Model Checking Performability Properties*

Boudewijn Haverkort, **Lucia Cloth**
Department of Computer Science, RWTH Aachen, D-52056 Aachen, Germany

Holger Hermanns, **Joost-Pieter Katoen**
Faculty of Computer Science, University of Twente, 7500 AE Enschede, the Netherlands

Christel Baier
Department of Computer Science, University of Bonn, D-53117 Bonn, Germany

Abstract

Model checking has been introduced as an automated technique to verify whether functional properties, expressed in a formal logic like computational tree logic (CTL), do hold in a formally-specified system.

In recent years, we have extended CTL such that it allows for the specification of properties over finite-state continuous-time Markov chains (CTMCs). Computational techniques for model checking have been developed and successfully applied in the dependability context. Further work in this area has recently led to the continuous stochastic reward logic (CSRL), a logic to specify measures over CTMCs extended with a reward structure (so-called Markov reward models). Well-known performability measures, most notably also Meyer's performability distribution, can be easily defined with CSRL. However, using CSRL it is possible to specify performability measures that have not yet been addressed in the literature, hence, for which no computational procedures have been developed yet.

In this paper we present a number of computational procedures to perform model checking of CSRL over finite Markov reward models, thereby stressing their computational complexity (time and space) and applicability from a practical point of view (accuracy, stability). A case study in the area of ad hoc mobile computing under power constraints shows the merits of CSRL and the new computational procedures.

Keywords: dependability evaluation, performability evaluation, measure specification, model checking, formal verification, uniformisation, ad hoc mobile computing.

*This work is performed in the context of the VOSS project ("Validation of Stochastic Systems") which is financially supported in Germany by the DFG (for the Universities of Bonn, Erlangen and the RWTH Aachen) and in the Netherlands by NWO (Universities of Nijmegen and Twente).

1. Introduction

Model checking is an automatic technique to verify whether certain properties, expressed in a formal logic like computational tree logic (CTL; see [7]), do hold in a model, typically expressed as a transition system. Originally, model checking procedures have been devised to verify functional properties, for instance, to verify reachability or to verify whether certain paths (state sequences) can occur in a given finite-state machine [7]. For an overview of the techniques and benefits of model checking see e.g. [8].

Recently, instead of using a (timeless) finite-state transition system as model, the use of CTMCs as models has been proposed [1, 2]. In combination with a logic that also allows for the specification of timed-properties (the logic CSL, for continuous stochastic logic), one can formally express steady-state and transient measures over CTMCs in a very flexible way. Moreover, CSL allows one to specify probabilistic measures over paths through CTMCs. As an example, it can be expressed what the probability is, that starting from a particular state, within t time units another state is reached, thereby avoiding or deliberately visiting particular intermediate states. This is a very powerful feature in the context of dependability evaluation, as we have demonstrated with a large case study [14]. The paper [2] formally specifies the logic CSL (syntax and semantics), whereas [3] presents efficient numerical procedures for model checking CSL over CTMCs.

To further strengthen the applicability of the stochastic model checking approach we recently considered Markov models involving costs or rewards. We extended the logic CSL to the *continuous stochastic reward logic* CSRL in order to specify steady-state, transient and path-based measures over CTMCs extended with a reward structure (Markov reward models) [4]. We showed that well-known performability measures, most notably also *the performability distribution* introduced by Meyer [18, 19, 20], can

Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)
0-7695-1597-5/02 $17.00 © 2002 IEEE

be specified using CSRL. However, CSRL allows for the specification of new measures that have not yet been addressed in the performability literature. For instance, when rewards are interpreted as costs, we can express the probability that, given a starting state, a certain goal state is reached within t time units, thereby deliberately avoiding or visiting certain immediate states, and with a total cost (accumulated reward) below a certain threshold. Such a measure has not been considered in the literature so far; [4] did not address computational procedures for full CSRL.

The only other work we are aware of that allows for the specification of path-based measures, has been reported by Obal and Sanders [21, 22]. Roughly speaking, they allow one to analyse more detailed path-based behaviour than we do, because they employ automata of quite a general shape to collect steady-state rewards. On the other hand, their approach neither supports nesting of path- and state properties, nor time- or reward- interval bounds. In any case, the type of measures that we aim at with the logic CSRL, in which bounds on both time and accumulated reward are taken into account independently, has not been addressed in the literature before.

The aim of the current paper is to present a number of computational procedures for the computation of new, CSRL-specified performability measures. The paper is further organised as follows. In Section 2, we introduce Markov reward models and the logic CSRL. In Section 3 we present the model checking approach and discuss the algorithmic problems when facing time as well as reward bounds. In Section 4, a number of computational procedures is presented. They are applied to a case study in Section 5 . Section 6 concludes the paper.

2. Markov reward models and CSRL

This section introduces Markov reward models (MRMs), the continuous stochastic reward logic (CSRL) and describes the features of this logic to specify performability measures over MRMs.

2.1. Markov reward models

Since MRMs play a central role in our approach, we briefly recapitulate their basic concepts and introduce some notation. An MRM is a tuple $\mathcal{M} = (S, \mathbf{R}, \rho)$ where S is a finite set of *states*, $\mathbf{R} : S \times S \to \mathbb{R}_{\geqslant 0}$ is the *rate matrix*, and $\rho : S \to \mathbb{R}_{\geqslant 0}$ is a *reward structure* that assigns to each state s a reward $\rho(s)$, also called gain or bonus, or dually, cost. The MRM has a fixed initial distribution α satisfying $\sum_{s \in S} \alpha_s = 1$, so the MRM starts in state s with probability α_s.

Intuitively, $\mathbf{R}(s, s')$ specifies that the probability of moving from state s to s' within t time-units (for positive t) is $1 - e^{-\mathbf{R}(s,s') \cdot t}$. The reward structure ρ imposes state-based rewards to the model; if t time-units are spent in state s, a reward of $\rho(s) \cdot t$ is earned. For the sake of simplicity, we do not consider impulse rewards here. There is no technical objection against including the latter into the logic setting, but the algorithms we develop in this paper are tailored to state-based rewards only. Let $\mathbf{E}(s) = \sum_{s' \in S} \mathbf{R}(s, s')$ be the total rate at which any transition emanating from state s is taken[1]. More precisely, $\mathbf{E}(s)$ specifies that the probability of leaving s within t time-units (for positive t) is $1 - e^{-\mathbf{E}(s) \cdot t}$. State s is called *absorbing* if $\mathbf{R}(s, s') = 0$ for any state s'.

2.2. Syntax of CSRL

CSRL is a specification formalism for performability measures over MRMs. It contains operators that refer to the stationary and transient behaviour of the system under consideration. As this paper concentrates on model-checking procedures for transient performability measures, we omit the steady-state operator (see [2] for the model-checking procedure). To specify performability measures as logical formulas over MRMs, it is assumed that each state is labelled with so-called *atomic propositions*, the most elementary formulas stating properties over states. Atomic propositions identify specific situations the system may be in, such as "acknowledgement pending", "buffer empty", or "variable X is positive". As generic example atomic propositions we use the properties "*red*" and "*green*"; note that in any state one, both or none of these properties may hold. We use a to range over the set of atomic propositions.

CSRL allows one to specify properties over *states* and over *paths*. A path is an alternating sequence $s_0\, t_0\, s_1\, t_1\, \cdots$ where s_i is a state of the MRM and $t_i > 0$ is the sojourn time in state s_i. The accumulated reward for a finite path of length n is now simply $\sum_{i=0}^{n-1} t_i \cdot \rho(i)$. Let I and J be intervals on the real line, p a probability and $\trianglelefteq$ a comparison operator, such as $>$ or $<$. The syntax of CSRL is defined by the following grammar:

$$\begin{array}{ll} \textit{State-formulas:} & \Phi ::= a \mid \neg\, \Phi \mid \Phi \ \vee \ \Phi \mid \mathcal{P}_{\trianglelefteq p}(\varphi) \\ \textit{Path-formulas:} & \varphi ::= X^I_J\, \Phi \mid \Phi\, \mathcal{U}^I_J\, \Phi \end{array}$$

[1] Note that $\mathbf{R}$ and $\mathbf{E}$ just form an alternative representation of the infinitesimal generator matrix $\mathbf{Q}$; more precisely, $\mathbf{Q} = \mathbf{R} - \mathit{diag}(\mathbf{E})$.

Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)
0-7695-1597-5/02 $17.00 © 2002 IEEE

2.3. Semantics of CSRL

We briefly discuss the intuitive meaning of the different types of formulae in CSRL; a formal definition of the semantics is provided in [4]. Atomic proposition a holds in a state if that state is labelled with a. The meaning of negation ($\neg$) and disjunction ($\vee$) is as usual; note that using these operators, other boolean operators such as conjunction ($\wedge$), implication ($\Rightarrow$) and so forth, can be defined. A state-formula $\mathcal{P}_{\trianglelefteq p}(\varphi)$ is valid in state s if the probability measure of the set of paths starting in s and satisfying path formula φ meets the bound $\trianglelefteq p$. The path-operators next X and until $\mathcal{U}$ are equipped with two parameters. Interval I can be considered as a timing constraint whereas J represents a bound for the cumulative reward. A path satisfies the formula $X^I_J\,\Phi$ if its first transition is made to a Φ-state at time point $t \in I$ such that the earned cumulative reward r until time t (in the current state) meets the bounds specified by J, i.e., $r \in J$. Thus, path $s_0\,2.5\,s_1\,4\ldots$ satisfies $X^{[2,4]}_{[0,4]}$ *red* if s_1 is labelled *red* and $2.5{\cdot}\rho(s_0) \leqslant 4$. A path satisfies $\Phi\,\mathcal{U}^I_J\,\Psi$ if *(i)* Φ holds at all states along the path until a state for which Ψ holds is encountered, *(ii)* the Ψ-state is reached at time $t \in I$, and *(iii)* the earned cumulative reward up to time t lies in J. To ease notation, the formula $\texttt{true}\,\mathcal{U}^I_J\,\Psi$ is abbreviated as $\Diamond^I_J\Psi$.

We restrict ourselves in this paper to intervals that start at 0, i.e., I and J are of the form $[0,t]$ and $[0,r]$, respectively. We will write $X^{\leqslant t}_{\leqslant r}$ for $X^{[0,t]}_{[0,r]}$, and do similarly for the until-operator. Note that the restriction to exactly these interval form eases the computational procedures; the required computational procedures for arbitrary intervals have yet to be found. In case of an unbounded interval $I = [0,\infty)$ or $J = [0,\infty)$ we omit the corresponding bound; the corresponding constraint is vacuously fulfilled.

2.4. Some example properties

To illustrate the expressive power of the until-operator we exemplify its use by means of the following properties:

P0. Property $\mathcal{P}_{>p}(\textit{green}\ \mathcal{U}\ \textit{red})$ holds in every state s if the probability measure of the set of paths (starting from s) that reach some *red* state while passing only through *green* states exceeds p. Note that both the time and reward constraint are ignored here, i.e., $I = J = [0,\infty)$.

P1. Property $\mathcal{P}_{>p}(\textit{green}\ \mathcal{U}^{\leqslant t}\ \textit{red})$ refines property **P0**. It holds in every state s if the probability measure of the set of paths (starting from s) that reach some *red* state *before time* t while passing only through *green* states exceeds p. The reward constraint is ignored in this case.

P2. Dually, $\mathcal{P}_{>p}(\textit{green}\ \mathcal{U}_{\leqslant r}\ \textit{red})$ holds in every state s if the probability measure of the set of paths (starting from s) that reach some *red* state *before accumulating reward* r while passing only through *green* states exceeds p. In this case the time constraint is ignored.

P3. Combining time and reward bounds, $\mathcal{P}_{>p}(\textit{green}\ \mathcal{U}^{\leqslant t}_{\leqslant r}\ \textit{red})$ holds in every state s if the probability measure of the set of paths (starting from s) that reach some *red* state before time t *and* before accumulating reward r while passing only through *green* states exceeds p.

Note that the syntax of CSRL allows nesting of state- and path formulas, as in $\mathcal{P}_{<p}(\textit{green}\,\mathcal{U}^{\leqslant t}_{\leqslant r}(\mathcal{P}_{>q}(\Diamond^{\leqslant t'}_{\leqslant r'}\ \textit{red})))$.

3. The model checking procedure

Once we have formally specified the measure-of-interest by the CSRL-formula Φ, and have obtained a model, i.e., an MRM $\mathcal{M}$, of the system under consideration, the crucial *model checking* step is addressing the question which states in S satisfy formula Φ. The basic algorithmic strategy is as for CSL and CTL. In order to check property Φ, the set *Sat*(Φ) of states that *sat*isfy Φ is computed recursively, followed by a check whether the state of interest belongs to this set. The recursive procedure is in fact a bottom-up traversal of the parse tree of the formula Φ under consideration. For atomic propositions (the leaves in the parse tree) this set is directly obtained from the labelling of the states; *Sat*($\Phi \wedge \Psi$) is obtained by computing *Sat*(Φ) and *Sat*(Ψ) recursively, and then intersecting these sets; *Sat*($\neg\Phi$) is obtained by taking the complement of the entire state space with respect to *Sat*(Φ). The algorithm for the temporal operators is more complicated and involves several numerical computations. For instance, for *Sat*($\mathcal{P}_{\trianglelefteq p}(X\,\Phi)$) we first compute the set *Sat*(Φ), then compute for each state the probability to move to one of these states (in one step), and compare this probability with p according to $\trianglelefteq$.

Model-checking until-formulas is even more involved. We discuss this procedure on the basis of the example properties presented before. Properties of the form **P0** are checked on the basis of the procedure in [13], which amounts to recursively computing the sets *Sat*(Φ) and *Sat*(Ψ) followed by solving a linear system of equations (with $\Phi = \{\textit{green}\}$ and $\Psi = \{\textit{red}\}$). The number of equations equals the number of states of the MRM. An efficient scheme for model-checking **P1**-properties has been proposed in [3]. First, the sets *Sat*(Φ) and *Sat*(Ψ)are computed. All states in *Sat*(Ψ) and all states that are neither in *Sat*(Φ) nor *Sat*(Ψ) are made absorbing. A transient analysis (for time instant t) on the resulting Markov chain then

Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)
0-7695-1597-5/02 $17.00 © 2002 IEEE

suffices to decide the validity of the formula: the probability bound p is compared with the probability mass accumulated in $Sat(\Psi)$ at time t. Properties à la **P2** need some additional preprocessing. By swapping the reward bound into a time bound inside Φ (turning each $\mathcal{U}_{\leqslant r}$ into $\mathcal{U}^{\leqslant r}$), we can resort to the procedure for **P1**, provided that the MRM $\mathcal{M}$ under consideration is *a priori* transformed into MRM $\overline{\mathcal{M}}$. Intuitively, a residence of r time-units in state s of $\overline{\mathcal{M}}$ corresponds to earning a reward r in state s of $\mathcal{M}$ (and vice versa). Details of this transformation can be found in [4, Theorem 1], together with a general duality result on which the transformation is based.

Unfortunately, the above transformation does not help for properties of type **P3**. The reason is that the role of time and rewards in $\mathcal{M}$ and $\overline{\mathcal{M}}$ are truly dual, and hence the transformation does not provide us with a simpler – or better studied – algorithmic problem if both time and rewards are measured. As a consequence, a computational procedure for properties of the form **P3** has not been devised so far. In the next section we discuss three different approaches to verify **P3**-type properties, all of which address a more specific problem, namely reward-bounded instant-of-time reachability, expressed by path operator $\Diamond_{\leqslant r}^{[t,t]}$. The following observation states that this is enough to decide properties of the form **P3**.

Theorem 1
Given the CSRL state formulas Φ and Ψ, and MRM $\mathcal{M}$, let $\mathcal{M}'$ be the MRM obtained from $\mathcal{M}$ by making all Ψ-states and all $\neg(\Phi \wedge \Psi)$-states absorbing and assigning reward 0 to these absorbing states. Then, state s in $\mathcal{M}$ satisfies $\mathcal{P}_{\trianglelefteq p}(\Phi\, \mathcal{U}_{\leqslant r}^{\leqslant t}\, \Psi)$ if and only if s in $\mathcal{M}'$ satisfies $\mathcal{P}_{\trianglelefteq p}(\Diamond_{\leqslant r}^{[t,t]}\, \Psi)$.

The intuitive justification for the above theorem is as follows. Once a path reaches a $\neg(\Phi \wedge \Psi)$-state, there is no way in which it can satisfy a $\Phi\, \mathcal{U}\, \Psi$-formula. We can thus safely make these states absorbing, as the rest of the path is not of interest anymore. Moreover, once a path reaches a Ψ-state at time $t' < t$, while not having accumulated more than r reward, it suffices to be trapped in that state until time t provided no reward will be earned anymore, i.e., $\rho(s) = 0$ for Ψ-state s. Note that we can amalgamate all states satisfying Ψ and all states satisfying $\neg(\Phi \wedge \Psi)$, thereby making the MRM considerably smaller.

Theorem 1 allows us to restrict our attention to the computation of reward-bounded instant-of-time reachability when designing algorithms for model-checking time- and reward-bounded until-formulas. The computational procedures in the next section are based on this observation.

4. Computational procedures

4.1. Problem characterisation

According to Theorem 1, the solution of the CSRL model checking problem for property **P3** can proceed via the computation of the reward-bounded instant-of-time reachability probability. We argue that the latter, in turn, can be computed via the transient accumulated reward distribution. To justify this, we consider a two-dimensional stochastic process $((X_t, Y_t), t \geq 0)$ on $S \times \mathbb{R}_{\geq 0}$, as illustrated in Figure 1. Informally speaking, this stochastic process has a discrete component that describes the transition behaviour in the original CTMC (underlying the MRM) combined with a continuous component that describes the accumulated reward gained over time. For $t = 0$ we have $Y_t = 0$, and for $t > 0$ the value of Y_t increases continuously with rate $\rho(X_t)$. Hence, the discrete states of the original CTMC become "columns" of which the height models the accumulated reward. To take into account the reward bound ($\leq r$), we introduce an absorbing barrier in the process whenever Y_t reaches the level r. Actually, we are interested in

$$\Pr\{Y_t \leqslant r, X_t \in S'\},$$

i.e., the probability of being in a certain subset S' of states at time t, having accumulated a reward smaller than r. For our purposes, S' shall be chosen to be the set $Sat(\Psi)$ of states satisfying Ψ and we start the process in state s under consideration.

Theorem 2
Given CSRL state formula Ψ, let MRM $\mathcal{M}'$ be defined as in Theorem 1, with $\alpha_{s_0} = 1$ for some state s_0 in $\mathcal{M}'$. Then s_0 satisfies $\mathcal{P}_{\trianglelefteq p}(\Diamond_{\leqslant r}^{[t,t]}\, \Psi)$ if and only if $\Pr\{Y_t \leqslant r, X_t \in Sat(\Psi)\} \trianglelefteq p$.

Together with Theorem 1 the above theorem allows us to decide the satisfaction of time- and reward-bounded until formulas via numerical recipes for calculating $\Pr\{Y_t \leqslant r, X_t \in S'\}$ on the two dimensional stochastic process (X_t, Y_t). It is worth to remark that similar processes (with mixed discrete-continuous state spaces) also emerge in the analysis of non-Markovian stochastic Petri nets (when using the supplementary variable approach, cf. [10]), Markov-regenerative stochastic Petri nets [5], and in fluid-stochastic Petri nets [16]. We do not address the algorithms presented in these papers here, since they are either not directly applicable, or suffer from yet-unresolved numerical problems (e.g., related to Laplace back-transformations). In future work, we will investigate them in more detail.

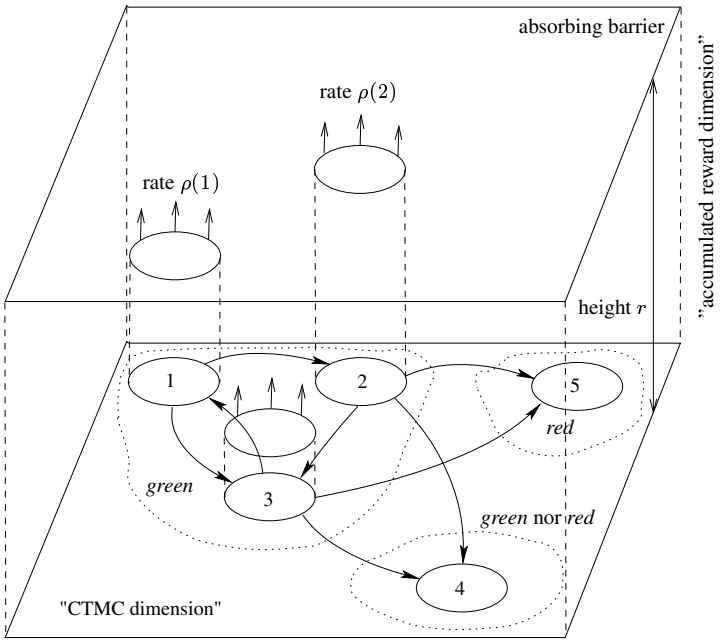

Figure 1. Two-dimensional stochastic process $((X_t, Y_t), t \geq 0)$ for model checking CSRL property P3.

4.2. A pseudo-Erlang approximation

Our first approach to compute $\Pr\{Y_t \leqslant r, X_t \in S'\}$ is to approximate the fixed reward bound r by a reward bound that is Erlang-k distributed with mean r. One may view this as some kind of discretisation of the continuous reward dimension into k steps. The main advantage of this approach is that the resulting model is both discrete-space and completely Markovian, and hence, standard techniques and tools developed for **P2**-like properties can be used to approximate the probabilities required by Theorem 2; reaching the reward bound in the original model corresponds to reaching a particular set of states in the approximated model. As a disadvantage we mention that an appropriate value for k – the number of phases in the Erlangian approximation – is not known *a priori*. Furthermore, when CSRL expressions are nested, it is yet unclear how the error in the approximation propagates. Furthermore, the resulting CTMC becomes substantially larger, especially if k is large. On the other hand, the resulting CTMC can be described in terms of a special tensor structure which can be exploited in the solution procedure (as far as the storage of the generator matrix is concerned).

Further considerations concern the effectiveness of the Erlangian approximation. Since the computation of **P2**-type property-bounds requires the transient analysis of the CTMC under study, one typically employs uniformisation, a generic method to analyse CTMCs via an underlying discrete-time Markov chain [12, 17]. As is well-known, the speed of uniformisation depends on the largest diagonal entry in the generator matrix (in absolute sense). In the context of the suggested Erlang-k approximation where the reward upper bound is r, the maximum diagonal entry is increased (additively) with $\frac{k}{r}\hat{\rho}$, with $\hat{\rho}$ the largest reward rate assigned to any state. This might considerably slow down the uniformisation procedure. For a given error bound $\varepsilon > 0$ one can determine the number of steps N_ε in the uniformised Markov chain needed to reach the given accuracy (see also the discussion in Section 4.4). The theoretical time complexity then becomes $\mathcal{O}(N_\varepsilon \cdot (|S| \cdot k)^2)$. Due to sparseness of the generator matrix the algorithm actually takes less time.

4.3. Discretisation

Recently, Tijms and Veldman [24] proposed a discretisation method for computing the transient distribution of the accumulated reward in an MRM. Their algorithm is a generalisation of an earlier algorithm by Goyal and Tantawi [11] for MRMs with only 0- and 1-rewards. The basic idea is to discretise both the time and the accumulated reward as multiples of the same step size d, where d is chosen such that the probability of more than one transition in the MRM in an interval of length d is negligible.

Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)
0-7695-1597-5/02 $17.00 © 2002 IEEE

The algorithm allows only natural number rewards, but this is no severe restriction since rational rewards can be scaled to yield natural numbers.

Let $F^n(s,k)$ be the function that discretises the joint probability density of being in state s at time $n{\cdot}d$ while having earned an accumulated reward $k{\cdot}d$. According to [24], we have:

$$\Pr\{Y_t \leqslant r, X_t \in S'\} \approx \sum_{s\in S'}\sum_{k=1}^{R} F^T(s,k){\cdot}d \quad \text{where } R=\frac{r}{d} \text{ and } T=\frac{t}{d}.$$

Note that R and T are integers, as r and t are both multiples of d. Matrix $F^T(s,k)$ is computed in an iterative manner where

$$F^1(s,k) = \begin{cases} 1/d, & \text{if } (s,k)=(s_0,\rho(s_0)) \\ 0, & \text{otherwise} \end{cases}$$

(recall that s_0 is the initial state of the MRM). For the subsequent iterations, the following recursive scheme[2] is used:

$$F^{j+1}(s,k) = F^j(s,k{-}\rho(s)){\cdot}(1-\mathbf{E}(s){\cdot}d) + \sum_{s'\in S} F^j(s',k{-}\rho(s')){\cdot}\mathbf{R}(s',s){\cdot}d,$$

where $k{-}\rho(s)$ is set to 0 if $\rho(s)>k$. This expression can be explained as follows. At the $(j{+}1)$-st time instant, either the MRM was in state s at the j-th time instant and remained there for d time-units (the first summand), or it has moved from a state s' to state s during that period (the second summand). Given that the accumulated (discretised) reward is k, the accumulated reward at the j-th instant is approximated by $k{-}\rho(s)$ and $k{-}\rho(s')$, respectively.

In total, t/d iterations are needed to obtain the desired result. Due to the state-dependent displacements, i.e., $k{-}\rho(s)$, both matrices F^j and F^{j+1} need to be stored, thus occupying $2{\cdot}|S|{\cdot}R$ floating point numbers. The time complexity of this method is $\mathcal{O}(|S|{\cdot}t{\cdot}|(t{-}r)|{\cdot}d^{-2})$. As the computational effort is proportional to d^{-2}, the computation time grows rapidly when a higher accuracy is required.

4.4. Occupation time distributions

In 2000, Sericola [23] derived a result for the computation of weighted sums of occupation times in CTMCs. The approach is based on uniformisation. Assume the MRM has $m{+}1$ different rewards $\rho_0<\rho_1<\cdots<\rho_{m-1}<\rho_m$. Reward ρ_0 needs to be 0, which in our case is ensured (since the rewards of the absorbing states are set to 0). Define $H_{i,j}(t,r)$ to be the probability that at time t the MRM is in state j and has accumulated a reward higher than r, having started in state i:

$$H_{i,j}(t,r) = \Pr\{Y_t > r, X_t = j \mid X_0 = i\}.$$

In [23, Theorem 5.6], Sericola states the following expression for the matrix $\mathbf{H}(t,r)$ which contains entries $H_{i,j}(t,r)$ (for all $i,j \in S$), for $r \in [\rho_{h-1}t, \rho_h t)$ with $1 \leqslant h \leqslant m$:

$$\mathbf{H}(t,r) = \sum_{n=0}^{\infty} e^{-\lambda t}\frac{(\lambda t)^n}{n!}\sum_{k=0}^{n}\binom{n}{k} x_h^k(1-x_h)^{n-k}\mathbf{C}(h,n,k), \tag{1}$$

where n corresponds to the number of steps in the uniformised discrete-time Markov chain. The value $x_h = \frac{x-\rho_{h-1}t}{(\rho_h-\rho_{h-1})t} \in [0,1)$ represents the normalisation of r to the interval $[\rho_{h-1}t, \rho_h t)$. $\mathbf{C}(h,n,k)$ is a square matrix defined recursively in terms of h, n and k, thereby using the uniformised matrix $\mathbf{P}=\mathbf{I}-\mathbf{Q}/\lambda$, with λ the uniformisation rate for the original CTMC.

The complementary probabilities we need, can then be computed as:

$$\begin{aligned}\Pr\{Y_t \leqslant r, X_t = j \mid X_0 = i\} &= \\ \Pr\{X_t = j \mid X_0 = i\} &- \Pr\{Y_t > r, X_t = j \mid X_0 = i\} \\ &= \Pr\{X_t = j \mid X_0 = i\} - H_{i,j}(t,r),\end{aligned}$$

that is, we additionally need the transient probabilities at time $t>0$. These can be computed simultaneously with $H_{i,j}(t,r)$.

To obtain what is needed according to Theorem 2, we use that s_0 is the unique initial state, and hence

$$\Pr\{Y_t \leqslant r, X_t = j\} = \Pr\{Y_t \leqslant r, X_t = j \mid X_0 = s_0\}.$$

Sericola proves [23, Corollary 5.8] that the matrices $\mathbf{C}(h,n,k)$ are non-negative and smaller than the matrix $\mathbf{P}^n$. Using this fact and recognising that the inner sum in (1) represents a binomial distribution, allows us to conclude that the inner sum is always smaller than one, hence we can use the Poisson probabilities to determine an a priori bound N_ε, the number of steps needed to reach an error bound $\varepsilon>0$:

$$\sum_{n=0}^{N_\varepsilon} e^{-\lambda t}\frac{(\lambda t)^n}{n!} > 1-\varepsilon.$$

N_ε increases with time t and the uniformisation constant λ.

The computational and storage requirements of the approach are considerable. In the n-th step of the outer sum,

[2] This recursive scheme exhibits quite some similarities with the uniformisation approach, but evaluation takes place in non-uniformised time steps.

we must compute the elements of $((m+1)\cdot n\cdot n)$ matrices of dimension $|S|\times|S|$. Since $m=\mathcal{O}(|S|)$, the complexity of computing the n-th step is $\mathcal{O}(n^2|S|^3)$. If we truncate after the N_ε-th step, we get an overall time complexity of $\mathcal{O}(N_\varepsilon^3|S|^3)$ and an overall space complexity of $\mathcal{O}(N_\varepsilon^2|S|)$.

5. Case study: performability of ad hoc networks under power constraints

This section illustrates the usefulness of CSRL to reason about complex dependability and performability measures. The example is taken from the area of battery powered mobile ad hoc networks, such as IEEE 802.11 or Hiperlan [25]. After describing the case study and the CSRL properties of interest, we report on initial implementations of the computational procedures discussed in the previous section.

5.1. Battery powered ad hoc networks

The distinguishing feature of an ad hoc network lies in its ability to establish a wireless connection between remote stations by allowing data to travel through intermediate neighbouring stations that are in mutual reach. From the point of view of a single station, the support for the ad hoc mode has the flavour of altruism, because the station does itself not benefit from offering its transmission capabilities for data transfer between third parties. This altruism adds of course to the out-reach of the local station, since it can use the transmission capabilities of the other members of the network.

On the other hand, battery powered mobile stations work under stringent power constraints, and therefore there is an interesting tradeoff when comparing the power consumption due to ad hoc traffic with the power requirements needed to fulfil the prime needs of the mobile station under study. An increased use of battery-powered devices is envisaged in the near future. Hence, techniques to reason about the dependability and performance of such systems under power constraints will become increasingly important.

5.2. The model

We consider a single battery powered mobile station and model a simplified behaviour of the station due to ad hoc traffic, and due to ordinary traffic, i.e., due to calls that are originating from (or are directed to) the station considered. The model is inspired by the state transition diagram described in [25, p. 508]. We model the station as a stochastic reward net (SRN) [6], which allows us to directly represent the concurrent handling of both types of traffic. Rate rewards are used to model the different degrees of power consumption in the various situations. As in the IEEE 802.11 standard, the station provides the possibility to turn into *doze* mode, where it is neither able to receive nor to transmit, and where its power consumption is very low [25, p. 705].

transition	mean time	rate (per hour)
accept	20 sec	180
connect	10 sec	360
disconnect	4 min	15
doze	5 min	12
give up	1 min	60
interrupt	1 min	60
launch	80 min	0.75
reconfirm	4 min	15
request	10 min	6
ring	80 min	0.75
wake up	16 min	3.75

place	reward
Ad hoc Active	150 mA
Ad hoc Idle	50 mA
Call Active	200 mA
Call Idle	50 mA
Call Incoming	150 mA
Call Initiated	150 mA
Doze	20 mA

Table 1. Transition rates and rewards for the SRN in Figure 2.

The SRN depicted in Fig. 2 describes the behaviour of the station. Whenever the station is not in *Doze* mode, it can handle *ad hoc* traffic and ordinary *calls* concurrently. Ad hoc traffic is handled upon a *request* issued from some neighbouring station. After having processed the call, the stations *reconfirm* each other about successful transmission, before the station turns back to *Ad hoc idle* mode. Outgoing calls from the station under study can either get *connect*ed after being *launch*ed by the user, or they can get interrupted by the user (*give up*) while being processed. Once connected, voice or data transfer continues until the call is *disconnect*ed. Incoming calls are indicated through a *ring*ing bell. If the call is *accept*ed, connection is established, otherwise the pending incoming call can be *interrupt*ed by the remote station. If both threads of control are idle, the station can decide to *doze*, until a *wake up* occurs.

The transition rates of the SRN are chosen as listed in Table 1, for instance the mean time until a disconnect occurs is assumed to be 4 minutes. This corresponds to a

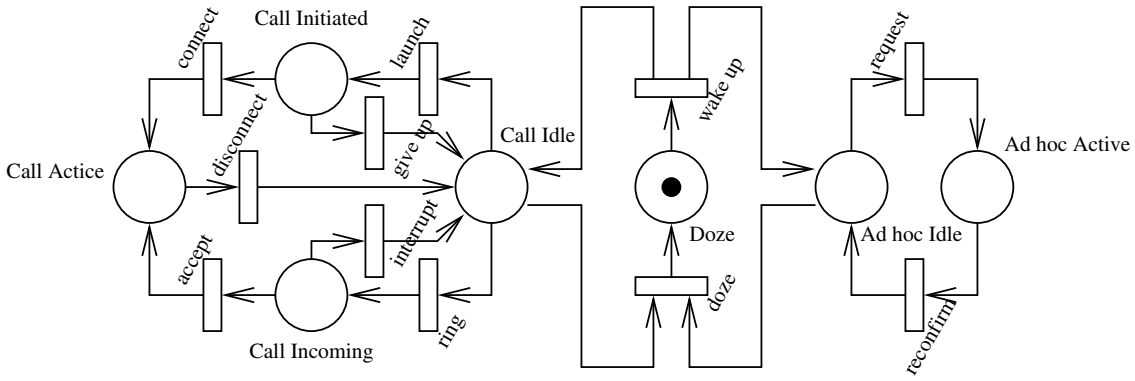

Figure 2. SRN description of a battery powered mobile station in an ad hoc network

mean length of a call of four minutes. Rewards are assigned to the system as follows: in *Doze* mode very low power is consumed (20 mA). In all other states we assume the power consumption to be additive for the two concurrent tasks carried out, that is, in all states the overall reward is the sum of the rewards of the non-empty places, according to the assignment in Table 1. We assume high power consumption for active calls (200 mA), and less power consumption for active ad hoc traffic and the call handling phases (150 mA). In idle phases, the station consumes low power (50 mA per task).

We emphasise that the rewards and rates used in the above model are debatable, because they are not resulting from any kind of measurement or analytical activities, they are instead results of educated guesses based on the power consumption of mobile phones.

5.3. Properties of the model

Performability properties of interest to be studied for this model are, for instance:

Q1. Is the probability larger than 0.5 to receive an incoming call before having consumed at most 80% of the power?

Q2. Is the probability larger than 0.5 to receive an incoming call within 24 hours?

Q3. Is the probability larger than 0.5 to launch an outbound call before having consumed at most 80% of the power within 24 hours, without using the phone except for ad hoc transfer beforehand?

We assume a battery to have 750 mAh capacity when fully charged and set the basic time unit to 1 hour and the basic reward unit to 1 mA. Atomic propositions are given by the place names in the SRN model, that is, in a state those atomic propositions hold, for which the corresponding place contains a token. Then, the above properties are translated into CSRL as follows:

Q1. $\mathcal{P}_{>0.5}(\Diamond_{\leqslant 600}\ \ \mathit{Call\_Incoming})$

Q2. $\mathcal{P}_{>0.5}(\Diamond^{\leqslant 24}\ \ \mathit{Call\_Incoming})$

Q3. $\mathcal{P}_{>0.5}((\mathit{Call\_Idle} \vee \mathit{Doze})\ \ \mathcal{U}^{\leqslant 24}_{\leqslant 600}\ \ \mathit{Call\_Initiated})$

5.4. Model-checking power consumption

Since the model-checking procedures of properties of type **Q1** and **Q2** are well investigated, we only study property **Q3**. More precisely, we compute the probability of satisfying the path formula $(\mathit{Call\_Idle} \vee \mathit{Doze})\ \ \mathcal{U}^{\leqslant 24}_{\leqslant 600}\ \ \mathit{Call\_Initiated}$. The MRM $\mathcal{M}$ underlying the given SRN has nine recurrent states. Applying our Theorem to property **Q3**, we obtain a reduced MRM $\mathcal{M}'$ with three transient and two absorbing states which will be the input for the three numerical methods.

All reported times are user CPU times on a PC with an Intel Pentium III 1GHz processor running under Linux. The pseudo-Erlang approximation has been modelled using the stochastic reward net tool SPNP (version 6) [6] while the other two algorithms have been implemented in C/C++.

Occupation time distributions. Table 2 lists the probabilities of satisfying the path formula as well as the computation time using the approach of Sericola for different error bounds ε. The convergence of the computation is rather good, and time consumption is moderate. Since this algorithm gives us the most exact result, we compute the error for the outputs of the other two algorithms relatively to this one.

ε	N	numerical value	time
10^{-1}	496	0.44831203	76.27 sec
10^{-2}	519	0.49068833	83.00 sec
10^{-3}	536	0.49492396	89.51 sec
10^{-4}	551	0.49536172	94.76 sec
10^{-5}	563	0.49539940	99.19 sec
10^{-6}	574	0.49540351	103.09 sec
10^{-7}	585	0.49540395	107.11 sec
10^{-8}	594	0.49540399	110.78 sec

Table 2. Results obtained by the occupation time distribution algorithm.

k	numerical value	relative error	time
1	0.41067310	17.10%	$<$ 0.01 sec
2	0.45466923	8.22%	$<$ 0.01 sec
4	0.47730297	3.65%	$<$ 0.01 sec
8	0.48742851	1.61%	$<$ 0.01 sec
16	0.49177955	0.73%	0.01 sec
32	0.49369656	0.34%	0.02 sec
64	0.49457832	0.17%	0.05 sec
128	0.49499840	0.08%	0.15 sec
256	0.49520304	0.04%	0.50 sec
512	0.49530398	0.02%	2.02 sec
1024	0.49535410	0.01%	21.34 sec

Table 3. Results obtained by the pseudo-Erlang approximation.

Pseudo-Erlang approximation. Table 3 lists the probabilities of satisfying the path formula using the pseudo-Erlang approach, the relative error and time consumption, for varying numbers k of phases. As expected, an increase in the number of phases increases the time consumption, but decreases the relative error. Notice that k should be considerable to obtain a good approximation. Also note that all probabilities are smaller than the ones computed with the previous technique. Whether this is always the case needs to be investigated further.

Discretisation. Table 4 lists the probabilities of satisfying the path formula using the approach of Tijms-Veldman, where the step size of the discretisation d is varied. Again, also the time consumption and the relative error are shown. We observe that the convergence of the computation is rather good, but time consumption is considerable.

d	numerical value	relative error	time
$\frac{1}{32}$	0.49566676	0.05%	26.71 sec
$\frac{1}{64}$	0.49553603	0.03%	107.62 sec
$\frac{1}{128}$	0.49547017	0.01%	431.93 sec
$\frac{1}{256}$	0.49543712	$<$ 0.01%	1712.00 sec

Table 4. Results obtained by the discretisation algorithm.

General observations. From these initial computational experiments, we can report the following observations:

- The three computational procedures converge to the same value, however, only for the occupation time distribution approach an a priori error bound (and hence a stopping criterion) is available.
- The method based on occupation time distributions is fast and accurate. In the current case study (which is small) we did not run into storage problems, however, the cubic storage requirements will limit this method to relatively small case studies.
- The discretisation method is slow when a fine-grain discretisation is used. Unfortunately, we have no method available (yet) to get a hold on the required step size to achieve a certain accuracy.
- The pseudo-Erlang approach is fast (where we did not exploit the special tensor structure in the generator matrix; we simply used SPNP), but also here, we have to guess a reasonable number of phases for the approximation.
- In the pseudo-Erlang approximation around 250 phases are required to obtain a three-digit accuracy.
- The discretisation method suffers particularly from large time-bounds and large state spaces, as these make the number of matrices to be computed larger.
- The method based on occupation time distributions becomes less attractive when the time bound is large in comparison to the uniformisation rate. We are currently investigating the convergence of the matrices $\mathbf{C}(h, n, k)$ to see whether some kind of steady-state detection can be employed to shorten the series.

6. Conclusions

In this paper we have investigated the use of three computational procedures for the computation of time- and

reward-bounded until-expressions of the logic CSRL over Markov reward models. This type of measure has not been considered in the literature before. In particular, it extends the scope of the performability measures as proposed by Meyer, and hence enriches the performability evaluation framework in a non-trivial way. With a small case study in the area of ad hoc mobile networking under power constraints we have shown the feasibility and the usefulness of our approach; due to increased mobility more and more systems are expected to perform well (in terms of performance and/or dependability) under power constraints. Using the logic CSRL we are able to express such system properties precisely.

In the near future, we will investigate whether algorithms proposed in the area of non-Markovian and fluid stochastic Petri nets can be used for our purpose. We will furthermore extend our algorithms to cases in which the time- and reward intervals are of a more general nature, i.e., not just starting at 0, and will extend the approach to impulse rewards.

References

[1] A. Aziz, K. Sanwal, V.Singhal, R. Brayton. Verifying continuous time Markov chains. *Lecture Notes in Computer Science*, 1102: 269–276, 1996.

[2] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. *Lecture Notes in Computer Science*, 1664: 146–161, 1999.

[3] C. Baier, B.R. Haverkort, J.-P. Katoen, and H. Hermanns. Model checking continuous-time Markov chains by transient analysis. *Lecture Notes in Computer Science*, 1855: 358–372, 2000.

[4] C. Baier, B.R. Haverkort, J.-P. Katoen, and H. Hermanns. On the logical specification of performability properties. *Lecture Notes in Computer Science*, 1853: 780–792, 2000.

[5] A. Bobbio, M. Telek. Markov regenerative SPN with non-overlapping activity cycles. *Proc. Int'l IEEE Performance and Dependability Symposium*: 124–133, 1995.

[6] G. Ciardo, J. Muppala, K. S. Trivedi. SPNP: Stochastic Petri Net Package. *Proc. 3rd Int'l Workshop on Petri Nets and Performance Models*: 142–151, 1989.

[7] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2): 244–263, 1986.

[8] E.M. Clarke, O. Grumberg, D. Peled. *Model Checking*. MIT Press, 1999.

[9] L. Donatiello and B.R. Iyer. Analysis of a composite performance reliability measure for fault-tolerant systems. *J. ACM*, 34(1): 179–199, 1987.

[10] R. German. *Performance Analysis of Communication Systems: Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley & Sons, 2000.

[11] A. Goyal and A.N. Tantawi. A measure of guaranteed availability and its numerical evaluation. *IEEE Trans. Comput.*, 37: 25–32, 1988.

[12] D. Gross and D.R. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov chains. *Oper. Res.*, 32(2): 343–361, 1984.

[13] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5): 512–535, 1994.

[14] B.R. Haverkort, H. Hermanns, J.-P. Katoen. On the use of model checking techniques for dependability evaluation. *Proc. 19th IEEE Symposium on Reliable Distributed Systems*: 228–237, 2000.

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

[16] G. Horton, V. Kulkarni, D. Nicol, K. Trivedi. Fluid stochastic Petri nets: Theory, application and solution techniques. *Eur. J. Oper. Res.*, 105(1): 184–201,1998.

[17] A. Jensen. Markov chains as an aid in the study of Markov processes. *Skand. Aktuarietidskrift*, 3: 87–91, 1953.

[18] J.F. Meyer. On evaluating the performability of degradable computer systems. *IEEE Trans. Comput.*, 29(8), 720–731, 1980.

[19] J.F. Meyer. Closed-form solutions of performability, *IEEE Trans. Comput.*, 31(7): 648–657, 1982.

[20] J.F. Meyer. Performability: a retrospective and some pointers to the future. *Performance Evaluation*, 14(3&4): 139–156, 1992.

[21] W.D. Obal, W.H. Sanders. State-space support for path-based reward variables. *Performance Evaluation*, 35: 233–251, 1999.

[22] W.D. Obal, W.H. Sanders. Measure-Adaptive State-Space Construction. *Performance Evaluation*, 44: 237–258, 2000.

[23] B. Sericola. Occupation times in Markov processes. *Stochastic Models*, 16(5): 339–351, 2000.

[24] H.C. Tijms, R. Veldman. A fast algorithm for the transient reward distribution in continuous-time Markov chains, *Oper. Res. Lett.*, 26: 155–158, 2000.

[25] B. Walke. *Mobile Radio Networks*. John Wiley & Sons, 1999.