., Haverkort, B. R., Hermanns, H. and Katoen, J.-P. (2003) Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering* **29** 524–541.

Baier, C., Haverkort, B. R., Hermanns, H. and Katoen, J.-P. (2008) Reachability in continuous-time Markov reward decision processes. In: Flum, J., Grädel, E. and Wilke, T. (eds.) *Logic and Automata*, Texts in Logic and Games **2**, Amsterdam University Press 53–72.

Baier, C., Haverkort, B. R., Hermanns, H. and Katoen, J.-P. (2010b) Performance evaluation and model checking join forces. *Communications of the ACM* **53** 76–85.

Baier, C. and Hermanns, H. (1997) Weak bisimulation for fully probabilistic processes. In: Computer Aided Verification (CAV). *Springer-Verlag Lecture Notes in Computer Science* **1254** 119–130.

Baier, C., Hermanns, H., Katoen, J.-P. and Haverkort, B. R. (2005a) Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theoretical Computer Science* **345** (1) 2–26.

Baier, C. and Katoen, J.-P. (2008) *Principles of Model Checking*, MIT Press.

Baier, C., Katoen, J.-P. and Hermanns, H. (1999) Approximate symbolic model checking of continuous-time Markov chains. In: Concurrency Theory (CONCUR). *Springer-Verlag Lecture Notes in Computer Science* **1664** 146–161.

Baier, C., Katoen, J.-P., Hermanns, H. and Wolf, V. (2005b) Comparative branching-time semantics for Markov chains. *Information and Computation* **200** (2) 149–214.

Beaudry, M. (1978) Performance-related reliability measures for computing systems. *IEEE Transactions on Computer Systems* **27** (6) 540–547.

Berendsen, J., Jansen, D. N. and Katoen, J.-P. (2006) Probably on time and within budget: On reachability in priced probabilistic timed automata. In: *Quantitative Evaluation of Systems (QEST)*, IEEE Computer Society Press 311–322.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Bernardo, M. and Bravetti, M. (2001) Reward based congruences: Can we aggregate more? In: Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM-PROBMIV). *Springer-Verlag Lecture Notes in Computer Science* **2165** 136–151.

Brázdil, T., Forejt, V., Krčál, J., Křetínský, J., and Kučera, A. (2009) Continuous-time stochastic games with time-bounded reachability. In: FSTTCS. *LIPIcs* **4**, Schloss Dagstuhl 61–72.

Buchholz, P., Hahn, E. M., Hermanns, H. and Zhang, L. (2011) Model checking algorithms for CTMDPs. In: CAV. *Springer-Verlag Lecture Notes in Computer Science* **6806** 225–242.

Caillaud, B., Delahaye, B., Larsen, K., Legay, A., Pedersen, M. and Wasowski, A. (2010) Compositional design methodology with constraint Markov chains. In: *Quantitative Evaluation of Systems (QEST)*, IEEE Computer Society Press 123–132.

Chiola, G. (1985) A software package of the analysis of generalized stochastic Petri nets. In: *Proceedings of the 1st International Workshop on Timed Petri Nets*, IEEE Computer Society Press 136–143.

Ciardo, G., Jones III, R. L., Miner, A. S. and Siminiceanu, R. (2003) Logical and stochastic modeling with SMART. In: Computer Performance Evaluation / TOOLS. *Springer-Verlag Lecture Notes in Computer Science* **2794** 78–97.

Ciardo, G., Miner, A. S., Wan, M. and Yu, A. J. (2007) Approximating stationary measures of structured continuous-time Markov models using matrix diagrams. *ACM SIGMETRICS Performance Evaluation Review* **35** (3) 16–18.

Clarke, E. M., Emerson, E. A. and Sistla, A. P. (1986) Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems* **8** 244–263.

Cloth, L. and Haverkort, B. R. (2005) Model checking for survivability! In: *Quantitative Evaluation of Systems (QEST)*, IEEE Computer Society 145–154.

Cloth, L., Jongerden, M. R. and Haverkort, B. R. (2007) Computing battery lifetime distributions. In: *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE Computer Society 780–789.

Desharnais, J. and Panangaden, P. (2003) Continuous stochastic logic characterizes bisimulation of continuous-time Markov processes. *Journal of Logic and Algebraic Programming* **56** (1-2) 99–115.

Donatiello, L. and Iyer, B. (1987a) Analysis of a composite performance reliability measure for fault-tolerant systems. *Journal of the ACM* **34** (1) 179–199.

Donatiello, L. and Iyer, B. (1987b) Closed-form solution for system availability. *IEEE Transactions on Reliability* **36** (1) 45–47.

Furchtgott, D. and Meyer, J. F. (1984) A performability solution method for degradable non-repairable systems. *IEEE Transactions on Computers* **33** (6) 550–554.

Gay, F. A. and Ketelsen, M. L. (1979) Performance evaluation for gracefully degrading systems. In: *Proceedings of the 9th Fault-Tolerant Computer Systems Systems (FTCS)*, IEEE Computer Society Press 51–58.

Geist, R. and Trivedi, K. S. (1990) Reliability estimation of fault-tolerant systems: Tools and techniques. *Computer* **23** (7) 52–61.

Ghemawat, S., Gobioff, H. and Leung, S.-T. (2003) The Google file system. In: *ACM Symposium on Operating Systems Principles (SOSP)*, ACM 29–43.

Goyal, A., Lavenberg, S. and Trivedi, K. S. (1987) The system availability estimator. *Annals of Operations Research* **8** 285–306.

Goyal, A. and Tantawi, A. (1988) A measure of guaranteed availability and its numerical evaluation. *IEEE Transactions on Computers* **37** (1) 25–32.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Goyal, A. and Tantawi, A. N. (1987) Evaluation of performability for degradable computer systems. *IEEE Transactions on Computers* **36** (6) 7.

Grassi, V., Donatiello, L. and Iazeolla, G. (1988) Performability evaluation of multicomponent fault-tolerant systems. *IEEE Transactions on Reliability* **37** (2) 2.

Gross, D. and Miller, D. (1984) The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research* **32** (2) 926–944.

Haverkort, B. R. (2006) Can we quantitatively assess security? In: Workshop on Emperical Evaluation of Dependability and Security; Supplement Proceedings DSN 2006 (online) 125–128.

Haverkort, B. R. and Niemegeers, I. G. (1996) Performability modelling tools and techniques. *Performance Evaluation* **25** 17–40.

Haverkort, B. R., Trivedi, K. S., Rubino, G. and Marie, R. (eds.) (2001) *Performability Modelling: Techniques and Tools*, John Wiley and Sons.

Hermanns, H. (2002) Interactive Markov Chains: The Quest for Quantitative Quality. *Springer-Verlag Lecture Notes in Computer Science* **2428**.

Hermanns, H. and Katoen, J.-P. (2009) The how and why of interactive Markov chains. In: 8th International Symposium on Formal Methods for Components and Objects (FMCO). *Springer-Verlag Lecture Notes in Computer Science* **6286** 311–337.

Hermanns, H., Kwiatkowska, M. Z., Norman, G., Parker, D. and Siegle, M. (2003) On the use of MTBDDs for performability analysis and verification of stochastic systems. *Journal of Logic and Algebraic Programming* **56** (1-2) 23–67.

Howard, R. A. (1971a) *Dynamic Probabilistic Systems; Volume I: Markov models*, John Wiley and Sons.

Howard, R. A. (1971b) *Dynamic Probabilistic Systems; Volume II: Semi-Markov and decision processes*, John Wiley and Sons.

Jensen, A. (1953) Markov chains as an aid in the study of Markov processes. *Skandinavisk Aktuarietidsskrift* **36** 87–91.

Jonsson, B. and Larsen, K. (1991) Specification and refinement of probabilistic processes. In: *6th IEEE Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society Press 266–277.

Katoen, J.-P., Klink, D., Leucker, M. and Wolf, V. (2007) Three-valued abstraction for continuous-time Markov chains. In: Damm, W. and Hermanns, H. (eds.) Computer Aided Verification 19th International Conference, CAV 2007. *Springer-Verlag Lecture Notes in Computer Science* **4590** 311–324.

Katoen, J.-P., Zapreev, I. S., Hahn, E. M., Hermanns, H. and Jansen, D. N. (2011) The ins and outs of the probabilistic model checker MRMC. *Performance Evaluation* **2** (68) 90–104.

Kattenbelt, M., Kwiatkowska, M., Norman, G. and Parker, D. (2009) Abstraction refinement for probabilistic software. In: Jones, N. and Muller-Olm, M. (eds.) Verification, Model Checking, and Abstract Interpretation (VMCAI). *Springer-Verlag Lecture Notes in Computer Science* **5403** 182–197.

Kleinrock, L. (1975) *Queueing Systems; Volume 1: Theory*, John Wiley and Sons.

Kleinrock, L. (1976) *Queueing Systems; Volume 2: Computer Applications*, John Wiley and Sons.

Kwiatkowska, M., Norman, G. and Parker, D. (2009) PRISM: Probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review* **36** (4) 40–45.

Larsen, K. and Skou, A. (1991) Bisimulation through probabilistic testing. *Information and Computation* **94** (1) 1–28.

Larsen, K. G. and Rasmussen, J. I. (2008) Optimal reachability for multi-priced timed automata. *Theoretical Computer Science* **390** (2-3) 197–213.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Meyer, J. F. (1976) Computation-based reliability analysis. *IEEE Transactions on Computers* **25** (6) 578–584.

Meyer, J. F. (1980) On evaluating the performability of degradable computing systems. *IEEE Transactions on Computers* **29** (8) 720–731.

Meyer, J. F. (1982) Closed-form solutions of performability. *IEEE Transactions on Computers* **31** (7) 648–657.

Meyer, J. F. (1992) Performability: A retrospective and some pointers to the future. *Performance Evaluation* **14** (3) 139–156.

Meyer, J. F. (1995) Performability evaluation: Where it is and what lies ahead. In: *Proceedings of the 1st International Performance and Dependability Symposium*, IEEE Computer Society Press 334–343.

Meyer, J. F. and Sanders, W. H. (2001) Specification and construction of performability models. In: *Performability Modelling*, John Wiley and Sons.

Miller, B. L. (1968) Finite state continuous time Markov decision processes with a finite planning horizon. *SIAM Journal on Control* **6** (2) 266–280.

Milner, R. (1971) An algebraic definition of simulation between programs. In: *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, William Kaufmann 481–489.

Milner, R. (1980) A Calculus for Communicating Processes. *Springer-Verlag Lecture Notes in Computer Science* **92**.

Molloy, M. (1982) Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers* **31** (9) 913–917.

Movaghar, A. and Meyer, J. F. (1984) Performability modelling with stochastic activity networks. In: *IEEE Real-Time Systems Symposium*, IEEE Computer Society Press 215–224.

Neuhäußer, M. R. and Zhang, L. (2010) Time-bounded reachability probabilities in continuous-time Markov decision processes. In: *Quantitative Evaluation of Systems (QEST)*, IEEE Computer Society Press 209–218.

Pattipati, K., Li, Y. and Blom, H. (1993) A unified framework for the performability evaluation of fault-tolerant computer systems. *IEEE Transactions on Computers* **42** (3) 312–326.

Qureshi, M. A. and Sanders, W. H. (1994a) Reward model solution methods with impulse and rate rewards: An algorithm and numerical results. *Performance Evaluation* **20** 413–436.

Qureshi, M. A. and Sanders, W. H. (1994b) Reward model solution methods with impulse and rate rewards: an algorithm and numerical results. *Performance Evaluation* **20** (4) 413–436.

Rabe, M. and Schewe, S. (2011) Finite optimal control for time-bounded reachability in continuous-time Markov games and CTMDPs. *Acta Informatica* **48** (5-6) 291–315.

Reibman, A. and Trivedi, K. S. (1988) Numerical transient analysis of Markov models. *Computers and Operations Research* **15** (1) 19–36.

Reibman, A. and Trivedi, K. S. (1989) Transient analysis of cumulative measures of Markov model behavior. *Stochastic Models* **5** (4) 683–710.

Sanders, W. and Meyer, J. F. (1987) Performability evaluation of distributed systems using stochastic activity networks. In: *Proceedings of the 2nd International Workshop on Petri Nets and Performance Models*, IEEE Computer Society Press 111–120.

Sanders, W. and Meyer, J. F. (1991) Reduced-base model construction for stochastic activity networks. *IEEE Journal on Selected Areas in Communications* **9** (1) 25–36.

Sanders, W. H. (2010) Quantitative evaluation of security metrics. In: *Quantitative Evaluation of Systems (QEST)*, IEEE Computer Society Press 306.

Segala, R. and Lynch, N. (1995) Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing* **2** (2) 250–273.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press

Sericola, B. (2000) Occupation times in Markov processes. *Communications in Statistics – Stochastic Models* **16** (5) 479–510.

Smith, R., Trivedi, K. S. and Ramesh, A. (1988) Performability analysis: measures, an algorithm and a case study. *IEEE Transactions on Computers* **37** (4) 406–417.

Souza e Silva, E. and Gail, H. (1992) Performability analysis of computer systems: from model specification to solution. *Performance Evaluation* **1** 157–196.

Tijms, H. and Veldman, R. (2000) A fast algorithm for the transient reward distribution in continuous-time Markov chains. *Operation Research Letters* **26** 155–158.

van Dijk, N. M., Haverkort, B. R. and Niemegeers, I. G. (1992) Guest editorial: Performability modelling of computer and communication systems. *Performance Evaluation* **14** 135–138.

Wachter, B., Zhang, L. and Hermanns, H. (2007) Probabilistic model checking modulo theories. In: *Quantitative Evaluation of Systems (QEST)*, IEEE Computer Society Press.

Zhang, L. and Neuhäußer, M. (2010) Model checking interactive Markov chains. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS). *Springer-Verlag Lecture Notes in Computer Science* **6015** 53–68.

https://doi.org/10.1017/S0960129512000254 Published online by Cambridge University Press