Model Checking Markov Reward Models with Impulse Rewards

Lucia Cloth[†], Joost-Pieter Katoen[‡,†], Maneesh Khattri[†] and Reza Pulungan[§,1]

[†]Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

[‡]Department of Computer Science, RWTH Aachen
Ahornstrasse 55, D-52072 Aachen, Germany

[§]Department of Computer Science, Saarland University
D-66123 Saarbrücken, Germany

{lucia, katoen, khattri}@cs.utwente.nl; pulungan@cs.uni-sb.de

Abstract

Markov reward models (MRMs) have been applied to the simultaneous analysis of performance and dependability of computer systems. The combination of performance and dependability is sometimes referred to as performability. *MRMs are continuous-time Markov chains with rewards assigned to their states (state reward structure) and transitions (impulse reward structure). Model checking for such MRMs has been introduced earlier along with a logic, Continuous Stochastic Reward Logic (CSRL), for the expression of relevant properties of MRMs. Existing model-checking methods are, however, applicable to MRMs with state reward structures only.*

In this paper MRMs, CSRL and model-checking algorithms for MRMs are extended to incorporate impulse reward structures. Numerical methods which are applicable for the computation of the measures defined are presented. The applicability of these algorithms is demonstrated in the context of dynamic power management.

1. Introduction

A *Markov reward model* (MRM) is a Markov chain augmented with a *reward assignment function*. The reward assignment function assigns a *state-based reward rate* to each state such that the residence time in a state entails the accumulation of overall reward gained based on the state-based reward rate. The reward assignment function also assigns an *impulse reward* to each transition such that the occurrence of the transition results in a gain in the overall reward governed by the impulse reward assigned to the transition.

Markov reward models have been applied to the simultaneous analysis of *performance* and *dependability* of com-

puter systems [13]. Whilst performance in computer systems is the efficacy by which the system delivers services under the assumption that the services delivered conform to the specification, dependability is the ability of the system to deliver services that conform to the specification. Although these two issues may be analyzed independently, simultaneous analysis becomes imperative when the performance of the system *degrades* under the presence of behavior that does not conform to the specification. The combination of performance and dependability is sometimes referred to as *performability* [14].

Model checking [6] is a *formal verification technique*. The verification of systems requires first a precise *specification* of the system. A *model* of the system is then constructed whose behavior should correspond to the specification. Once such a model is available, *statements* about functional correctness of the system can be made. Subsequently it has to be ascertained whether the model satisfies these statements. If the model satisfies all these statements then it is said to be *correct*. This prominent technique to ascertain correctness is called model checking.

2. Related Work and Contributions

Model checking for MRMs has been studied in [2, 3, 9]. This paper extends these techniques to MRMs with state-based reward rates and impulse rewards. This includes the extension of the logic Continuous Stochastic Reward Logic, CSRL, and the associated logical characterization.

Based on [9], this paper also presents numerical methods for MRMs with impulse rewards. To treat until formulas, discretization [21] and uniformization techniques for computing transient reward distributions are *generalized* for reward models with impulse rewards. Uniformization based methods in [15, 16] are considered. Shortcomings of these methods are that they are computationally expensive and that they are susceptible to numerical instability. The computational performance was improved in [20], however the

---

[1]Part of this work was done when the author was at the University of Twente.

resulting algorithm is still susceptible to numerical instability. The main concentration here is on enhancing the numerical stability. This is achieved by the use of the recent algorithm in [7] with that in [15, 16].

The motivation in extending model-checking techniques for MRMs to incorporate impulse rewards has been to study the instantaneous costs related to transitions. It has been argued, for instance in [5], that the cost related to transitions is not negligible. One may also consider the examples in [14, 15, 16, 20]. An example application in the context of *dynamic power management* is presented to demonstrate the applicability of the model-checking algorithms.

3. Markov Reward Models

MRMs are a class of models which have been studied for the simultaneous analysis of performance and dependability issues of computer systems [13, 14]. In this section, the definition of MRMs for the expression of qualitative and quantitative properties of such models is presented. Furthermore, the logic CSRL for expressing properties is presented.

3.1. CTMCs and MRMs

Firstly, for the expression of qualitative properties, consider a set AP of atomic propositions. These are the most elementary facts that can be said about a model.

Definition 1 *A labeled continuous-time Markov chain (CTMC) is a triple $(S, \boldsymbol{R}, Label)$ where S is a finite set of states, $\boldsymbol{R} : S \times S \longrightarrow \mathbb{R}_{\geq 0}$ is a function and $Label : S \longrightarrow 2^{AP}$ is a labeling function.*

$\boldsymbol{R}$ is called the *rate matrix* of the CTMC, where $\boldsymbol{R}(s, s')$ is the rate of moving from state s to s'. There is a transition from state s to state s' if and only if $\boldsymbol{R}(s, s') > 0$ for s, $s' \in S$. The total rate[1] of taking an outgoing transition from state $s \in S$ is given by $E(s) = \sum_{s' \in S} \boldsymbol{R}(s, s')$. A state s with $E(s) = 0$ is called an *absorbing* state. The probability of moving from state s to a state s' is $\boldsymbol{P}(s, s') = \dfrac{\boldsymbol{R}(s, s')}{E(s)}$. The probability of making a transition from state s to s' within time t is $\frac{\boldsymbol{R}(s, s')}{E(s)} \cdot (1 - e^{-E(s) \cdot t})$. The initial distribution of the CTMC is given by $\underline{p}(0)$.

Definition 2 *A labeled Markov reward model (MRM) $\mathcal{M}$ is a triple $((S, \boldsymbol{R}, Label), \underline{\rho}, \boldsymbol{\iota})$ where $(S, \boldsymbol{R}, Label)$ is the underlying labeled CTMC, $\underline{\rho} : S \longrightarrow \mathbb{R}_{\geq 0}$ is the state reward structure, and $\boldsymbol{\iota} : S \times \tilde{S} \longrightarrow \mathbb{R}_{\geq 0}$ is the impulse reward structure satisfying $\iota(s, s) = 0$ for any state s.*

An MRM is a labeled CTMC augmented with state-based reward and impulse reward structures. The state

---

[1] As we allow for self-loops (i.e., states s with $\boldsymbol{R}(s, s) > 0$) as e.g. in [4], "leaving" state s includes that the self-loop $s \to s$ (if any) may be taken.

reward structure is a function ρ that assigns to each state $s \in S$ a reward rate $\underline{\rho}(s)$ such that if t time-units are spent in state s, a reward of $\rho(s) \cdot t$ is acquired.

The impulse reward structure, on the other hand, is a function $\boldsymbol{\iota}$ that assigns to each transition from s to s', where $s, s' \in S$, a reward $\boldsymbol{\iota}(s, s')$ such that if the transition from s to s' occurs, a reward of $\boldsymbol{\iota}(s, s')$ is acquired.

Example 1 *Consider a WaveLAN modem modeled as MRM cf. Figure 1. This modem is designed to be energy-efficient. Typical operating modes in such an interface are* off, sleep, idle, receive *and* transmit *[10]. In the* transmit *mode it is transmitting and is receiving data in the* receive *mode. In the* idle *mode it is idle but ready to either transmit or to receive. In the* sleep *mode the interface is powered down.*
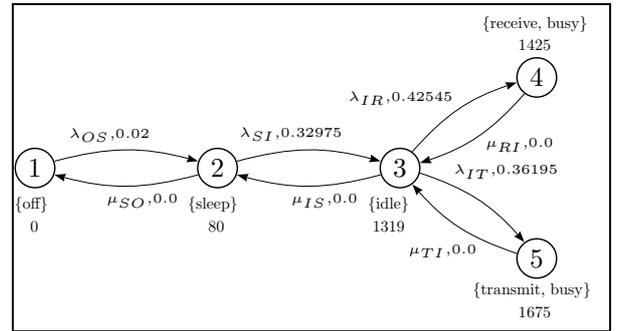

Figure 1. WaveLAN Modem MRM

The state-based reward and impulse reward structures are interpreted as energy consumption. A WaveLAN modem, as described in [10], typically consumes $1675 \ mW$ while transmitting, $1425 \ mW$ while receiving, $1319 \ mW$ while being idle and $80 \ mW$ while in sleep *mode. It is also described that transitions from* sleep *to* idle *take $250 \ \mu s$ with the power consumption of the* idle *mode; and that $254 \ \mu s$ is required before the payload is transmitted. It is further assumed that the same duration is required before a payload can be received. Similarly the transition from* off *to* sleep *is assumed to take $250 \ \mu s$ with the power consumption of the* sleep *mode.*

3.2. Paths in MRM

During its execution an MRM makes transitions from states to states. The sequence of states that the MRM visits is called a *path*. Consider an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \underline{\rho}, \boldsymbol{\iota})$.

Definition 3 *An infinite path σ in $\mathcal{M}$ is a sequence $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \cdots$, with $s_i \in S$ and $t_i \in \mathbb{R}_{>0}$ the amount of time spent in state s_i where $i \in \mathbb{N}$ and $\boldsymbol{R}(s_i, s_{i+1}) > 0$ for all i. A finite path σ in $\mathcal{M}$ is a sequence $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \cdots \xrightarrow{t_{n-1}} s_n$, such that s_n is an absorbing state and $\boldsymbol{R}(s_i, s_{i+1}) > 0$ for*

all $i < n$. For finite paths that end in s_n, $t_n = \infty$. For infinite path σ, let $\sigma[i] = s_i, i \in \mathbb{N}$. For a finite path σ that ends in s_n, $\sigma[i] = s_i$ is only defined for $i \leq n$. The last state in the finite path σ is referred to as $last(\sigma)$. The state being occupied at time t on path σ is defined as:

$$\sigma@t = \sigma[i] \Leftrightarrow \sum_{j=0}^{i-1} t_j \leq t \ \wedge \ \sum_{j=0}^{i} t_j > t.$$

Accumulated rewards at time t in a path σ such that $\sigma@t = \sigma[i]$ is a function $y_\sigma : \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}_{\geq 0}$ defined by:

$$
\begin{aligned}
y_\sigma(t) &= \underline{\rho}(\sigma@t) \cdot \left(t - \sum_{j=0}^{i-1} t_j \right) + \sum_{j=0}^{i-1} \underline{\rho}(\sigma[j]) \cdot t_j \\
&\quad + \sum_{j=0}^{i-1} \boldsymbol{\iota}(\sigma[j], \sigma[j+1]).
\end{aligned}
$$

State $\sigma@t$ is the i-th state on path σ if the residence time in states before the i-th state is at most t and the residence time of all states including the i-th state is greater than t. Let $\sigma@t = \sigma[i]$. The first, second and third summand of $y_\sigma(t)$ are the accumulated state-based reward due to residence in the i-th state until time t, the accumulated state-based reward due to residence in states prior to the visit to the i-th state and the accumulated impulse rewards on the path σ, respectively.

Example 2 *Consider the infinite path in the WaveLAN modem:*

$$\sigma = 1 \xrightarrow{10} 2 \xrightarrow{4} 3 \xrightarrow{2} 4 \xrightarrow{3.75} 3 \xrightarrow{1} 5 \xrightarrow{2.5} 3 \xrightarrow{5} \cdots .$$

The state that is occupied at time 21.75 is 5 since:

$$\sum_{j=0}^{4} t_j = 20.75 \leq 21.75 \ \wedge \ \sum_{j=0}^{5} t_j = 23.25 > 21.75.$$

The accumulated reward at that time is:

$$
\begin{aligned}
y_\sigma(21.75) &= \underline{\rho}(\sigma[5]) \cdot \left(21.75 - \sum_{j=0}^{4} t_j \right) \\
&\quad + \sum_{j=0}^{4} \underline{\rho}(\sigma[j]) \cdot t_j + \sum_{j=0}^{4} \boldsymbol{\iota}(\sigma[j], \sigma[j+1]) \\
&= 11984.38715 \ mJ.
\end{aligned}
$$

A probability measure over sets of paths in an MRM can be defined in the standard way using a cone construction. For a formal definition of the Borel space that underlies such measure we refer to [4]. For the remainder of this paper it suffices to assume that such measure exists.

3.3. A Logic for MRMs

This section presents the syntax of the *Continuous Stochastic Reward Logic* (CSRL) [2, 9]. In CSRL, two kinds of formulas are distinguished viz, *state formulas* and *path formulas*.

Definition 4 *Let $p \in [0, 1]$ be a real number, $\trianglelefteq \in \{<, \leq, \geq, >\}$ be a comparison operator, $a \in AP$, I and J intervals of non-negative real numbers. The syntax of CSRL state formulas (Φ) and path formulas (φ), respectively, over the set of atomic propositions AP is defined as follows:*

$$
\begin{aligned}
\Phi &::= \quad \mathtt{tt} \mid a \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathcal{S}_{\trianglelefteq p}(\Phi) \mid \mathcal{P}_{\trianglelefteq p}(\varphi). \\
\varphi &::= \quad \mathcal{X}_J^I \Phi \mid \Phi \mathcal{U}_J^I \Psi.
\end{aligned}
$$

CSRL formulas are interpreted over labeled MRMs by a satisfaction relation ($\vDash$) between a state s and a state formula Φ, and between a path σ and a path formula φ. A satisfaction relation is called valid iff a state formula is satisfied in a state or a path formula is satisfied in a path. If state s satisfies state formula Φ then: $s \vDash \Phi$ and if path σ satisfies path formula φ then: $\sigma \vDash \varphi$ are said to be valid. The formula $\mathcal{S}_{\trianglelefteq p}(\Phi)$ is referred to as the *steady-state formula*. This formula asserts that the steady-state probability for the set of Φ-states meets the bound $\trianglelefteq p$. The formula $\mathcal{P}_{\trianglelefteq p}(\varphi)$ is called the *path probability formula*, which asserts that the probability measure of paths satisfying φ meets the bound $\trianglelefteq p$.

Interval I is a timing constraint, while interval J is a bound for the accumulated reward. The formula $\mathcal{X}_J^I \Phi$ is called *next formula*, and asserts that a transition can be made to a Φ-state at time $t \in I$ such that the accumulated reward until time t, lies in J. The formula $\Phi \mathcal{U}_J^I \Psi$ is referred to as *until formula*. This formula asserts that Ψ is satisfied at some time $t \in I$ such that the accumulated reward until t, lies in J and Φ is satisfied at all instants before t. A detailed description of the semantics of CSRL is outside the scope of this paper and can be found in [2].

Example 3 *Energy consumption in WaveLAN can be reduced if the interface spends most of the time in* sleep *mode. Accordingly, requirements on the ability of the system to reach the* sleep *state by spending a defined amount of energy are imposed, e.g. the probability is at least 0.8 that the system reaches the* sleep *state via* busy *or* idle *states before 10 s have elapsed and by spending at most 5 J of energy. This property can be expressed in CSRL by: $\mathcal{P}_{\geq 0.8}((busy \vee idle)\mathcal{U}_{[0,5]}^{[0,10]} sleep)$.*

4. Characterizing CSRL

MRMs [14] enrich stochastic processes with information related to the consumption or completion of resources or of activities. The logic CSRL [9] allows most traditional measures for MRMs to be expressed in a precise fashion and facilitates the expression of new and more complicated measures. For the realization of a model checker for this logic over MRMs, appropriate numerical computation techniques have to be identified. In this section, a characterization of the computation associated with each of the formulas in CSRL is presented. In the sequel, let $\mathcal{M} = ((S, \boldsymbol{R}, Label), \underline{\rho}, \boldsymbol{\iota})$ be an MRM and $Sat(\Phi)$ be the set of states in $\mathcal{M}$ that satisfy state formula Φ.

Given $\mathcal{M}$, a starting state $s_0 \in S$ and a formula of the form $\mathcal{S}_{\trianglelefteq p}(\Phi)$, the question whether s_0 satisfies the given formula, can be established by using techniques in [4] and is not further explored here.

4.1. Ne$\mathcal{X}$t

To resolve whether the path probability formula of the form $s \vDash \mathcal{P}_{\trianglelefteq p}(\varphi)$ is satisfied by the given model and state s, it is first necessary to determine the probability with which the given path formula φ is satisfied. Define:

$$K(s) = \{x \in I | \underline{\rho}(s) \cdot x \in J\}, \text{ and}$$
$$K(s,s') = \{x \in I | (\underline{\rho}(s) \cdot x + \iota(s,s')) \in J\},$$

for closed intervals I and J. $K(s)$ is an interval of time in I such that the accumulated state-based reward by residing in state s for any length of time lies in J. $K(s,s')$ is an interval of time in I such that the sum of the accumulated state-based reward by residing in state s for any length of time in $K(s,s')$ and the impulse reward acquired by moving from state s to s' lies in J. The probability density of moving from state s to s' within x time-units is $P(s,s',x) = \boldsymbol{P}(s,s') \cdot E(s) \cdot e^{-E(s) \cdot x}$. Hence, the probability of leaving state s within the interval I while respecting the reward bound after the outgoing transition has taken place is:

$$P_J^I(s, S') = \sum_{s' \in S'} \int_{K(s,s')} P(s,s',x) dx, \qquad (1)$$

where $S' \subseteq S$ and $s \in S$.

For formula $\mathcal{P}_{\trianglelefteq p}(\mathcal{X}_J^I \Phi)$, let $P^{\mathcal{M}}(s, \mathcal{X}_J^I \Phi)$ be the probability of all paths starting from state s satisfying the formula $(\mathcal{X}_J^I \Phi)$. Then:

$$P^{\mathcal{M}}(s, \mathcal{X}_J^I \Phi) = P_J^I(s, Sat(\Phi)). \qquad (2)$$

4.2. $\mathcal{U}$ntil

For formula $s \vDash \mathcal{P}_{\trianglelefteq p}(\Phi \mathcal{U}_J^I \Psi)$, let $P^{\mathcal{M}}(s, \Phi \mathcal{U}_J^I \Psi)$ be the probability of all paths starting from state s satisfying the formula $(\Phi \mathcal{U}_J^I \Psi)$. First let $L \ominus y = \{l - y | l \in L \wedge l \geq y\}$, $K_1(s) = \inf(K(s))$, $K_1(s,s') = \sup(K(s,s'))$, $I(x) = I \ominus x$ and $J(x,s,s') = J \ominus (\underline{\rho}(s) \cdot x + \iota(s,s'))$, then $P^{\mathcal{M}}(s, \Phi \mathcal{U}_J^I \Psi)$ is the least solution of the following set of equations:

$$P^{\mathcal{M}}(s, \Phi \mathcal{U}_J^I \Psi) \qquad (3)$$
$$= \begin{cases} 1, \text{ if } s \vDash \Psi \text{ and } \inf(I) = 0 \text{ and } \inf(J) = 0, \\ \sum_{s' \in S} \int_0^{K_1(s,s')} P(s,s',x) \\ \quad \cdot P^{\mathcal{M}}(s', \Phi \mathcal{U}_{J(x,s,s')}^{I(x)} \Psi) dx, \\ \qquad \qquad \text{ if } s \vDash \Phi \wedge \neg \Psi, \\ e^{-E(s) \cdot K_1(s)} \\ + \sum_{s' \in S} \int_0^{K_1(s)} P(s,s',x) \\ \quad \cdot P^{\mathcal{M}}(s', \Phi \mathcal{U}_{J(x,s,s')}^{I(x)} \Psi) dx, \\ \qquad \qquad \text{ if } s \vDash \Phi \wedge \Psi, \\ 0, \text{ otherwise.} \end{cases}$$

The justification of this characterization is as follows:

1. If s satisfies Ψ and $\inf(I) = 0$ and $\inf(J) = 0$ then all paths starting from state s satisfy the formula and consequently the probability is 1.

2. If s satisfies $\Phi \wedge \neg \Psi$ then the probability of reaching a Ψ-state from state s within the interval I and by accumulating reward $r \in J$ is the probability of reaching a direct successor s' within x time-units, $(x \leq \sup(I)$ and $\underline{\rho}(s) \cdot x + \iota(s,s') \leq \sup(J)$, that is $x \leq \sup(K(s,s')))$ multiplied with the probability of reaching a Ψ-state from state s' within the interval $I \ominus x$ and by accumulating reward $r - (\underline{\rho}(s) \cdot x + \iota(s,s'))$.

3. If s satisfies $\Phi \wedge \Psi$ then the path formula is satisfied if the state s is not left for $\inf(K(s))$ time-units. Alternatively, state s should be left before $\inf(K(s))$ time-units have elapsed in which case the probability is defined as in case 2.

Whilst the system of equations described above completely characterizes the until formula some trivial cases still exist. If $J = [0, \infty)$ and $I = [0, t]$ or if $J = [0, \infty)$ and $I = [0, \infty)$ for $t \in \mathbb{R}_{\geq 0}$ then the characterization of the until formula coincides with the characterization in [4].

5. Model Checking CSRL

For model-checking MRMs, given a system modeled as an MRM and a formula expressed in CSRL, it has to be ascertained whether the formula is valid for the MRM. Initially the formula is parsed and its sub-formulas are determined. Then a post-order recursive traversal of the parse-tree is carried out to determine the values of the sub-formulas. The value of a formula Φ is the set of states that satisfy the formula, $Sat(\Phi)$. For most of the measures the algorithms are fairly straightforward given the characterization of CSRL and the previous work in [2, 3, 9]. The primary difference with the incorporation of impulse rewards is in the evaluation of the until formulas.

For analyzing until formulas the numerical analysis of equation (3) is susceptible to numerical instability. This section presents methods to transform the MRM such that alternative algorithms for the analysis of MRMs can be used that would avoid computing the integral.

Definition 5 *For $\mathcal{M}$ and CSRL state formula Φ, let $\mathcal{M}[\Phi] = ((S, \boldsymbol{R}', Label), \underline{\rho}', \iota')$ be the MRM with:*

$$\boldsymbol{R}'(s,s') = \begin{cases} \boldsymbol{R}(s,s') \text{ if } s \nvDash \Phi \\ 0 \text{ otherwise} \end{cases},$$
$$\underline{\rho}'(s) = \begin{cases} \underline{\rho}(s) \text{ if } s \nvDash \Phi \\ 0 \text{ otherwise} \end{cases},$$
$$\iota'(s,s') = \begin{cases} \iota(s,s') \text{ if } s \nvDash \Phi \\ 0 \text{ otherwise} \end{cases},$$

for all $s, s' \in S$.

$\mathcal{M}[\Phi]$ is obtained from $\mathcal{M}$ where all the Φ-states are made absorbing. In the sequel let $P^{\mathcal{M}}(s, \Phi \mathcal{U}_{[0,r]}^{[0,t]} \Psi) = P^{\mathcal{M}}(s, \Phi \mathcal{U}_{\leq r}^{\leq t} \Psi)$.

Proposition 1 *For $\mathcal{M}$, $s \in S$ and $\underline{p}_s(0) = 1$:*

1. $P^{\mathcal{M}}(s, \Phi \mathcal{U}^{\leq t}_{\leq r} \Psi) = P^{\mathcal{M}[\neg \Phi \vee \Psi]}(s, \mathtt{tt} \mathcal{U}^{[t,t]}_{\leq r} \Psi).$

2. $P^{\mathcal{M}}(s, \mathtt{tt} \mathcal{U}^{[t,t]}_{\leq r} \Psi) = \Pr\{Y(t) \leq r, X(t) \vDash \Psi\}.$

Proof *Similar to* [4]. □

Note that $\Pr\{Y(t) \leq r\}$ is the distribution of accumulated reward in an MRM [14].

6. Numerical Methods for $\mathcal{U}^{\leq t}_{\leq r}$

Based on these results appropriate numerical recipes can now be identified which are generalizations of existing algorithms. In the following subsections, generalizations of the discretization and uniformization techniques are presented.

6.1. Discretization

This method is based on the discretization algorithm in [21] for transient performability measures in MRMs with state rewards only. It discretizes both the time interval and the accumulated reward as multiples of the same step size d. d is chosen such that the probability of more than one transition in the MRM in an interval of length d is negligible. Using this discretization, the probability of accumulating at most r reward at time t is given by:

$$\Pr\{Y(t) \leq r\} = \sum_{s \in S} \sum_{k=1}^{k=R} F^T(s,k) \cdot d, \qquad (4)$$

where $R = \frac{r}{d}$, and $T = \frac{t}{d}$.

$F^T(s,k)$ is the probability of being in state s at discretized time T with accumulated discretized reward k. The initial condition $F^1(s,k)$ is given by:

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

For subsequent intervals the following recursive scheme applies:

$$F^{j+1}(s,k) = F^j(s, k - \underline{\rho}(s)) \cdot (1 - E(s) \cdot d)$$
$$+ \sum_{s' \in S} F^j\left(s', k - \underline{\rho}(s') - \frac{\boldsymbol{\iota}(s',s)}{d}\right) \cdot \boldsymbol{R}(s',s) \cdot d.$$

For the MRM to be in state s at the $(j+1)$-st time-instant either the MRM was in state s in the j-th time-instant and remained there for d time-units without traversing a self-loop (the first summand) or it was in state s' and has moved to state s in that period (the second summand). Given that the accumulated reward at the $(j+1)$-st time-instant is k, the accumulated reward in the j-th time-instant is approximated by $k - \underline{\rho}(s)$ in the first summand and $k - \underline{\rho}(s') - \frac{\boldsymbol{\iota}(s',s)}{d}$ in the second summand. Now applying proposition 1:

$$P^{\mathcal{M}}(s, \Phi \mathcal{U}^{\leq t}_{\leq r} \Psi) = \sum_{s' \vDash \Psi} \sum_{k=1}^{k=R} F^T(s', k) \cdot d, \qquad (5)$$

where in $F^1(s,k)$, $s = s_0$. Note that the discretization method requires non-negative rewards. If the method is implemented by using matrices to store F^j and F^{j+1}, then it is also necessary to have integer state-based reward rates.

Both matrices F^j and F^{j+1} need to be stored to obtain the matrix F^{j+1}. In the worst case $2 \cdot |S| \cdot R$ floating point numbers need to be stored. The computational effort reported in [21] is $\mathcal{O}(|S| \cdot t \cdot |(t - r)| \cdot d^{-2})$. However, in each iteration for discretized interval in time and reward every transition in the MRM needs to be explored which in the worst case amounts to $|S|^2$. Consequently, the correct computational cost is $\mathcal{O}(|S|^2 \cdot t \cdot |(t - r)| \cdot d^{-2})$.

6.2. Uniformization

Uniformization is one of the most widely applied strategies to evaluate measures defined over MRMs [19].

Definition 6 *For $\mathcal{M}$ and $\Lambda \geq max_{s \in S}(E(s))$, $\mathcal{M}^u = (S, \boldsymbol{U}, \Lambda, Label, \underline{\rho}, \boldsymbol{\iota})$ is a uniformized process where:*

$$\boldsymbol{U} = \boldsymbol{I_N} + \frac{\boldsymbol{R} - diag(E)}{\Lambda},$$

$\boldsymbol{I_N}$ *is an identity matrix of cardinality $N = |S|$ and $diag(E)$ is the diagonal matrix of E.*

The distribution, $\underline{p}(t)$, of state occupation probability at time t in the underlying CTMC is given by:

$$\underline{p}(t) = \sum_{i=0}^{\infty} \psi(i, \Lambda \cdot t) \cdot \underline{p}(0) \cdot \boldsymbol{U}^i,$$

where

$$\psi(i, \Lambda \cdot t) = \frac{e^{-\Lambda t} \cdot (\Lambda t)^i}{i!}.$$

Using $\mathcal{M}^u$ the distribution of accumulated reward was derived by de Souza e Silva and Gail [19]:

$$\Pr\{Y(t) \leq r\} = \sum_{n=0}^{\infty} \psi(n, \Lambda \cdot t) \cdot \sum_{\forall \underline{k}} \Pr\{\underline{k}|n\} \cdot \Pr\{Y(t) \leq r|n, \underline{k}\}, \qquad (6)$$

$\underline{k} = \langle k_1, k_2, k_3, \cdots, k_{K+1} \rangle$ is a vector where $K + 1$ is the number of distinct state-based reward rates $r_1 > r_2 > r_3 > \cdots > r_{K+1} \geq 0$ in $\mathcal{M}$ and k_l is the number of entrances to a state with reward rate r_l such that $\sum_{l=1}^{K+1} k_l = n + 1$.

If impulse rewards are present in the MRM Qureshi and Sanders [15] generalized equation (6):

$$\Pr\{Y(t) \leq r\} = \sum_{n=0}^{\infty} \psi(n, \Lambda \cdot t) \cdot \sum_{\forall \underline{k}} \sum_{\forall \underline{j}} \Pr\{\underline{k}, \underline{j}|n\}$$
$$\cdot \Pr\{Y(t) \leq r|n, \underline{k}, \underline{j}\}, \qquad (7)$$

$\underline{j} = \langle j_1, j_2, j_3, \cdots, j_J \rangle$ is a vector where J is the number of distinct impulse rewards $i_1 > i_2 > i_3 > \cdots > i_J \geq 0$ in $\mathcal{M}$ and j_l is the number of occurrences of transitions with impulse reward i_l such that $\sum_{l=1}^{J} j_l = n$.

Typically, the computation of (7) is truncated at some depth. Alternatively, the computation may be truncated on the basis of the probability of paths.

Definition 7 *A finite path σ of length n in $\mathcal{M}^u$ is a sequence of states $s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n$ where $s_i \in S$ and $\boldsymbol{U}(s_i, s_{i+1}) > 0$, for $0 \le i < n$. Let $FP^{\mathcal{M}}$ denote the set of all finite paths in $\mathcal{M}^u$ and $FP^{\mathcal{M}}(s)$ its subset starting in $s \in S$.*

Definition 8 *For $\sigma \in FP^{\mathcal{M}}(s)$ in $\mathcal{M}^u$ with $n = |\sigma|$, initial probability distribution $\underline{p}(0)$ and $t \in \mathbb{R}_{\ge 0}$:*

1. $P(\sigma) = \underline{p}_s(0) \cdot \prod_{i=0}^{n-1} \boldsymbol{U}(s_i, s_{i+1})$.
2. $P(\sigma, t) = \psi(n, \Lambda \cdot t) \cdot P(\sigma)$.

Definition 9 *For $0 < w \le 1$, a probability threshold, state $s \in S$ and $t \in \mathbb{R}_{\ge 0}$:*

$$FP_w^{\mathcal{M}}(s, t) = \{\sigma \in FP^{\mathcal{M}}(s) | P(\sigma, t) \ge w\}.$$

If path truncation is employed then [16]:

$$\Pr\{Y(t) \le r\} \approx \sum_{\sigma \in FP_w^{\mathcal{M}}(s,t)} P(\sigma, t) \cdot \Pr\{Y(t) \le r | \sigma\}. \quad (8)$$

Note that several paths may be represented by the same triple $(n, \underline{k}, \underline{j})$. Now applying Proposition 1:

$$P^{\mathcal{M}}(s, \Phi \mathcal{U}_{\le r}^{\le t} \Psi) \approx \sum_{\substack{\sigma \in FP_w^{\mathcal{M}}(s,t), \\ last(\sigma) \vDash \Psi}} P(\sigma, t) \cdot \Pr\{Y(t) \le r | \sigma\}. \quad (9)$$

6.2.1 Error Bounds

For path σ of length n such that $P(\sigma, t) < w$, σ and suffixes of σ are to be discarded. Let $pre(\sigma)$ be the path obtained from σ by removing $last(\sigma)$.

Definition 10 *For $0 < w \le 1$, state $s \in S$ and $t \in \mathbb{R}_{\ge 0}$:*

$$DP_w^{\mathcal{M}}(s, t) = \{\sigma \notin FP_w^{\mathcal{M}}(s, t) | pre(\sigma) \in FP_w^{\mathcal{M}}(s, t)\}.$$

The error by discarding σ and all its suffixes is [16]:

$$E^{\mathcal{M}}(\sigma, t) = P(\sigma) \cdot \left(1 - \sum_{i=0}^{n-1} \psi(i, \Lambda \cdot t)\right).$$

The error-bound for until formulas computed using equation (9) is the error introduced by discarding all paths that are in $DP_w^{\mathcal{M}}(s, t)$:

$$E_{\mathcal{U}}^{\mathcal{M}}(s, w, t) = \sum_{\substack{\sigma \in DP_w^{\mathcal{M}}(s,t) \\ \wedge \ last(\sigma) \vDash (\Phi \ \vee \ \Psi)}} E^{\mathcal{M}}(\sigma, t). \quad (10)$$

The computation of the value of equation (9) has two stages viz, computing path probabilities like in Definition 8 and computing the conditional probability. Both these issues are discussed in the following subsections.

6.2.2 Path Generation and Path Probabilities

The generation of paths can either be performed in breadth-first or depth-first manner. Of particular interest with reference to the evaluation of equation (9) is the depth-first strategy due to the optimization of storage space.

Consider Algorithm 1. If a state that satisfies neither Φ nor Ψ or the probability of path is less than the truncation threshold then further exploration of paths is not necessary. If the present state in a path of length n is a Ψ-state then the path satisfies the path formula and the details regarding the path are stored. For the computation of the Poisson probabilities the well-known Fox-Glynn [8] algorithm can be appplied.

Algorithm 1 Depth First Path Generation
DFPG(integer n, Vector $\underline{k}$, Vector $\underline{j}$, State s, Probability p): void
    if $s \vDash (\neg\Phi \wedge \neg\Psi)$ or $p < w$ then return
    if $s \vDash \Psi$ then store$(n, \underline{k}, \underline{j}, s, p)$
    for all $s, s' \in S$ with $\boldsymbol{U}(s, s') > 0$
        DFPG$(n+1, \underline{k}+\underline{1}_{[s']}, \underline{j}+\underline{1}_{[s,s']}, s', \frac{\Lambda t}{n+1} \cdot p \cdot \boldsymbol{U}(s, s'))$
    end for
end DFPG

$\underline{1}_{[s']}$ is a vector of length $|\underline{k}|$ and the value at index corresponding to $\varrho(s')$ in $\underline{k}$ is 1 and 0 for other indices. $\underline{1}_{[s,s']}$ is a vector of length $|\underline{j}|$ and the value at index corresponding to $\boldsymbol{\iota}(s, s')$ in $\underline{j}$ is 1 and 0 for other indices.

6.2.3 Computing Conditional Probability

Methods for the computation of the conditional probability given by $\Pr\{Y(t) \le r | \sigma\}$ or $\Pr\{Y(t) \le r | n, \underline{k}, \underline{j}\}$ have been presented by de Souza e Silva and Gail [19] and Qureshi and Sanders [15, 16]. The main ideas are briefly presented here.

Let $U_1(t), U_2(t) \cdots, U_n(t)$ be independent, identical and uniformly distributed over $(0, t)$. Now let $U_{(j)}(t)$ be the j-th smallest amongst the random variables, $U_{(0)}(t) = 0$ and $U_{(n+1)}(t) = t$. Then $U_{(1)}(t), U_{(2)}(t) \cdots, U_{(n)}(t)$ are the order statistics of $U_1(t), U_2(t) \cdots, U_n(t)$.

It is known [18] that the joint distribution of the times of n transitions of a Poisson process in $[0, t]$ is identical to the joint distribution of the order statistics of n uniformly distributed random variables over $[0, t]$. Now following a construction similar to that presented in [15] and using some properties of uniform order statistics [17, 18] it follows:

$$\Pr\{Y(t) \le r | n, \underline{k}, \underline{j}\} = \Pr\{\sum_{l=1}^{K}(r_l - r_{l+1}) \cdot U_{(k_1 + \cdots + k_l)}(1) \le r'\},$$
$$(11)$$

where $r' = \frac{r}{t} - r_{K+1} - \frac{1}{t} \cdot \sum_{l=1}^{J} i_l \cdot j_l$.

The RHS of equation (11) is typically evaluated by the methods of Weisberg [22] or Matsunawa [12]. Previous experiments for the evaluation of the distribution of accumulated reward [15, 19] were performed using the algorithm

in [22]. This method however suffers from numerical instability and requires careful implementation [15]. Another experiment [16] using the algorithm in [12] also faced difficulties in the implementation.

A new and numerically stable algorithm is presented in [7] for the computation of equations of the form of equation (11) in a recursive manner. The stability of this algorithm is attributed to calculations being restricted to multiplications of real numbers in $[0, 1]$. This algorithm is adopted here for the computation of the distribution of the linear combination of uniform order statistics. Let $\Omega(r', \underline{k})$ be a function which computes RHS of equation (11) then:

$$\Pr\{Y(t) \leq r | n, \underline{k}, \underline{j}\} = \Omega(r', \underline{k}). \tag{12}$$

Let $d_i = r_i + \cdots + r_n$ and $\mathcal{C} = \{c_1, c_2, \cdots, c_S\}$ be a set of distinct d_i and $\underline{k} = \langle k_1, k_2, \cdots, k_S \rangle$ where k_l is the number of intervals associated to c_l. Now define two sets: $\mathcal{G} = \{l | c_l > r'\}$ and $\mathcal{L} = \{l | c_l \leq r'\}$.

Algorithm 2   [7] Calculating $\Omega(r, \underline{k})$
For $\|\underline{k}_\mathcal{G}\| > 0, \|\underline{k}_\mathcal{L}\| > 0$ choose $i \in \mathcal{G}$ and $j \in \mathcal{L}$ such that both $\underline{k}_i > 0$ and $\underline{k}_j > 0$, then:

$$\Omega(r, \underline{k}) = \frac{c_i - r}{c_i - c_j} \cdot \Omega(r, \underline{k} - \underline{1}_j) + \frac{r - c_j}{c_i - c_j} \cdot \Omega(r, \underline{k} - \underline{1}_i)$$

with initial conditions: for either $\|\underline{k}_\mathcal{G}\| = 0$ or $\|\underline{k}_\mathcal{L}\| = 0$ (but not both):

$$\Omega(r, \underline{k}) = \begin{cases} 1 \text{ if } \|\underline{k}_\mathcal{G}\| = 0, \\ 0 \text{ if } \|\underline{k}_\mathcal{L}\| = 0. \end{cases}$$

Recall that $\underline{1}_j$ is a vector of length $|\underline{k}|$ and the value at index corresponding to j in $\underline{k}$ is 1 and is 0 for other indices.

In summary, for determining $P^\mathcal{M}(s, \Phi \mathcal{U}_{\leq r}^{\leq t} \Psi)$ given a truncation threshold w initially Algorithm 1 is used to obtain all the paths in which the last state satisfies the state formula Ψ. Each of these paths is characterized with a certain combination of $\underline{k}$ and $\underline{j}$ vectors. The path probabilities of paths that are characterized with the same value of these vectors are added. Subsequently, for each path the value of the conditional probability in equation (9) is determined by the use of equation (11) and Algorithm 2. Then $P^\mathcal{M}(s, \Phi \mathcal{U}_{\leq r}^{\leq t} \Psi)$ is determined using equation (9). Finally, the worst-case error-bound is found using equation (10).

6.2.4 Computational Requirements

Let us now consider the worst-case time-complexity of checking whether a state s satisfies an until formula. Assume that the MRM is fully connected and that all 1-step probabilities in the uniformized process are equal. Let N be the number of states and n be the length of paths considered. Computing the path probabilities using the depth-first path generation algorithm (cf. Algorithm 1) requires $\mathcal{O}\left(\frac{N(1-N^{n-1})}{1-N}\right)$ multiplications. Let J be the number of distinct impulse rewards and K be the number of distinct state rewards in the original MRM. Assuming that all paths (of length n) starting in s satisfy the until formula, then

the number of distinct $\underline{k}$-vectors is $\binom{K+n+1}{n+1}$. Similarly, the number of distinct $\underline{j}$-vectors is $\binom{J+n}{n}$.

Given that Algorithm 2 takes maximally $\frac{n^2}{2}$ steps, then the entire computation of the conditional probability for all paths considered requires $\mathcal{O}\left(\binom{K+n+1}{n+1} \times \binom{J+n}{n} \times \frac{n^2}{2}\right)$. The total time-complexity of checking an until formula in a single state is thus given by the sum of the two complexity figures given above. The space complexity is determined by the need to store all $\underline{k}$-vectors, yielding a space complexity of $\mathcal{O}\left(\binom{K+n+1}{n+1}\right)$. There is no need to explicitly store the $\underline{j}$-vectors as only the sum of its elements is needed (cf. equation (11)).

7. Dynamic Power Management

Recently there have been significant advances in the technology and the deployment of mobile devices. Most of these devices are powered by batteries. Hence there has been interest to optimize the usage of battery power in such devices. An important concept in this direction is that of *dynamic power management*.

Dynamic power management (DPM) is a strategy to minimize the number of active components in a system and still provide "optimal" service. Many strategies have been proposed for such a module, for instance, predictive techniques, stochastic control, etc. For a complete discussion on these techniques consider [5].

In this section an example, inspired by [1], in the context of dynamic power management is presented. The goal is to demonstrate the modeling of a system and properties for model-checking MRMs and the comparison of the two numerical methods for until formulas.

7.1. Modeling

The complete setup of the model consists of a server and a mobile device as presented in Figure 2. The server consists of a serving module referred to as the server and a server queue referred to as SQ. The mobile device consists of a network interface (NIC) which can intercept radio signals, a dynamic power management module (DPM), a client queue (CQ) and a client.

The NIC may be in one out of four states viz, *idle*, *busy*, *awaking*, *sleep*. The DPM can receive two signals from SQ viz, *Empty* and *Not-empty*. It can send two signals to the mobile device viz, *Sleep* and *Wake-up*. A simple DPM strategy is: if the present state of the NIC is *idle* and SQ is empty then after an exponentially distributed interval (av.'DPM Shutdown Time') the DPM signals the NIC to go to *sleep* state and if the NIC is in the *sleep* state and if SQ is not empty then after an exponentially distributed interval (av.'DPM Awake Time') the DPM signals the NIC to go to the *awaking* state. Parameters ([1] except PPrate and NICCrate) of the exponential distribution are presented in Table 1. This setup has been modeled in PRISM [11] cf.
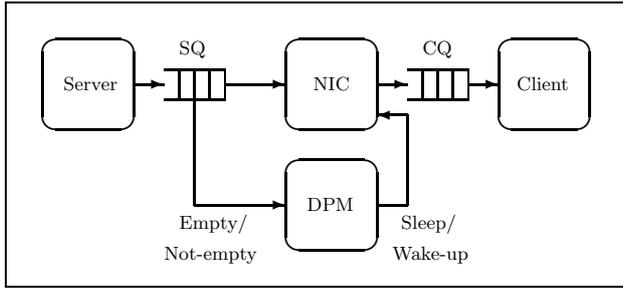
Figure 2. Dynamic Power Management

Figure 3. The model was then exported to a text format. Note that PRISM does not support CSRL model checking.

Table 1. Rates of the DPM Model (ms^{-1})

Transition	Rate
Server Service Rate (SSrate)	0.0149
Packet Propagation Rate (PPrate)	0.025
NIC Checking Rate (NICCrate)	0.02
NIC Awaking Rate (NICArate)	0.0667
Client Rendering Rate (CRrate)	0.0149
DPM Shutdown Rate (DPMSDrate)	0.02
DPM Awake Rate (DPMArate)	0.00125
observationrate	0.002

```
module server
    SQsize: [0..sbuffersize] init 0;
    [sendpacket] SQsize<sbuffersize -> SSrate: SQsize'=SQsize+1;
    [sendpacket] SQsize=sbuffersize -> SSrate: SQsize'=SQsize;
    [deliver] SQsize>0 -> PPrate : SQsize' = SQsize-1;
endmodule

module nic
    NS: [0..3] init 3;
    [deliver] NS=0 -> 1 : NS'= 1;
    [deliver] NS=3 -> 1 : NS'=3;
    [receiveshutdown] SQsize=0&NS=0 -> DPMSDrate : NS'= 3;
    [nicchecking] NS=1->NICCrate : NS'= 0;
    [receiveawake] SQsize>0&NS=3 -> DPMArate : NS'= 2;
    [awaking] NS=2 -> NICArate : NS'= 0;
endmodule

module cbuffer
    CQsize: [0..cbuffersize] init 0;
    [nicchecking] CQsize<cbuffersize -> 1 : CQsize'=CQsize+1;
    [nicchecking] CQsize=cbuffersize -> 1 : CQsize'=CQsize;
    [empcbuff] CQsize>0 -> CRrate : CQsize'=CQsize-1;
endmodule

module final
    [stop] !done -> observationrate: done'=true;
endmodule
```

Figure 3. PRISM Model: DPM

A queue size of 5 for both queues is considered which results in a model of 276 states. There are four kinds of atomic propositions viz, number of frames in SQ (ns), number of frames in CQ (nc), state of the NIC and '$done$'.

The state-based reward rates are rounded-off values from the WaveLAN 1 example. For impulse rewards assume that

it takes 2.5 ms for transitions to take place which require the state of the NIC to be changed with the power consumption of the new state. Then consider the original rewards in the WaveLAN modem and calculate the impulse rewards and round-off the rewards.

7.2. Model Properties

The following properties are considered:

1. $\mathcal{S}_{<0.1}(NotEmpty \wedge sleep)$: A frame loss is possible if the SQ is not empty and if the NIC is in the sleeping state. The upper-bound to the long-run probability of loss is required to be 0.1.

2. $\mathcal{P}_{>0.5}(\mathtt{tt} \, \mathcal{U}_{[0,200]}^{[0,2000]} done)$: This property states that the probability is larger than 0.5 that eventually a $done$-state is reached within 2000 ms such that at most 200 mJ is spent.

7.3. Verification Results

The experiments were conducted on a computer with an Intel P4 3 GHz. processor, 2 GB of RAM running SuSe Linux ver. 9.1. The implementation was performed in Java and executed on Java 2 runtime environment, standard edition.

In the first experiment, both properties were considered and the impact of varying the average 'DPM Awake Time' from 100 ms to 800 ms is studied. The results for state 1 ($ns = 0$, $nc = 0$, $NS = 0$ and $done = false$) as starting state are presented in Figure 4. The results for the until formula are obtained by the use of discretization with $d = 8$. Although the formula results in a boolean value, probabilities are explicitly noted in Figure 4.

In the second experiment the discretization and the uniformization methods are compared. The results for state 1 as starting state and the formula $\mathcal{P}_{>0.5}(\mathtt{tt}\mathcal{U}_{[0,20]}^{[0,t]} done)$ are presented in Figure 5. The results are obtained, varying w, such that the error-bound for uniformization is at most 10^{-3}, 10^{-4} and 10^{-5}. For discretization, d is varied such that the relative error with respect to the results of uniformization is at most 10^{-3}, 10^{-4} and 10^{-5}. The computation time of the two methods by varying the number of states (by varying the queue sizes) in the model is presented in Figure 6. The formula considered is $\mathcal{P}_{>0.5}(\mathtt{tt}\mathcal{U}_{[0,20]}^{[0,50]} done)$. For uniformization the value of w is selected such that error-bound is at most 10^{-3}, 10^{-4} and 10^{-5}. For discretization the plots presented are for values of d such that the relative error with respect to the results of uniformization is at most 10^{-3} and 10^{-4}.

7.4. Discussion of Results

From Figure 4, one can observe that for average 'DPM Awake Time' of 100 ms the probability to eventually reach a $done$-state is 0.44 and the probability of loss of frames is 0.02. On the other hand, for average 'DPM Awake Time'
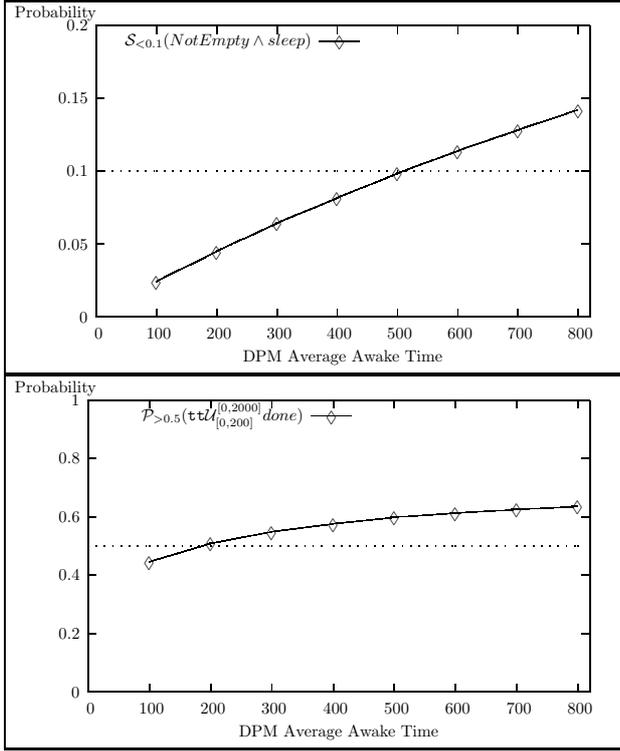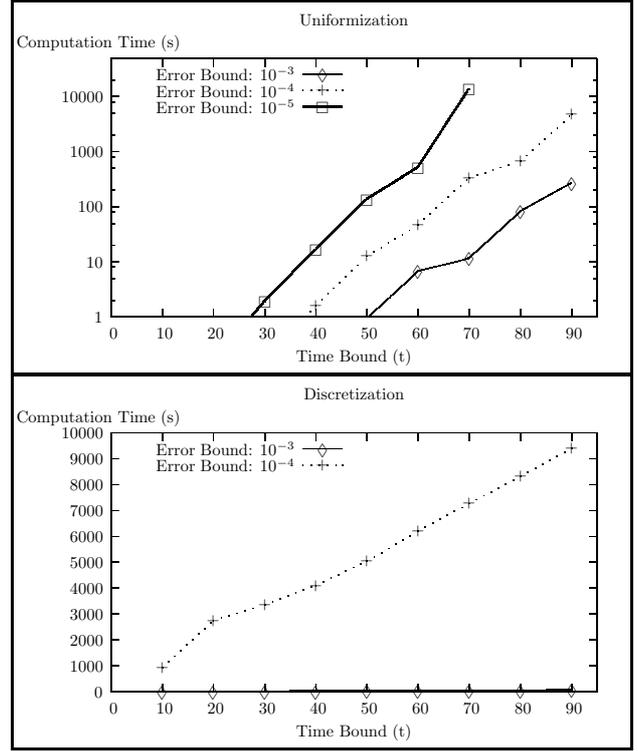
Figure 4. Property 1 and 2 vs. DPM Average Awake Time

Figure 5. Computation Time of Uniformization and Discretization vs. t

of 800 ms the probability to eventually reach a *done*-state is raised to 0.64 and the probability of loss of frames equals 0.15. As expected, the probability of losing a frame increases when increasing the average 'DPM Awake Time'. The probability of eventually reaching a *done*-state also increases with increasing average 'DPM Awake Time' as less energy is consumed in states where the interface is in *sleep* mode.

From Figure 5, it can be observed that the computation time for uniformization method grows exponentially as t is increased. Note that the y-scale is logarithmic for uniformization whereas it is linear for discretization. Inspect, for instance, the computation time for time-bounds 50, 60 and 70 for the three error-bounds. For increasing t, longer paths in the MRM need to be explored, by decreasing the value of w, to maintain a given error-bound. The number of paths to be generated when longer paths are considered grows exponentially hence the rapid increase in the computation time. For the discretization method, the computation time for error-bound 10^{-3} and 10^{-4} only could be obtained. For error-bound 10^{-5}, the computation time is prohibitively large. Nevertheless, it can still be observed that if the error-bound is reduced from 10^{-3} to 10^{-4} then the computation time increases rapidly.

In Figure 6, the computation time of the two methods are plotted against the size of the model, in number of

states, in order to maintain a specified error-bound. In case of uniformization it can be observed that the computation time increases linearly with the size of the state-space for a given error-bound. This could be attributed to the nature of the model and the fact that the observation interval in the formula is that of 50 ms. However, it can be observed that when a lower error-bound is required the computation time increases rapidly. In case of discretization the computation time does not increase linearly with the size of the model. In addition, when a lower error-bound is required the computation time increases very rapidly. The figure includes plots for error-bound of 10^{-3} and 10^{-4} only, since the computation time to maintain an error-bound of 10^{-5} for this model is prohibitively large.

8. Conclusions

In this paper the logic CSRL for MRMs is extended to cover MRMs with state-based reward rate *and* impulse rewards. A logical characterization for all formulas in CSRL is presented. Appropriate numerical methods have been identified to model-check CSRL formulas. An application in the context of 'dynamic power management' has been presented. The general impression about these algorithms is that they are simple to implement and numerically sta-
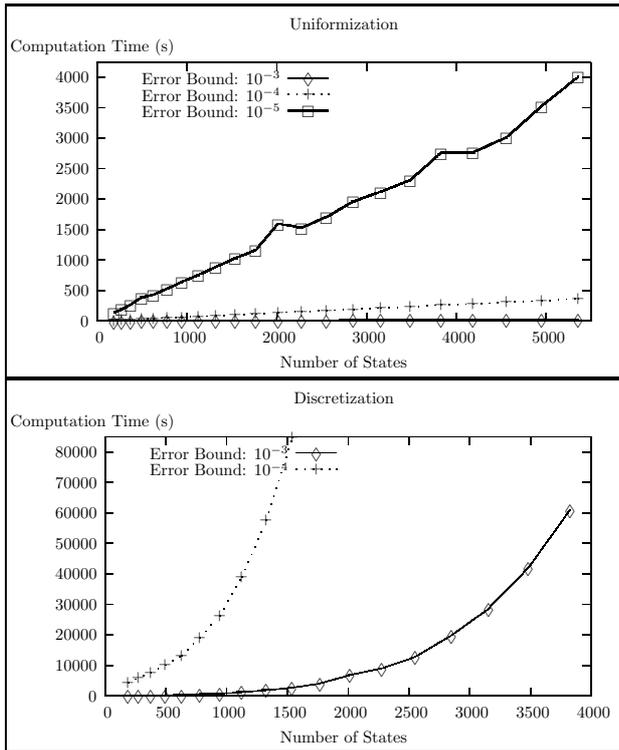
Figure 6. Computation Time of Uniformization and Discretization vs. Number of States

ble. The computational requirements for the until formula are particularly extreme. The uniformization method is applicable when the value of $\Lambda \cdot t$ is small. If this value is increased, the number of paths to be explored grows exponentially if a given error-bound is to be maintained. The discretization method provides no error-bounds. Recall that the error-bounds mentioned in the experiments are relative error-bounds with respect to the values obtained using uniformization. Both methods are applicable only when $J = [0, r]$ and $I = [0, t]$. In the future efficient algorithms for path-probability formulas and algorithms for general reward and time-bounds $J = [r_1, r_2]$ and $I = [t_1, t_2]$ will be considered.

Acknowledgements

References

[1] A. Acquaviva, A. Aldini, M. Bernardo, A. Bogliolo, E. Bonta and E. Lattanzi. *Assessing the Impact of Dynamic Power Management on the Functionality and the Performance of Battery-Powered Appliances.* DSN 2004, IEEE CS Press, pp. 731-740, 2004.

[2] C. Baier, B.R. Haverkort, H. Hermanns and J.-P. Katoen. *On the Logical Characterisation of Performability Properties.* ICALP 2000, LNCS Vol. 1853, Springer, pp. 780-791, 2000.

[3] C. Baier, B.R. Haverkort, H. Hermanns and J.-P. Katoen. *Automated Performance and Dependability Evaluation using Model Checking.* Computer Performance Evaluation, LNCS Vol. 2459, Springer, pp. 261-289, 2002.

[4] C. Baier, B.R. Haverkort, H. Hermanns and J.-P. Katoen. *Model Checking Algorithms for Continuous-Time Markov Chains.* IEEE Transactions on Software Engineering, Vol. 29, No. 6, pp. 524-541, 2003.

[5] L. Benini, A. Bogliolo and G. De Micheli. *A Survey of Design Techniques for System-Level Dynamic Power Management.* IEEE Transactions on VLSI Systems, Vol. 8, No. 3, pp. 299-316, 2000.

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

[7] M.C. Diniz, E. de Souza e Silva and H.R. Gail. *Calculating the Distribution of a Linear Combination of Uniform Order Statistics.* INFORMS Journal on Computing, Vol. 14, No. 2, pp. 124-131, 2002.

[8] B.L. Fox and P.W. Glynn. *Computing Poisson Probabilities.* Comm. ACM, Vol. 31, No. 4, pp. 440-445, 1988.

[9] B.R. Haverkort, L. Cloth, H. Hermanns, J.-P. Katoen and C. Baier. *Model Checking Performability Properties.* DSN 2002, IEEE CS Press, pp. 103-112, 2002.

[10] P.J.M. Havinga and G.J.M. Smit. *Energy-Efficient Wireless Networking for Multimedia Applications.* Wireless Communications and Mobile Computing, Vol. 1, Wiley, pp. 165-184, 2001.

[11] M. Kwiatkowska, G. Norman and D. Parker. *PRISM 2.0: A Tool for Probabilistic Model Checking.* QEST 2004, IEEE CS Press, pp. 322-323, 2004.

[12] T. Matsunawa. *The Exact and Approximate Distributions of Linear Combinations of Selected Order Statistics from Uniform Distributions.* The Annals of the Institute of Statistical Mathematics, Vol. 37, pp. 1-16, 1985.

[13] J.F. Meyer. *On Evaluating Performability of Degradable Computing Systems.* IEEE Transactions on Computers, Vol. C-29, pp. 720-731, 1980.

[14] J.F. Meyer. *Performability Evaluation: Where It is and What Lies Ahead.* Computer Performance and Dependability Symposium, IEEE CS Press, pp. 334-343, 1995.

[15] M.A. Qureshi and W.H. Sanders. *Reward Model Solution Methods with Impulse and Rate Rewards: An Algorithm and Numerical Results.* Performance Evaluation, Vol. 20, No. 4, pp. 413-436, 1994.

[16] M.A. Qureshi and W.H. Sanders. *A New Methodology for Calculating Distributions of Reward Accumulated during a Finite Interval.* Fault-Tolerant Computing Symposium, IEEE CS Press, pp. 116-125, 1996.

[17] S.M. Ross. *Stochastic Processes.* John Wiley & Sons, 1995.

[18] S.M. Ross. *A First Course in Probability.* 6th edition, Prentice Hall, 2001.

[19] E. de Souza e Silva and H.R. Gail. *An Algorithm to Calculate Transient Distributions of Cumulative Reward.* Tech. Rep. CSD940021, UCLA, 1994.

[20] E. de Souza e Silva and H.R. Gail. *An Algorithm to Calculate Transient Distributions of Cumulative Rate and Impulse based Rewards.* Comm. in Statistics - Stochastic Models, Vol 14., No. 3, pp. 509-536, 1998.

[21] H.C. Tijms and R. Veldman. *A Fast Algorithm for the Transient Reward Distribution in Continuous-Time Markov Chains.* Operations Research Letters, Vol. 26, pp. 155-158, 2000.

[22] H. Weisberg. *The Distribution of Linear Combinations of Order Statistics from the Uniform Distribution.* Annals of Institute of Statistics, Vol. 42, No. 2, pp. 704-709, 1971.