Symbolic model checking of stochastic reward nets

Martin Schwarick

Computer Science Department
Brandenburg University of Technology
Cottbus, Germany

Abstract. This paper describes a symbolic model checking approach for the Continuous Stochastic Reward Logic (CSRL) and stochastic reward nets, stochastic Petri nets augmented with rate rewards. CSRL model checking requires the computation of the joint distribution of time and accumulated reward, which is done by Markovian approximation. An implementation is available in the model checker MARCIE. It applies a symbolic on-the-fly computation of the underlying matrix using Interval Decision Diagrams. The result is a multi-threaded model checker which is able to evaluate CSRL formulas for models with millions of states depending on the required accuracy.

1 Introduction

A Continuous-time Markov chain (CTMC) augmented by a reward structure defines a Markov reward model (MRM). The reward structure associates rate rewards (also interpretable as costs) with the states of the CTMC. In [BHHK00] the Continuous Stochastic Reward Logic (CSRL) has been introduced as an expressive and flexible language to specify properties of MRMs. CSRL model checking is based on the computation of the joint distribution of time and accumulated reward, which can be realized in different ways [Clo06]. However, with increasing model size and accuracy most of these techniques fail because of an infeasible runtime. So far there are no efficient tool implementations available. In this paper we present a symbolic model checking approach for CSLR and its implementation in MARCIE [SRH11], a symbolic and multi-threaded on-the-fly model checker for stochastic Petri nets (SPN). We consider MRMs induced by stochastic reward nets, which are SPNs augmented by a reward structure.

2 Markov Reward Models

Coupling a CTMC $C = [S, \mathbf{R}, L, s_0]$, where S is a finite set of states, $\mathbf{R} : S \times S \to \mathbb{R}^+$ the transition rate matrix, $L : S \to AP$ a labelling function mapping to each state a set of atomic propositions from AP and s_0 the initial state, with a reward structure $\varrho : S \to \mathbb{R}^+$ yields a MRM $M = [C, \varrho]$. The reward structure ϱ defines for each state a possibly state-dependent rate reward which will be weighted with the sojourn time the CTMC resides in this state and accumulated over time, as the Markov process evolves. Note that one could also associate impulse rewards with the transitions.

In general a CTMC is specified using some kind of high level formalism, in our case by means of an SPN $N = [P, T, F, I, R, V, s_0]$, where P is a finite set of places, T a finite set of transitions, $F : P \times T \cup T \times P \to \mathbb{N}$ the set of arc weights, $I : P \times T \to \mathbb{N}$ the set of inhibitor arc weights, and $R : P \times T \to \mathbb{N}$ the set of read arc weights. A zero weight means that the arc does not exist. The latter two relations extend the common firing semantics of transitions and allow to define additional enabling conditions; an inhibitor arc defines an upper bound, a read arc a lower bound for the number of tokens. The mapping $s : P \to \mathbb{N}$ associates with each place an amount of tokens and represents a state of the SPN. s_0 is the initial state. The mapping $V : T \to H$, where H is a set of generally state-dependent functions, associates with each transition a rate function h_t from H, defining a negative exponentially distributed delay. If a transition t is enabled in state s, its firing leads to the state s' which is denoted as $s \xrightarrow{t} s'$.

Assuming that a state transition is given by a unique Petri net transition the entry $\mathbf{R}(s, s')$ is defined as:

$$\mathbf{R}(s, s') = \begin{cases} h_t(s) & \text{if } \exists\, t \in T : s \xrightarrow{t} s' \\ 0 & \text{otherwise.} \end{cases}$$

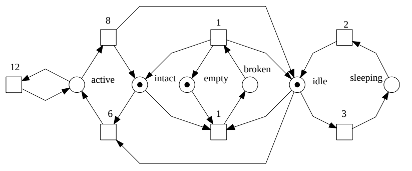
The exit rate $E(s) = \Sigma_{s' \in S} \mathbf{R}(s, s')$ is the sum of the matrix row indexed with s. A state s with $E(s) = 0$ is called an absorbing state, since there is no way to leave it when reached. The probability of the state transition $s \xrightarrow{t} s'$ is $\mathbf{P}(s, s') = h_t(s)/E(s)$. Its probability within τ time units is $\mathbf{P}(s, s') \cdot (1 - e^{-E(s) \cdot \tau})$. The matrix $\mathbf{P}$ describes the so-called embedded discrete-time Markov chain. A path $\sigma = (s_0, \tau_0), \ldots, (s_i, \tau_i), \ldots$ is any finite or infinite sequence with $\mathbf{R}(s_i, s_{i+1}) > 0$, where τ_i is the sojourn time in state s_i. $\sigma[i]$ gives state s_i, $\sigma(\tau)$ the index of the occupied state at time τ, and $|\sigma|$ denotes the length of σ. The set $Path_s$ contains all paths starting in s.

The transient probability $\pi^C(\alpha, s, \tau)$ is the probability to be in state s at time τ starting from a certain probability distribution α, with $\alpha : S \to [0, 1]$ and $\Sigma_{s \in S}\, \alpha(s) = 1$. If we consider a single initial state s_0 we have $\alpha(s_0) = 1$ and write $\pi^C_{s_0}(s, \tau)$ for short.

The computation of transient probabilities (transient analysis) of CTMCs can be realized by uniformization [Ste94]. The basic operation is vector-matrix multiplication which must be done for a certain number of iterations.

The tuple $N^M = [N, \varrho]$ where N is an SPN and ϱ a reward structure is a simple type of stochastic reward net. In contrast to the definition given in [CBC+93] we do not allow impulse rewards, state-dependent arc weights and immediate transitions.

A stochastic process X is often formalized as a collection of random variables $\{X_\tau : \tau \in \mathcal{I}\}$ over a domain Ω and an index set $\mathcal{I}$. In our context, Ω is the finite set of discrete states S and $\mathcal{I} = \mathbb{R}^+$ is the set of real numbers which we



```
RS [ demo ]
 {
  active >0 : 100;
  broken > 0  : 5;
  idle > 0  : 50;
  sleeping >0 : 20;
 }
```

Fig. 1. A stochastic reward net inducing the MRM given in [Clo06]. The reward structure *demo* is defined by four reward items [KNP07]. Each state where e.g. place *active* has at least one token, specified by the guard *active* > 0, earns a reward of 100. If a state satisfies the guard of several reward items, the actual rate reward is given as the sum of the single rewards.

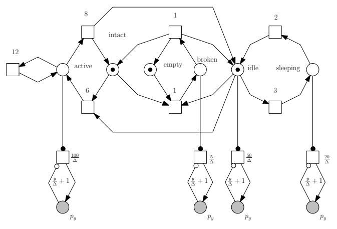


Fig. 2. The bounded approximating SPN $N^{\mathcal{A}}$ for the *stochastic reward net* given in Fig. 1

interpret as time. $X_\tau = s'$ denotes that at time τ the random variable has the value s', $Pr\{X_\tau = s' | X_0 = s\} = Pr_s\{X_\tau = s'\}$ denotes the probability of this situation given that initially its value is s. In the following we may use C and X synonymously.

The reward structure ϱ defines the stochastic process Y with the continuous domain $\mathbb{R}^+$ but the same index set as X. Y_τ denotes the accumulated reward at time τ. For a path σ the accumulated reward at time τ is given as

$$Y_\tau(\sigma) = \sum_{i=0}^{\sigma(\tau)-1} \varrho_{s_i} \cdot \tau_i + \varrho_{s_{\sigma(\tau)}} \cdot (\tau - \sum_{i=0}^{\sigma(\tau)-1} \tau_i).$$

3 Continuous Stochastic Reward Logic

CSRL [BHHK00] is an extension of the popular Computation Tree Logic (CTL) [CES86]. Temporal operators are decorated with a time interval I, and a reward interval J, and path quantifiers are replaced by the probability operator $\mathcal{P}$.
Syntax. The CSRL syntax is defined inductively given state formulas

$$\phi ::= true \mid ap \mid \neg\phi \mid \phi \vee \phi \mid \mathcal{P}_{\bowtie p}[\psi] \ ,$$

and path formulas

$$\psi ::= X^I_J \phi \mid \phi \, U^I_J \phi$$

with $ap \in AP$, $\bowtie \in \{<, \leq, \geq, >\}$, $p \in [0, 1]$, and $I, J \subseteq \mathbb{R}^+$. For convenience we introduce the operators $F^I_J \, \phi \equiv true \, U^I_J \phi$, $\Phi \wedge \Psi \equiv \neg(\neg\Phi \vee \neg\Psi)$ and $\Phi \Rightarrow \Psi \equiv \neg\Phi \vee \Psi$.
Semantics. CSRL state formulas are interpreted over the states and path formulas over the paths of the MRM M. For state formulas we define the satisfaction relation $\models$ as follows

- $s \models ap \Leftrightarrow ap \in L(s)$

- $s \models \neg\Phi \Leftrightarrow s \not\models \Phi$
- $s \models \Phi \vee \Psi \Leftrightarrow s \models \Phi \vee s \models \Psi$
- $s \models \mathcal{P}_{\bowtie p}[\psi] \Leftrightarrow Prob_s^M(\psi) \bowtie p$,

with $Prob_s^M(\psi) = Pr\{Paths_{s,\psi}\}$ as the probability to satisfy the path formula ψ starting in s. The set $Paths_{s,\psi} = \{\sigma \in Paths_s \mid \sigma \models \psi\}$ is measurable [Clo06]. $Sat(\phi)$ denotes the set of states satisfying a state formula ϕ. For path formulas $\models$ is defined as

- $\sigma \models X_J^I \Phi \Leftrightarrow |\sigma| \geq 1 \wedge \tau_0 \in I \wedge \sigma[1] \models \Phi \wedge Y_{\tau_0}(\sigma) \in J$
- $\sigma \models \Phi U_J^I \Psi \Leftrightarrow \exists \tau \in I : \sigma(\tau) \models \Psi \wedge \forall \tau' < \tau : \sigma(\tau') \models \Phi \wedge Y_\tau(\sigma) \in J$

Examples. Survivability: In [Clo06] CSRL is used to define a notion of survivability for critical systems. For a state the survivability can be specified as the state formula

$$disaster \Rightarrow \mathcal{P}_{\leq p}[F_{[0,y]}^{[0,\tau]} recovered],$$

which means that with a given probability the system is able to be recovered from a disaster state within τ time units and recovery costs of at most of y.
Advanced path properties: CRSL formulas can be used to specify additional constraints on paths. Consider the reward structure $\rho^\phi : S \to \{0,1\}$ with

$$\rho^\phi(s) = \begin{cases} 1 & \text{if } s \in Sat(\phi) \\ 0 & \text{otherwise,} \end{cases}$$

which assigns to each ϕ-state a reward of one. The induced process Y_ϕ accumulates for each path the sojourn time in ϕ-states. Thus we can specify for instance the following property

$$\mathcal{P}_{\bowtie p}[\Phi U_{[0,\frac{\tau}{2}]}^{[\tau,\tau]} \Psi],$$

which is only fulfilled on $(\Phi U^{[\tau,\tau]} \Psi])$-paths, if at most the half of the time τ is spent in ϕ-states. Further examples for the usefulness of CSRL can be found in [BCH$^+$10].
Model checking. Model checking is the technique which answers the question, whether a model specification M satisfies a given property ϕ, formally written as $M \models \phi$? In the scenario on hand, M is given as a *stochastic reward net* and ϕ as CSRL formula.

The basic CSRL model checking algorithm is similar to that of CTL [CES86]. Given the formula tree, each sub-formula is evaluated for all states starting with the innermost formulas and ending with the top formula. The most time-consuming part is the computation of the probability $Prob_s^M(\psi)$ for all $s \in S$. In the following we are interested in the complete vector $\underline{Prob}^M(\psi)$. Especially in the case of the U-operator expensive numerical computations have to be done.
Continuous Stochastic Logic (CSL). Before discussing CSRL specific model checking we will sketch the procedure for CSL [ASSB00], the CSRL subset where $J = [0, \infty)$. In this case we omit J for readability. CSL model checking can be reduced to the numerical computation of standard CTMC measures [BHHK03]

and has been implemented in several tools.
Next. The evaluation of the path formula $\psi = X^I \Phi$ means to compute for each state the probability

$$Prob_s^M(X^I \Phi) = (e^{-E(s)\cdot \inf(I)} - e^{-E(s)\cdot \sup(I)}) \cdot \sum_{s' \in \Phi} \mathbf{P}(s, s')$$

that a state transition leading to a ϕ-state occurs within the given time interval [BHHK03]. The evaluation requires a single multiplication of a vector and the matrix $\mathbf{P}$.
Until. The evaluation of this operator can be reduced to transient analysis on a modified Markov chain where several states have been made absorbing depending on the given formula. In this context $M[\phi]$ denotes the Markov chain derived from M making all ϕ-states absorbing.

The following path formulas have to be considered:
1) $\mathbf{\Phi U^{[0,\tau]} \Psi}$: In this case it holds that

$$Prob_s^M(\Phi U^{[0,\tau]} \Psi) = \sum_{s' \in Sat(\Psi)} \pi_s^{M[\neg\Phi \vee \Psi]}(s', \tau)$$

A simple transient analysis is necessary while making $(\neg\Phi \vee \Psi)$-states absorbing.
2) $\mathbf{\Phi U^{[\tau,\tau]} \Psi}$: Paths leading to $(\neg\Phi)$-states are cut and it holds

$$Prob_s^M(\Phi U^{[\tau,\tau]} \Psi) = \sum_{s' \in Sat(\Psi)} \pi_s^{M[\neg\Phi]}(s', \tau)$$

3) $\mathbf{\Phi U^{[\tau,\tau']} \Psi}$: In this case we deal with an inhomogeneous CTMC, as the transition relation $\mathbf{R}$ is time-dependent. Before reaching the lower time bound, it is possible to leave a Ψ-state, provided that it fulfills Φ. After reaching the lower time bound also Ψ-states become absorbing. The computation requires two steps. It holds that
$Prob_s^M(\Phi U^{[\tau,\tau']} \Psi) =$

$$\sum_{s' \in Sat(\Phi)} \left(\pi_s^{M[\neg\Phi]}(s', \tau) \cdot \sum_{s'' \in Sat(\Psi)} \pi_{s'}^{M[\neg\Phi \vee \Psi]}(s'', \tau' - \tau) \right)$$

We ignore the case that $sup(I) = \infty$, as it can not be handled solely by transient analysis. In this case we have to solve a linear system of equations [BHHK03]. It requires a transient analysis for all reachable states to compute $\underline{Prob}^M(\Phi U^I \Psi)$. In [KKNP01] this method has been improved such that one global transient analysis is carried out backwards to compute the probability to fulfill a path formula for all states in one pass; the result is the vector $\underline{Prob}^M(\Phi U^I \Psi)$.

CSRL. Now the evaluation of path formulas requires to consider the given reward interval J.
Next. Assuming that state s earns a positive reward ϱ_s, it is possible to derive

the time interval I_s by dividing all elements of J by ϱ_s. I_s represents the set of sojourn times in s which do not break the given reward constraints. This observation results in

$$Prob_s^M(X_J^I \Phi) = Prob_s^M(X^{I \cap I_s} \Phi).$$

Given that $\varrho_s = 0$ we have to distinguish two cases: if $\inf(J) = 0$, we have only to consider I, otherwise state s can not fulfill the formula.
Until. At the first glance there is no standard technique which can be easily applied to evaluate the full spectrum of possible combinations of time and reward bounds. Contributions in the literature [HCHK02,BCH$^+$10] often consider the special case $\Phi U_{[0,y]}^{[0,\tau]} \Psi$ for which one can apply a similar strategy as for CSL. In this case it is possible to derive a MRM M' by making Ψ- and $\neg(\Phi \wedge \Psi)$-states absorbing. For M' it holds [HCHK02]

$$Prob_s^M(\Phi U_{[0,y]}^{[0,\tau]} \Psi) = Prob_s^{M'}(F_{[0,y]}^{[\tau,\tau]} \Psi).$$

This allows to reduce the problem to the computation of the joint distribution of time and reward, also known as performability,

$$Pr_s\{X_\tau \in S', Y_\tau \in [0,y]\},$$

which is the probability to be at time τ in a state $s \in S'$ (here S' is $Sat(\Psi)$) and having accumulated at most a reward of y. An elaborated consideration of CSRL model checking and related techniques to compute the joint distribution of time and accumulated reward, among them Markovian approximation, can be found in [Clo06].

Markovian approximation. Markovian approximation is a technique to compute the joint distribution by applying standard techniques as transient analysis to a CTMC which approximates the behaviour of the actual MRM. The accumulated reward will be encoded in the discrete states. This requires to replace the continuous accumulation of rewards by a discrete one. The discretized accumulated reward increases stepwise by a fixed value Δ. It requires y/Δ steps to approximately accumulate a reward of y. After n steps the actual accumulated reward lies in the interval $[n \cdot \Delta, (n+1) \cdot \Delta)$, which is the nth reward level. When the reward level is considered as a part of the model, the states of this CTMC are tuples (s,i) where $s \in S$ is a state of the original model and $i \in \mathbb{N}$ the reward level. On level i all state transitions of the original model, given by the transition relation $\mathbf{R}$, are possible and now denoted as $\mathbf{R}_i$. The increase of the accumulated reward, the steps to the next level, must be realized by an additional state transition relation $\mathbf{R}_{i,i+1}$ which maps the reward function of the MRM to stochastic rates. During one time unit the original MRM accumulates a reward of ϱ_s in the state s. In the derived CTMC $C^{\mathcal{A}}$, we must observe $\frac{\varrho_s}{\Delta}$ transitions per time unit in state s which increase the discretized reward level. For each state (s,i) there must be a state transition to the state $(s,i+1)$ with rate $\frac{\varrho_s}{\Delta}$.

The state space is infinite but countable as the reward growths monotonically without an upper bound. The resulting CTMC's rate matrix $\mathbf{R}^{\mathcal{A}}$ has the following structure and represents a special type of a so-called quasi birth-death process (QBD) [RHC05].

$$\mathbf{R}^{\mathcal{A}} = \begin{pmatrix} \mathbf{R}_0 & \mathbf{R}_{0,1} & 0 & \dots & \dots & \dots\dots \\ 0 & \mathbf{R}_1 & \mathbf{R}_{1,2} & 0 & \dots & \dots\dots \\ \dots & 0 & \ddots & \ddots & 0 & \dots\dots \\ \dots & \dots & 0 & \mathbf{R}_i & \mathbf{R}_{i,i+1} & 0 \;\dots \\ \dots & \dots & \dots & \dots & \ddots & \ddots\dots \end{pmatrix}$$

The probability $Pr_s\{X_\tau \in S', Y_\tau \in (j\Delta, (j+1)\Delta]\}$ to reach at time τ a state from S' while accumulating a reward of $(j\Delta, (j+1)\Delta]$ is approximated by

$$Pr_{(s,0)}\{X_\tau^{\mathcal{A}} \in \{(s', j) | s' \in S'\}\}.$$

Uniformization-based transient analysis and CSL model checking of infinite QBDs is feasible [RHC05] as transient analysis for time t requires only a finite number n of computation steps. This number only depends on the given time t, a constant λ, which should be at least the maximal exit rate, and an error bound ϵ. Thus only a finite number of states will be considered; in fact all the states which are reachable from the initial state by paths with a maximal length of n. In the scenario on hand we can use y instead of t to define the finite subset of reward levels. Let S_i denote the state at, $S_{>i}$ above and $S_{\leq i}$ below or at reward level i. For given y and Δ we define the boundary level $r = \frac{y}{\Delta}$ representing the reward value y. States with a higher reward value than y can be abstracted to the set $S_{>r}$ and states with at most a value of y to the set $S_{\leq r}$. Starting at level 0 one has to cross $r + 1$ levels (including level 0) to exceed the specified reward bound. Thus we can make all states in S_{r+1} absorbing and get a finite CTMC. Alternatively we can use the CSL model checking algorithm to justify that only this finite subset of levels has to be considered. Given the infinite CTMC $C^{\mathcal{A}}$ the joint distribution $Pr_s\{X_\tau \in S', Y_\tau \in [0, y]\}$ is approximated by $Pr_s\{X_\tau^{\mathcal{A}} \in S' \cap S_{\leq r}\}$ whereby only the probability of states $s \in S_0$ is of interest. For all $s \in S_0$ we have to compute $Prob_s^{C^{\mathcal{A}}}(S_{\leq r} U^{[\tau,\tau]} S' \cap S_{\leq r})$, the probability for all states at reward level 0 to be at t time in a state from S' while residing only in $S_{\leq r}$-states. As the algorithm makes all $(\neg\Phi)$-states, all $(S_{>r})$-states absorbing, $(S_{>r+1})$-states will not be reachable in the resulting CTMC $C^{\mathcal{A}}$. **Stochastic Petri net interpretation.** To approximately compute the joint distribution we will perform transient analysis on the derived CTMC $C^{\mathcal{A}}$ using the fast method suggested in [KKNP01]. As we achieve this using an on-the-fly computation of the rate matrix, a high-level description of it by means of a stochastic Petri net is required. Thus we will extend the existing net by an additional place, whose markings represent the reward levels. The number of tokens on this additional place p_y increases only by the firing of the transitions which define the relation $\mathbf{R}_{i,i+1}$.

We derive this set of transitions T^{ϱ} directly from the reward structure definition. Similar to [KNP07] we define a reward structure as a set of reward items, as can be seen in Figure 1. A reward item consists of a guard g and a reward function r. The guard g is given as an interval logic function defining a set of states. *Reduced Ordered Interval Decision Diagrams* (IDD) are a canonical representation for interval logic functions [ST10] and allow easily to extract the set of reward transitions, as it is illustrated in [Sch12]. These transitions induce the SPN $N^{\varrho} = [P, T^{\varrho}, \emptyset, I^{\varrho}, R^{\varrho}, V^{\varrho}, \emptyset]$.

So far the transitions T^{ϱ} only define a reward function for certain sets of states, but do not affect the set of reachable states. Two steps remain:

1. The firing of these transitions must increase the number of tokens on p_y.
2. The rate reward function must be transformed to a stochastic rate. For all transitions $t \in T^{\varrho}$ the new function must be changed to h_t/Δ.

Given the original *stochastic reward net* as the SPN $N = [P, T, F, I, R, V, s_0]$, the reward structure ϱ in turn as the net $N^{\varrho} = [P, T^{\varrho}, \emptyset, I^{\varrho}, R^{\varrho}, V^{\varrho}, \emptyset]$ and a specified reward step Δ we can define the SPN $N^{\mathcal{A}} = [P^{\mathcal{A}}, T^{\mathcal{A}}, F^{\mathcal{A}}, I^{\mathcal{A}}, R^{\mathcal{A}}, V^{\mathcal{A}}, s_0^{\mathcal{A}}]$ with $P^{\mathcal{A}} = P \cup \{p_y\}$, $T^{\mathcal{A}} = T \cup T^{\varrho}$, $F^{\mathcal{A}} = F \cup \{((t, p_y), 1) | t \in T_{\varrho}\}$, $I^{\mathcal{A}} = I \cup I^{\varrho} \cup \{(p_y, t, \frac{y}{\Delta} + 1) | t \in T^{\varrho}\}$, $R^{\mathcal{A}} = R \cup R^{\varrho}$, $V^{\mathcal{A}} = V \cup \{(t, h_t/\Delta) | (t, h_t) \in V^{\varrho}\}$ and $s_0^{\mathcal{A}} = s_0$. The set of inhibitor arc weights is already prepared to define the finite version of $C^{\mathcal{A}}$. Figure 2 shows the stochastic Petri net derived from the *stochastic reward net* in Figure 1.

CSRL model checking based on CSL. We argued with the CSL model checking algorithm to derive a finite subset of the approximating CTMC $C^{\mathcal{A}}$. We have now specified the related high level description $N^{\mathcal{A}}$. This idea also suggests to realize CSRL model checking on top of an existing CSL model checker using the SPN $N^{\mathcal{A}}$ and the CSL formulas given in Table 1. The constraints concerning the discretized accumulated reward given as interval J are just encoded into state formulas. However, this explicit approach has several drawbacks. First, we will have in most cases a significant higher memory consumption as even a symbolic CSL model checker stores the exit rates explicitly and has to allocate a vector in the size of the state space of $C^{\mathcal{A}}$. To illustrate this problem, Table 6 shows the state space for different accuracy levels for the scalable FMS system [CT93]. Second, when nesting CSRL formulas or checking a set of formulas this would require to reconstruct $C^{\mathcal{A}}$ as we will consider different reward bounds. Moreover some combinations of time and reward intervals I and J allow some optimization in terms of the necessary reward levels or the time bounds. E.g. if $\varrho_{min} > 0$, which means that the reward function is total, we can possibly decrease the lower time bound. If $\frac{y}{\varrho_{min}} > \tau$ then τ can be replaced by $\frac{y}{\varrho_{min}}$ which may reduce the number of required uniformization steps. If $\tau' \cdot \varrho_{max} < y'$ then y' can be replaced by $\varrho_{max} \cdot \tau'$ which may reduce the number of reward levels. Further we can check whether the given bounds are consistent: e.g. if $\tau' \cdot \varrho_{max} < y$ or $\tau \cdot \varrho_{min} > y'$ no path will fulfill the CSRL formula.

For these reasons we propose to automatically

1. generate the approximating SPN from the actual MRM specification as described above,
2. encode the actual CSRL formula implicitly for the CSL model checking algorithms by adapting the transition relation.

We offer a reference implementation of this approach in the model checker MARCIE. In the next section we will sketch some implementation issues.

4 Implementation in MARCIE

MARCIE [SRH11] is a symbolic model checker for stochastic Petri nets offering besides symbolic state space analysis (e.g. CTL model checking) simulative and numerical engines for quantitative analysis as model checking of CSL and the reward extensions defined in [KNP07]. Its symbolic model checking engines are based on Interval Decision Diagrams which are used to encode state sets of bounded Petri nets [ST10]. In contrast to other approaches, numerical analysis is based on a multi-threaded on-the-fly computation of the rate matrix. This means that the matrix is not encoded using dedicated date structures as sparse matrices or multi terminal decision diagrams. The computation of the matrix entries is realized by an emulation of the firing of Petri net transitions given the IDD-based state space encoding. This technique does not require structured models, is not sensitive concerning the number of distinct matrix entries, and is often able to outperform other symbolic approaches aiming on the numerical analysis of huge CTMCs [SH09,HRSS10,SRH11]. However, the on-the-fly computation requires a high-level description of the CTMC in terms of a stochastic Petri net. Thus the discussed Markovian approximation is the method of choice for the implementation of our CSRL model checker.

CSRL	CSL
$\Phi U^I_{[y,y]}\Psi$	$\Phi U^I(\Psi \wedge p_y = \frac{y}{\Delta})$
$\Phi U^I_{[0,y]}\Psi$	$\Phi U^I(\Psi \wedge p_y \leq \frac{y}{\Delta})$
$\Phi U^I_{[y,y']}\Psi$	$\Phi U^I(\Psi \wedge p_y > \frac{y}{\Delta} \wedge p_y \leq \frac{y'}{\Delta})$

Table 1. CSL representation of several CSRL formulas for the approximating SPN $N^{\mathcal{A}}$.

J	$\underline{v}[i] = 1 \Leftrightarrow s_{(i \mod \lvert S \rvert)} \in Sat(\Psi)$ and
$[y,y]$	$\lvert S\rvert \cdot (l-2) \leq i < \lvert S\rvert \cdot (l-1)$
$[0,y]$	$0 \leq i < \lvert S\rvert \cdot (l-1)$
$[y,y']$	$\lvert S\rvert \cdot \lfloor \frac{y}{\Delta} \rfloor < i \leq \lvert S\rvert \cdot (l-1)$

Table 2. Initial values for the computation vectors depending on the CRSL formula.

In the following we exclude the case $sup(I) \equiv \infty$ as we concentrate on the cases where we can apply transient analysis. However, our tool supports full CSRL which includes time and reward intervals with unspecified upper bounds. Then we use the algorithm for the un-timed CSL until-operator, which is based on solving linear systems of equations using iterative methods as Jacobi or Gauss-Seidel, see [BHHK03].

The implementation which extends the existing symbolic engine considers the place p_y only implicitly. This means that the state space of the model will remain as it is. The entries in the submatrices $\mathbf{R}_{i\,:\,i\geq 1}$ and $\mathbf{R}_{i,i+1\,:\,i\geq 1}$, representing higher reward levels, must be computed additionally. Given $l = \frac{y}{\Delta} + 2$ as the number of required reward levels, including the levels for $[0, \Delta)$ and $[(y+1)\Delta, \infty)$, we have l times to consider the original rate matrix $\mathbf{R}$ and each time we have to shift the related matrix positions (row and column indices) by the size of the state space. In the i-th step, which means to consider reward level $[i \cdot \Delta, (i+1) \cdot \Delta)$, we access the vector at the indices in the interval $[i \cdot |S|, (i+1) \cdot |S|)$ although the on-the-fly matrix extraction computes the indices in $[0, |S|)$. This behaviour is realized by a special extraction policy, which we call *MultiExtraction*, as the rate matrix will be computed in one step several times by shifting the row and column indices. Obviously the used computation vectors have to be of dimension $|S| \cdot l$.

These aspects are not directly affected by the actual reward bounds given in the path formula. However, what does depend on the formula are: the set of states for which transition firing is considered depending on specified step bounds, the initialization of the argument vector, the size of the vector representing the exit rates and the access to it depend on the formula.

Exit rates. If for all states s the exit rates of the states (s, i) are unique for all $0 \leq i \leq l$, we can redirect the access to an exit rate independently of the reward level to the related exit rate of reward level $\mathbf{R}_0$. This requires a translation of the actual index, but the exit rate vector has only dimension $|S|$.

Vector initialization. We use the method described in [KKNP01] which allows to compute $\underline{Prob}^{C^{\mathcal{A}}}(\Phi U \Psi)$ in one pass. Thus the argument vector $\underline{v}$ is initialized in each position with the probability of the related state to reach a Ψ-state, which is 1 for all Ψ-states and 0 otherwise. The adaption to the different cases of reward intervals is given in Table 2.

Restricting the state transition relation. According to Section 3 we encode the level-dependent CSL formulas given in Table 1 implicitly into the transition relation of the net $N^{\mathcal{A}}$. As the enabledness of a transition will now depend not only on the actual system state but also on the current reward level, we decorate transitions of the net internally with step bounds, in which they are allowed to fire. For the specification of the restricted transition relation we have to distinguish basically the following two cases:

1. We are dealing with a homogeneity in the reward dimension ($J = [0, y)$) or consider a single time point $I = [t, t]$. This represents the simple case. Homogeneity in the reward dimension allows to fire all transitions independent of the reward bound and the absorbing states are derived solely from the time interval I. If the latter specifies a time point, the situation is similar, as only $(\neg\Phi)$-states are made absorbing. In this case the vector storing the exit rates can have dimension $|S|$.
2. If $J \neq [0, y]$ and $I \neq [t, t]$ we have to consider an inhomogeneous model possibly in both dimensions. We have now to consider at least the lower reward bound when specifying the absorbing states. In this case the exit rate vector has the size of the implicit state space. Further the inhomogeneity must be encoded into the transition relation as follows:

(a) all transition fire in $(\Phi \wedge \neg\Psi)$-states for all reward levels
(b) reward transitions fire in $(\neg\Phi \wedge \Psi)$-states only until reaching the lower reward level bound, which is $\frac{y}{\Delta}$
(c) all transitions fire in $(\Phi \wedge \Psi)$-states only until reaching the lower reward level bound, which is $\frac{y}{\Delta}$
(d) $(\neg\Phi \wedge \neg\Psi)$-states are absorbing.

5 Related Work

We could provide a long list of analysis and model checking tools for CTMCs or SPN but we are not aware of a CSRL model checker for MRM, stochastic reward nets or a comparable high-level formalism. The probabilistic model checker PRISM [KNP11] offers CSL model checking and allows to augment models with reward structures but only supports their analysis concerning expected instantaneous and cumulative measures. The Markov Reward Model Checker MRMC [KZH$^+$09] extends CSL by the path formulas $X^{[0,\tau]}_{[0,y]}\Phi$ and $\Phi U^{[0,\tau]}_{[0,y]}\Psi$. It deploys the path exploration by Qureshi and Sanders [QS96] and the discretization by Veldman [TV00] for the computation of the joint distribution of time and reward. It also supports impulse rewards. The results presented in [Clo06] base upon a proprietary implementation.

6 Experimental Results

We now present some experimental results for the FSM system [CT93]. All computational experiments were done on a 8×2.26 Mac Pro workstation with 32GB RAM running Cent OS 5.5. We restricted all experiments to a runtime of at most 10 hours. For the experiments we considered a very simple reward structure, which assigns to states satisfying $P3s > 0$ a reward of 1.

We made the following experiments:

1. We compared the implicit implementation for the formula $\mathcal{P}_{\geq 0}[F^{[0,2]}_{[0,1]} P12 > 0]$ with the explicit approach using the SPN N^C and the related level-dependent CSL formula (compare Table 1) concerning runtime and memory consumption. We used the FMS system with $N = 6$ and increased the accuracy by changing the number of reward levels. The experiments were performed single-threaded and multi-threaded with eight threads. Figure 3 shows memory consumption and runtime. To judge the results we used for the explicit approach also the CSL model checker of the PRISM tool.
2. We compared the implicit implementation with the methods supported in MRMC using the CSRL formula $\mathcal{P}_{\geq 0}[F^{[0,2]}_{[0,1]} P12 > 0]$. Table 3 compares the computed probabilities for different accuracy settings. The needed time to check the formula for different accuracy settings and different model sizes is given in Table 5. For comparison, for $N = 3$ and $d = 0.05$ MRMC requires nearly 95 hours using discretization. However compared to the Markovian approximation the memory consumption of MRMC is not noteworthy.

As expected [CH06,Clo06] the Markovian approximation outperforms the methods supported by MRMC. The results in Table 3 show that we can achieve a good approximation with a moderate number of reward levels. Our implementation shows, that on current workstations CSRL model checking is feasible for systems with millions of states. We can see that the direct implementation has a lower memory consumption as it omits the redundant storage of exit rates. Multi-threading reduces the runtime in both cases significantly. Experimental results for further case studies are given in [Sch12].

	MARCIE				MRMC			
	Markovian Approximation				Discretization		Path exploration	
N	$\varDelta = 0.05$	$\varDelta = 0.025$	$\varDelta = 0.0125$	$\varDelta = 0.00625$	$d = 0.05$	$d = 0.025$	$w = 0.0001$	$w = 0.00001$
1	0.0030576	0.0030546	0.0030529	0.0030521	0.0025259	0.0027855	0.00304942	0.00304942
2	0.0088666	0.0088535	0.0088463	0.0088426	0.0074412	0.0081340	–	–
3	0.0142780	0.0142573	0.0142461	0.0142403	–	–	–	–

– means that the experiment was canceled after 10 hours

Table 3. Comparison of the results using MARCIE and MRMC for different levels of accuracy and different initial states.

N	$\mid S_{org} \mid$	$\mid S_{10} \mid$	$\mid S_{100} \mid$	$\mid S_{1000} \mid$
1	54	648	5,508	54,108
3	6,520	78,240	665,040	6,533,040
5	152,712	1,832,544	15,576,624	153,017,424
7	1,639,440	19,673,280	167,222,880	1,642,718,880
9	11,058,190	132,698,280	1,127,935,380	11,080,306,380

Table 4. Size of the reachability graphs for different configurations for the SPN model of the FMS system. The model is scalable concerning the initial marking of the places $P1, P2, P12$ and $P3$. The column $\mid S_{org} \mid$ contains the state space size of the original model. The columns $\mid S_n \mid$ show the number of states for the derived CTMCs, where $n = \frac{y}{\varDelta}$. In our experiments we set y to 1, thus $\mid S_{1000} \mid$ represents an approximation where the rewards increases in each step with $\varDelta_y = 0.001$.

7 Conclusions

In this paper we presented an efficient implementation of a model checker for CSRL and Markov reward models specified by means of stochastic reward nets. We sketched some specific aspects to implement the described techniques in our symbolic and multi-threaded model checker MARCIE. A case study has been used to empirically illustrate its capabilities concerning manageable state space. We are not aware of a comparable implementation of CSRL model checking in any other public tool.

N	$\Delta = 0.05$	$\Delta = 0.025$	$\Delta = 0.0125$	$\Delta = 0.00625$
1	$\ll 1s$	$\ll 1s$	$\ll 1s$	$\ll 1s$
3	$1s$	$2s$	$10s$	$29s$
5	$30s$	$1m7s$	$3m37s$	$12m1s$
7	$6m09s$	$13m27s$	$42m29s$	$162m37s$
9	$40m18s$	$109m32s$	$353m50s$	–

– memory limit of 31 GB reached

Table 5. Total runtime for different values of Δ and different model sizes.

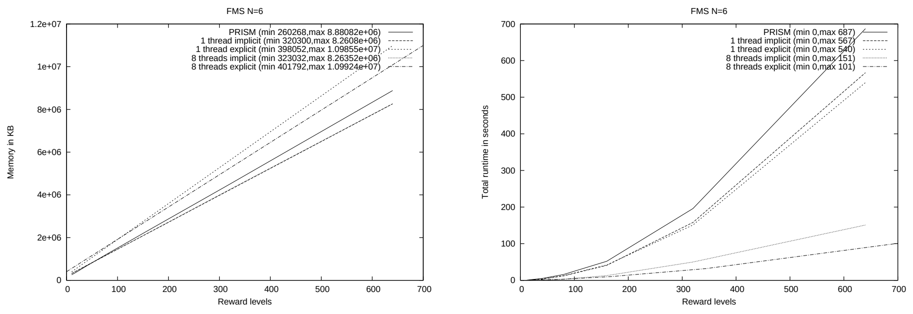


Fig. 3. Comparison of explicit and an implicit realization of the Markovian approximation for the FMS system with $N = 6$ with varying reward levels.

Currently the CSRL model checker is restricted to stochastic reward nets defining rate rewards. In the future we want to incorporate also impulse rewards. The Markovian approximation increases the memory consumption significantly as the computation vectors have to be of size $|S| \cdot (\frac{y}{\Delta} + 2)$. We are convinced that the implicit representation of the underlying QBD allows an efficient application of so-called "out-of-core" methods, which store the complete vector on hard disc while providing fragments to the algorithms which will be used next.

We will explore the use of our tool in further case studies. MARCIE is available for noncommercial use; we provide statically linked binaries for Mac OS X and Linux. The tool, the manual and a benchmark suite can be found on our website
http://www-dssz.informatik.tucottbus.de/DSSZ/Software/Marcie.

References

[ASSB00] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model checking continuous time Markov chains. *ACM Trans. on Computational Logic*, 1(1), 2000.

[BCH$^+$10] Christel Baier, Lucia Cloth, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Performability assessment by model checking of markov reward models. *Formal Methods in System Design*, 36(1):1–36, 2010.

[BHHK00] Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. On the logical characterisation of performability properties. In *Au-*

tomata, Languages and Programming, volume LNCS 1853 of *Lecture Notes in Computer Science*, pages 780–792. Springer-Verlag, 2000.

[BHHK03] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. on Software Engineering*, 29(6), 2003.

[CBC$^+$93] G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi. Automated generation and analysis of markov reward models using stochastic reward nets. *IMA Volumes in Mathematics and its Applications: Linear Algebra, Markov Chains, and Queueing Models / Meyer, C.; Plemmons, R.J*, 48:145–191, 1993.

[CES86] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986.

[CH06] Lucia Cloth and Boudewijn R. Haverkort. Five performability algorithms. a comparison. In *MAM 2006: Markov Anniversary Meeting*, pages 39–54, Raleigh, NC, USA, June 2006. Boson Books.

[Clo06] L. Cloth. *Model Checking Algorithms for Markov Reward Models*. PhD thesis, University of Twente, 2006.

[CT93] G. Ciardo and K. S. Trivedi. A Decomposition Approach for Stochastic Reward Net Models. *Performance Evaluation*, 18(1):37–59, 1993.

[HCHK02] Boudewijn R. Haverkort, Lucia Cloth, Holger Hermanns, and Joost-Pieter Katoen. Model checking performability properties. In *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, DSN '02, pages 103–112, Washington, DC, USA, 2002. IEEE Computer Society.

[HRSS10] M. Heiner, C. Rohr, M. Schwarick, and S. Streif. A comparative study of stochastic analysis techniques. In *Proc. CMSB 2010*, pages 96–106. ACM, 2010.

[KKNP01] J.-P. Katoen, M. Kwiatkowska, G. Norman, and D. Parker. Faster and symbolic CTMC model checking. In L. de Alfaro and S. Gilmore, editors, *Proc. 1st Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM/PROBMIV'01)*, volume 2165 of *LNCS*, pages 23–38. Springer, 2001.

[KNP07] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In M. Bernardo and J. Hillston, editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *LNCS (Tutorial Volume)*, pages 220–270. Springer, 2007.

[KNP11] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[KZH$^+$09] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The Ins and Outs of The Probabilistic Model Checker MRMC. In *Quantitative Evaluation of Systems (QEST)*, pages 167–176. IEEE Computer Society, 2009. www.mrmc-tool.org.

[QS96] M. A. Qureshi and W. H. Sanders. A new methodology for calculating distributions of reward accumulated during a finite interval. In *International Symposium on Fault-Tolerant Computing*, pages 116–125, Sendai, Japan, 1996.

[RHC05] Anne Remke, Boudewijn R. Haverkort, and Lucia Cloth. Model checking infinite-state markov chains. In *TACAS*, pages 237–252, 2005.

[Sch12] M Schwarick. Symbolic model checking of stochastic reward nets. Technical Report 05-12, Brandenburg University of Technology Cottbus, Department of Computer Science, June 2012.

[SH09] M. Schwarick and M. Heiner. CSL model checking of biochemical networks with interval decision diagrams. In *Proc. CMSB 2009*, pages 296–312. LNCS/LNBI 5688, Springer, 2009.

[SRH11] M Schwarick, C Rohr, and M Heiner. MARCIE - Model checking And Reachability analysis done effiCIEntly. In *Proc. 8th QEST*, pages 91 – 100. IEEE CS Press, September 2011.

[ST10] M. Schwarick and A. Tovchigrechko. IDD-based model validation of biochemical networks. *TCS 412*, pages 2884–2908, 2010.

[Ste94] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton Univ. Press, 1994.

[TV00] H. C. Tijms and R. Veldman. A fast algorithm for the transient reward distribution in continuous-time Markov chains. In *Oper. Res. Lett.*, volume 26, pages 155–158, 2000.