Statistical Model Checking of Markov Chains using Perfect Simulation [⋆]

Diana El Rabih and Nihal Pekergin

LACL, University of Paris-Est (Paris 12),
61 avenue Général de Gaulle 94010, Créteil, France
email : delrabih@univ-paris12.fr, nihal.pekergin@univ-paris12.fr

Abstract. In this paper, we propose to perform the statistical model checking of Markov chains by generating sample paths by means of perfect simulation. The model checking of probabilistic models by statistical methods has received increasing attention in the last years since they provide an interesting alternative to the model checking by numerical methods which is poorly scalable with the increasing model size. In the previous statistical model checking works, the unbounded until formula can not be efficiently verified and the steady-state formulas have not been considered due to the burn-in time problem to detect the steady-state. Perfect simulation by coupling in the past is a Monte Carlo method letting to obtain the samples according to the steady-state distribution of the underlying Markov chain. Therefore we propose to check the unbounded until and steady-state formulas for large Markov chains by combining perfect simulation and statistical hypothesis testing.

1 Introduction

Probabilistic model checking is an extension of the formal verification methods for systems exhibiting stochastic behavior. The system model is usually specified as a state transition system, with probabilities attached to transitions, for example Markov chains. A wide range of quantitative performance, reliability, and dependability measures can be specified using temporal logics such as Continuous Stochastic Logic (CSL) defined over Continuous Time Markov Chains (CTMC)[?,?] and Probabilistic Computational Tree Logic (PCTL) defined over Discrete Time Markov Chains (DTMC) [?]. There are two distinct approaches to perform probabilistic model checking: numerical techniques based on computation of transient-state or steady-state distribution of the underlying Markov chain and statistical techniques based on hypothesis testing and on sampling by means of discrete event simulation or by measurement. The numerical approach is highly accurate but it suffers from the state space explosion problem. The statistical approach overcomes this problem but it does not guarantee that the verification result is correct. However it is possible to bound the probability of generating an incorrect answer so they provide probabilistic guarantees of correctness. Hence statistical model checking techniques constitute an interesting

[⋆] This work is supported by a french research project, ANR-06-SETI-002

alternative to numerical techniques for large scale systems. The comparison of numerical and statistical techniques for probabilistic model checking is done in [?].

The statistical approaches for model checking have been received increasing attention in the last years [?,?,?,?,?,?,?]. Younes et al. have proposed and refined the approach based on the hypothesis testing and discrete event simulation but with a focus on time bounded until formula [?,?]. The steady-state formula was not studied before and the unbounded until formula can not be checked efficiently by this approach because of the problem of detecting the steady-state. In [?] Sen et al. has proposed a statistical approach for verifying unbounded until properties by introducing a stopping probability, p_s, which is the probability of terminating the generation of a trajectory after each state transition. In fact, this stopping probability must be extremely small to give correctness guarantees and the accuracy of their verification result would depend on the state space size, making the approach impractical, except for models with small state spaces.

In this paper we propose to perform statistical probabilistic model checking by combining perfect simulation and statistical hypothesis testing in order to check the steady-state and unbounded until formulas. Perfect simulation is an extension of MCMC methods and allows to obtain exact steady-state samples of the underlying Markov chain thus it avoids the burn-in time problem to detect the steady-state. Propp and Wilson has designed the algorithm of coupling from the past to perform perfect simulation [?]. A web page dedicated to this approach is maintained by them (http://research.microsoft.com/en-us/um/people/dbwilson/exact/). As perfect sampler, we use ψ^2 proposed in [?,?], designed for the steady-state evaluation of various monotone queuing networks (http://psi.gforge.inria.fr/website/Psi2-Unix-Website/). This tool permits to simulate stationary distribution or directly a cost function or a reward of large Markov chains by keeping only trajectories issued from the set of minimal and maximal states.

The rest of this paper is organised as follows. In section 2, we present some preliminaries on considered temporal logics CSL and PCTL and on the concept of statistical hypothesis testing. Section 3 is devoted to the perfect simulation. In section 4, we present our contribution for statistical probabilistic model checking using perfect simulation. Section 5 presents the case study with experimental results. Finally, we conclude in section 6.

2 Preliminaries

2.1 Temporal logics for Markov chains

In this subsection we give a brief introduction for the syntax and the semantic for the considered temporal logic operators. We consider essentially until formulas for the verification over execution paths and the steady-state operator for long run behaviours of the underlying model. The stochastic behaviour of the underlying system is described by a labelled Markov chain, $\mathcal{M}$, which may be in discrete or continuous time. These operators are defined in CSL defined over CTMCs [?,?] and in PCTL defined over DTMC [?]. We do not distinguish these cases while transition probabilities are computed from the transition probability matrix for PCTL and from the associated embedded DTMC obtained by uniformisation of the underlying infinitesimal generator for CSL [?].

Let $\mathcal{M}$ take values in a finite set of states S, AP denote the finite set of atomic propositions. $L : S \rightarrow 2^{AP}$ is the labeling function which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions $a \in AP$ those are valid in s. Let p be a probability, I a time interval, and $\bowtie$ a comparison operator: $\bowtie \in \{<, >, \leq, \geq\}$. The syntax is given as follows:

$$\varphi ::= true \mid a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathcal{P}_{\bowtie p}(\varphi_1\, \mathcal{U}^I \varphi_2) \mid \mathcal{P}_{\bowtie p}(\varphi_1\, \mathcal{U}\varphi_2) \mid \mathcal{S}_{\bowtie p}(\varphi)$$

The satisfaction operator is denoted by $\models$, then for all state $s \in S$, $\ s \models true$. Atomic proposition a is satisfied by state s ($s \models a$) iff $a \in L(s)$. The logic operators are obtained using standard logic equivalence rules : $s \models \neg\varphi \;\; iff \;\; s \not\models \varphi$, $s \models \varphi_1 \wedge \phi_2 \;\; iff \;\; s \models \varphi_1 \wedge s \models \varphi_2$. Until formulas are evaluated over the paths initiated from a given initial state s. A state s satisfies $\mathcal{P}_{\bowtie p}(\varphi_1\, \mathcal{U}^I \varphi_2)$, iff the sum of probability measures over paths starting from s, passing through only states satisfying φ_1 and reaching to a state satisfying φ_2 in time interval I meets the bound θ. The until formula without time interval is the time unbounded until which means that $I \in [0, \infty[$. The steady-state operator $\mathcal{S}_{\bowtie p}(\varphi)$ lets us to analyze the long-run behaviour of the system. If the sum of steady-state probabilities of states satisfying φ meets θ, this operator is satisfied. In the case $\mathcal{M}$ is ergodic, the steady-state distribution is independent of the initial state, then this formula is satisfied or not whatever the initial state.

2.2 Statistical model checking

The probabilistic model checking consists in deciding whether the probability that the considered system satisfies the underlying property φ meets a given threshold θ or not. Without lack of generality, we consider the case $\mathcal{P}_{\geq\theta}(\varphi)$ where φ is a path formula. Obviously, this is equivalent to verify $\mathcal{P}_{\leq 1-\theta}(\neg\varphi)$ and $\neg\mathcal{P}_{<\theta}(\varphi)$. Let p be the probability that the system satisfies φ, then this verification problem $\mathcal{P}_{\geq\theta}(\varphi)$ can be formulated as a hypothesis testing: $H : p \geq \theta$ against the alternative hypothesis $K : p < \theta$. For solving hypothesis testing problems with statistical approaches it is not possible to guarantee a correct result but the probability to accept a false hypothesis can be bounded. The strength of the statistical test was determined by two parameters, α and β, where α is a bound on the probability of accepting K when H holds (known as a type I error, or false negative) and β is a bound on the probability of accepting H when K holds (a type II error, or false positive), where $\alpha + \beta \leq 1$. Thus the probability of accepting H can be determined for an hypothesis testing with ideal performance in the sense that the probability of a type I error is exactly α and the probability of a type II error is exactly β. The above formulation is problematic since it is impossible to control two probability errors independently.

These conditions are relaxed by introducing an indifference region $]p_1, p_0[$ of width 2δ, where $p_0 = \theta + \delta$ and $p_1 = \theta - \delta$. Then, instead of testing $H : p \geq \theta$ against $K : p < \theta$, we test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$. The probability of accepting H is therefore at least $1-\alpha$ if $p \geq \theta+\delta$ and at most β if $p \leq \theta-\delta$. For the indifference region $\mid p-\theta \mid < \delta$, the test gives no bound on the probability of accepting false hypothesis, thus we are indifferent whether H or K is accepted.

Suppose that we have generated n samples (simulations), and a sample X_i is a positive sample ($X_i = 1$) if it satisfies φ and negative ($X_i = 0$) otherwise. X_i is a random variable with Bernoulli distribution with parameter p. Thus the probability to obtain a positive sample is p. There are mainly two methods for statistical model checking decision with constraints on error bounds [**?**,**?**,**?**]. One is based on the acceptance sampling with fixed sample size with a given acceptance strength α, β. If $\sum_{i=1}^{n} X_i \geq m$, then H_0 is accepted otherwise H_1 is accepted, where m is the acceptance threshold. The hypothesis H_1 will be accepted with probability $F(m, n, p)$ and the null hypothesis H_0 will be accepted with the probability $1 - F(m, n, p)$, where $F(m, n, p)$ is a binomial distribution: $F(m, n, p) = \sum_{i=1}^{m} C(n, i)p^i(1-p)^{n-i}$ with $C(n, i)$ is the combination of i from n. It is required that the probability of accepting H_1 when H_0 holds is at most α, and the probability of accepting H_0 when H_1 holds is at most β. These constraints can be illustrated as below:

- Pr[H_1 is accepted $\mid$ H_0 is true] $\leq \alpha$ which implies F (m, n, p_0)$\leq \alpha$ (C1)
- Pr[H_0 is accepted $\mid$ H_1 is true] $\leq \beta$ which implies 1- F (m, n, p_1)$\leq \beta$ (C2)

The sample size and acceptance threshold must be chosen under these constraints and for optimal performance n must be minimised. The approximations of n to optimise performance are given in [**?**,**?**].

This second method is based on the sequential probability ratio test in which observations are taken into account ina sequential manner [**?**,**?**]. After making the i^{th} simulation, one computes the following quotient:

$$q_i = \prod_{j=1}^{i} \frac{Pr[X_j = x_j \mid p = p_1]}{Pr[X_j = x_j \mid p = p_0]} = \frac{p_1^{d_i}(1-p_1)^{i-d_i}}{p_0^{d_i}(1-p_0)^{i-d_i}}$$

where d_i=$\sum_{j=1}^{i} X_j$ denoting the number of positive samples. H_0 is accepted if $q_i \leq B$, and H_1 is accepted if $q_i \geq A$. Finding A and B with a given strength α, β is non trivial, in practice A is chosen as (1-β)/α and B as β/(1-α). Then a new test whose strength is (α^*,β^*) is obtained, but such that $\alpha^* + \beta^* \leq \alpha + \beta$, meaning that either $\alpha^* \leq \alpha$ or $\beta^* \leq \beta$. In practice, it is often found that both inequalities hold.

3 Perfect simulation

Let $\{X_i, i \geq 0\}$ be a time-homogeneous DTMC taking values in a finite set S. The dynamic of the chain can be defined by the following stochastic recursive function:

$$X_{n+1} = \eta(X_n, E_n) \quad (1)$$

where X_n is the n^{th} observed state of the system, and $\{E_n\}$ an innovation process, $n \in N$. Clearly, if $\{E_n\}$ are independent and identically distributed then stochastic process $\{X_i, i \geq 0\}$ defined by an initial value X_0 and recursive equations of Eq. **??** is a Markov chain. In the sequel, we consider the notations of discrete event systems (the system is governed by a set of events) thus E_is in Eq. **??** are events $e \in \Sigma$.

Conversely, given a transition probability matrix $\mathbf{P} = (p_{i,j})$, it is possible to find a function η such that Markov chain given by Eq. **??** has $\mathbf{P}$ as transition matrix: $p_{i,j} = \sum_{E_k | \eta(i,E_k)=j} \mathcal{P}(E_k)$. A natural way to construct the transition function η is to consider the inverse of probability distribution function.

Let us remark that this characterization is suitable for discrete event simulation. Practically, the sequence $\{E_i\}$ can be generated by a standard *random function* of programming languages which is uniformly distributed in the interval $[0, 1]$. Sample paths (trajectories) initiated from all possible initial values are generated with the same sequence of random numbers (events) by considering Eq. **??**. If two sample paths reach to the same state, we say that they couple and then their trajectories will be the same. If the sample paths are generated beginning at time t=0 and evolving in the future $t = 1, 2, ...$ then it is called coupling in the future (forward coupling). It has shown that if samples (observations) are constituted from states where all paths initiated at time 0 from all possible initial values are coupled, then this set of samples is not distributed according to the steady-state distribution [**?**].

Prop and Wilson [**?**] have ingeniously overcome this problem by reversing the time (by coming from the past to the present). In their algorithm called coupling from the past (backward coupling), they have shown that if trajectories begin at time $-\infty$ and are generated by coupling in the future, then the set of states where coupling of all trajectories occurred at time 0 (if it exits), are generated according to the steady-state distribution. We do not give here mathematical background but only present an intuitive sketch of proof to be able to explain its application for the verification of path formulas.

In the case E_i are independently and identically distributed random variables, the evolution in n steps from time 0 to n or from time $-n$ to 0 are stochastically equivalent (they have the same distribution)

$$\eta(\cdots\eta(\eta(s_0, E_0), E_1), \cdots), E_{n-1}) =_{st} \eta(\cdots\eta(\eta(s_0, E_{-n+1}), E_{-n+2}), \cdots), E_0) \tag{2}$$

Let suppose that all the trajectories initiated at time $-\tau$ from all initial values are coupled at time 0. Even if these trajectories have initiated earlier in the past, the coupling at time 0 would occur at the same state for identical $E_{-\tau}, E_{\tau+1}, \cdots E_0$. Therefore if the coupling at time 0 occurs, then it can be also considered as the sample state when trajectories have initiated at time $-\infty$. This corresponds indeed to the limiting behavior due to Eq. **??**. Thus we can generate samples according to the steady-state distribution by coupling from the past. We now present the algorithm given in [**?**], which is an adaptation of [**?**] from the implementation point of view. $Generate - event()$ is a function to generate the random events. For the first iteration the trajectories begin at time

Algorithm 1 Backward coupling simulation

```
1: $t \leftarrow 1$; $E[1] \leftarrow Generate-event()$;
2: repeat
3:     $t \leftarrow 2.t$;
4:     for all $x \in S$ do
5:         $y(x) \leftarrow x$; {initialization of trajectories}
6:     end for
7:     for i=t downto t/2+1 do
8:         $E[i] \leftarrow Generate-event()$; {generation of new events from -t/2 +1 to -t}
9:     end for
10:    for i=t downto 1 do
11:        for all $x \in S$ do
12:            $y(x) \leftarrow \eta(y(x), E[i])$; {generation of trajectories through events $E[i]$, }
13:        end for
14:    end for
       {$y(x)$ is the state reached at time 0 for the trajectory issued from x at time -t}
15: until all $y(x)$ are equal; {coupling of trajectories at time 0}
16: return $y(x)$
```

$t = -2$, if there is no coupling at time 0 (line 15), then $t = -4$ for the next iteration (line 4). Therefore if coupling exists, $\tau = -(2^i)$. The optimality of this scheme has been discussed in [**?**]. Let us remark that when one goes more back to the past, we keep already generated events (lines 8-9). Thus Algorithm **??** let us to generate the samples of the limiting behavior (steady-state) of the Markov chain described by η (see Eq. **??**). Obviously, the coupling time depends on η and we refer to the web page of perfect simulation for the large literature on these issues.

3.1 Monotone perfect simulation

It has been shown that if the underlying model has monotone dynamic, it is sufficient to consider only trajectories issued from the minimal and maximal states since all other trajectories are evolved between them [**?**]. Obviously, this leads to a considerable reduction of the simulation time and the storage complexity. It has been shown that many of the discrete event systems have a monotone dynamic [**?**,**?**]. Note that, a system defined by Eq. **??** is said to be monotone if all events $e \in \Sigma$ are monotone where Σ is the set of events. Formally, an event e is said to be monotone, if it preserves the considered partial ordering ($\leq$) on the state space S:

$$x \leq y \Rightarrow \eta(x, e) \leq \eta(y, e) \qquad (3)$$

Since state space is finite, there exists a set M (respectively m) of maximal (respectively minimal) elements. Going from the past when all trajectories issued from M $\cup$ m states couple at time 0 then global coupling occurs. Note that, in the case the model is monotone, the monotone perfect simulation algorithm will be same as Algorithm 1 unless lines 4 and 11. In fact, we consider only the maximal and minimal states as initial values among all states. So lines 4 and 11 will be

4 and 11: for all x$\in$ M $\cup$ m

In the sequel, we will call this algorithm as *Algorithm 1(monotone version).*

3.2 Functional perfect simulation

In many studies, the steady-state distribution is needed to compute steady-state rewards (functional of the steady-state distribution). In this case, it is possible to do functional coupling by generating directly reward values at the steady-state. [**?**]. The backward simulation is stopped when all the trajectories collapse at time 0 on the same reward regardless the coupling state. Since the reward set is generally smaller than the state space, the coupling occurs more quickly in functional perfect simulation. Note that the functional perfect simulation algorithm will be the same as Algorithm 1 unless lines 15,16. In fact, we modify the stopping criteria of this algorithm on a reward value. So lines 15, 16 will be

15: until all $reward(y(x))$ **are equal**

16: return $reward(y(x))$

In the sequel, we will call this algorithm as *Algorithm 1(functional version).*

3.3 Monotone functional perfect simulation

It is possible to do functional perfect simulation for monotone models, if the considered reward function r on the state space is monotone that means

$$\forall(x, y) \in S, \;\; if \;\; x \leq y \;\; then \;\; r(x) \leq r(y) \tag{4}$$

Consequently, if for some states x $\leq$ y, r (x) = r (y), then for all z such that x $\leq$ z $\leq$ y, r (z) = r (x) = r (y). In fact, when rewards are monotone,the improvement is twofold: the number of trajectories that must be coupled is diminished (only that issued maximal and minimal states) and the number of possible outcomes is diminished. Thus this leads to an important reduction of the coupling time and the storage complexity [**?**]. Note that, the monotone functional perfect simulation algorithm will be the same as Algorithm 1 unless lines 4,11,15 and 16. In fact, we consider only the maximal and minimal states as initial values among all states:

4 and 11: for all x$\in$ M $\cup$ m ,

the stopping criteria:

15: until all $reward(y(x))$ **are equal** ,

16: return $reward(y(x))$.

In the sequel, we will call this algorithm as *Algorithm 1(monotone functional version).*

4 Statistical Probabilistic Model Checking using Perfect Simulation

In this section, we present how sample paths are generated and tested for the verification of the steady-state formula ψ=$S_{\bowtie\theta}(\varphi)$ and the unbounded until formula $\phi = P_{\bowtie\theta}(\varphi_1 \; \mathcal{U} \varphi_2)$ through perfect simulation. Sample generation for the time bounded until is straightforward: starting from from an initial state $X_0 = s$, the evolution in time interval I is generated by Eq. **??**. Once sample paths are generated, the statistical hypothesis testing can be applied on these observations (see section **??**) for the decision procedure. Moreover, the case of nested formulas are not considered here and we refer to [**?**,**?**,**?**,**?**]. Let us remark here that the proposed sample path generation is compatible with these proposed methods to check nested formulas either by combining numerical and statistical methods or by computing new precision bounds.

4.1 Steady-state operator

Without lack of generality we consider steady-state formula $\psi = S_{\geq\theta}(\varphi)$. As it has been stated in section **??**, there are different cases to provide perfect samples depending on the monotonicity or not of the underlying model and depending also on the monotonicity or not of the associated reward in the case of monotone model. However the samples are generated essentially from perfect simulation Algorithm 1 with some modifications for each case.

In fact, in order to perform statistical model checking using monotone and/or functional perfect simulation of the steady-state formula $\psi = \mathcal{S}_{\geq\theta}(\varphi)$, we need to test if the obtained steady-state samples satisfy φ or not. Thus we associate the reward $r_\varphi(x)$ to each state $x \in S$ for the given property φ:

$$\begin{aligned} r_\varphi(x) &= 1, \;\; \text{if} \;\; x \models \varphi \\ r_\varphi(x) &= 0, \;\; \text{otherwise} \;\; x \not\models \varphi \end{aligned} \tag{5}$$

Therefore at time 0, we test if the rewards are coupled at reward 0 or 1. In other words, we test if it is a positive or negative sample. There are two cases:

- **Case 1:** the underlying model is not monotone. Thus perfect simulation samples can be generated by *Algorithm 1(functional version)* by considering reward function $r_\varphi(x)$
- **Case 2:** the underlying model is monotone. We keep only trajectories issued from the set of maximal and minimal states. The functional monotone perfect simulation can be applied depending on the monotonicity of r_φ.

The following proposition lets us to know, if we can apply functional perfect simulation to check a given state formula φ.

Proposition 1. *For a given φ, to be able to apply functional perfect simulation, one must show that $\forall x, y \in S^2$ such that $x \leq y$, if $r_\varphi(x) = 1$ then $r_\varphi(y) = 1$.*

Proof We can see from the definition of the reward function r_φ (see Eq. **??**) that it suffices to consider only the case when $r_\varphi(x) = 1$. If in this case $r_\varphi(y) = 1$ is verified then $r_\varphi(x)$ is monotone.

- **Case 2.1:** Case reward r_φ is monotone. In this case, samples are generated by *Algorithm 1 (monotone, functional version).*

– **Case 2.2:** Case reward r_φ is not monotone. then samples are generated through *Algorithm 1(monotone version)*. Once the sample state is obtained, let say x, it is a positive sample if $r_\varphi(x) = 1$ and a negative sample otherwise.

4.2 Unbounded Until formula

The time unbounded until formula $\mathcal{P}_{\geq\theta}(\varphi_1\, \mathcal{U}\varphi_2)$ is checked for an initial state s_0, by considering probabilities over paths starting from s_0. Thus we must generate a sample path starting from s_0, and test if it is a positive sample or not. We modify backward simulation algorithm to generate samples for the verification of unbounded until formula to design Algorithm 2. There are three stopping conditions (conclusions) for the path starting from initial state s_0:

1. the sample path reaches a φ_2 state, this path satisfies the property, we can conclude that it is a positive sample (lines 13 and 14 of Algorithm 2).
2. the sample path reaches a $\neg\varphi_1 \vee \varphi_2$ states, we can conclude that it is a negative sample (see 15 and 16 of Algorithm 2).
3. the sample path visits always $\varphi_1 \wedge \neg\varphi_2$ states, it must be stopped when the steady-state is reached (line 24 of Algorithm 2). If the perfect sample state (steady-state) satisfies φ_2, it is a positive sample, otherwise it is a negative one.

Let us remind that evolution in n steps from the past to present and from the present to the future are stochastically equivalent (see Eq. **??**). Thus if the algorithm is stopped for iteration $t = -2^i$ due to the first two conditions, then it means that in time interval $I = [0, t]$ a φ_2 or $\neg\varphi_1$ state is reached. For the third condition, we need to test if the steady-state is reached or not. This is done by means of the coupling from the past : we must verify if the coupling at time 0 has occurred or not. The sample paths that must be considered for the coupling at time 0 depends on the monotonicity properties of the model. If any of three stopping conditions is verified we continue to go back to the past by increasing time interval $I = [0, 2t]$ (line 3).

In the case when the underlying model is monotone, the considered sample paths are generated by keeping only trajectories issued from the set of maximal and minimal states (see *Algorithm 1(monotone version)*).

Then lines 4 and 18 of Algorithm 2 will be modified as below:
4: for all x$\in$ s_0 $\cup$ **Max** $\cup$ **Min**
18: for all x$\in$ **Max** $\cup$ **Min**

5 Case study: Multistage interconnection delta queueing network

We present as case study the verification of a multistage interconnection queueing network to illustrate the efficiency of the proposed methodology. In telecommunication networks, multistage models are used for modelling switches [**?**]. The

Algorithm 2 Sample generation for unbounded Until

```
1: $t \leftarrow 1$; $E[1] \leftarrow Generate-event()$; $STOP \leftarrow false$;  $Result \leftarrow 0$;
2: repeat
3:   $t \leftarrow 2.t$;
4:   for all $x \in s_0 \cup S$ do
5:     $y(x) \leftarrow x$; {initialisation of trajectories}
6:   end for
7:   for i=t downto t/2+1 do
8:     E[i]=Generate-event() {generate new events from -t/2 +1 to -t}
9:   end for
10:  $i \leftarrow t$;
11:  while $(i \geq 1) \wedge \neg STOP$ do
12:    $y(s_0) \leftarrow \eta(y(x), E[i])$;
13:    if $y(s_0) \models \varphi_2$ then
14:      $STOP \leftarrow true$;  $Result \leftarrow 1$;
15:    else if $y(s_0) \models \neg\varphi_1$ then
16:      $STOP \leftarrow true$;  $Result \leftarrow 0$;
17:    else
18:      for all $x \in S$ do
19:        $y(x) \leftarrow \eta(y(x), E[i])$;
20:      end for
21:    end if
22:    $i \leftarrow i - 1$;
23:  end while
24: until $STOP \vee$ (all $y(x)$ are equal) { $\varphi_2$ or $\neg\varphi_1$ state or the steady-state reached}
25: return $Result$
```

considered model is a delta network with 4 stages and 8 buffers at each stage (see Fig. **??**). Thus the total number of queues (buffer) is $n = 32$. With markovian arrival and service hypothesis, the model can be defined as a CTMC with a state vector $(N_1 N_2 \cdots N_n)$ where N_i is the number of packets in the i^{th} queue. The size of the state space is then $(N_{max}+1)^{32}$, if the maximum queue size is N_{max}. We suppose homogeneous input trafic with arrival rate λ to the first stage and service rate is $\mu = 1$ in each queue. The routing policy is rejection (packets are lost if the queue is full) and at the end of a service the routing probabilities are 1/2 for both buffers in the next stage. There are 64 events (8 external arrivals at the 1^{st} level + 8 departures at the 4^{th} level + 2*8 routing events in first three levels). The monotonicity of these events (with respect to the component-wise order) and so the monotonicity of the model has been shown in [**?**].

The availability and saturation properties at long run of the considered model can be checked through the CSL steady-state formula. State labels are defined through atomic propositions depending on the number of packets in queues. For a given $k \in \{0, \cdots, N_{max}\}$, $a_i(k)$ is true if $N_i \geq k$ and false otherwise. For example, $a_i\ (N_{max})$ is true if the i^{th} buffer is full. The underlying CTMC is labelled with these atomic propositions depending on the considered property. For instance, with formula $\psi = S_{<\theta}(a_i\ (N_{max}))$ we can check the saturation property in the i^{th}

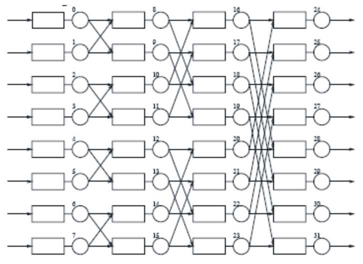

Fig. 1. Interconnection delta network

buffer to see whether the long run saturation probability of the i^{th} buffer is less than θ or not. This lets us also to check the availability property, $S_{>1-\theta}(\neg\ a_i\ (N_{max}))$.

Let φ_1 (resp. φ_2) be the state formula to specify if at least a queue (resp. all queues) at the fourth level is saturated, thus it is defined as the disjunction (resp. conjuction) of atomic propositions $a_i(N_{max}), 25 \leq i \leq 32$. Steady-state formulas $\psi_1{=}S_{<\theta_1}(\varphi_1)$, $\psi_2{=}S_{<\theta_2}(\varphi_2)$ let us to study saturation or availability properties $(S_{>1-\theta_1}(\neg\ \varphi_1),\ S_{>1-\theta_2}(\neg\ \varphi_2))$.

Since the underlying model is monotone, sample paths are generated by applying the monotone perfect simulation *(Algorithm 1 (monotone version))*. In order to apply functional monotone perfect simulation *(Algorithm 1 (functional monotone version))*, one must show that the underlying reward r_φ is monotone. In fact, it follows from proposition **??** that for a state formula φ which can be defined as the conjunction or the disjunction of the $a_i(k)$s for any set of queues, the corresponding reward function (Eq. **??**) is monotone. Therefore the underlying formulas ψ_1 and ψ_2 can be checked by applying functional monotone perfect simulation. Contrary if the state formula is defined from $a_i(k)$ for some queues and $\neg a_i(k)$ for others, we can only apply *(Algorithm 1 (monotone version))*, since the reward is not monotone.

We suppose that $N_{max} = 30$ and $\lambda{=}0.75$ at the first stage. Remark that the state space size is huge and intractable by conventional techniques ($31^{32} \simeq 5.10^{47}$). The perfect samples are generated by using tool Ψ^2 which provides monotone perfect samples [**?**]. In this tool, the number of samples n is an input parameter, and we consider the acceptance threshold $m = \lfloor n\theta + 1 \rfloor$. We compute the strength of the hypothesis testing α, β from condition C1, C2 given in section **??**. We give in the following for different values of θ, δ, the decision for the considered steady-state formulas and values of α and β. In the following tables, $Nsamp$ denotes the sample size, $SMCD_1$: statistical model checking decision for the formula $S_{<\theta}(\varphi_1)$, $SMCD_2$: statistical model checking decision for the formula $S_{<\theta}(\varphi_2)$, α: the computed hypothesis testing Type I error(false negative), β: the computed hypothesis testing Type II error(false positive).

We have verified properties on a very large system (state space 5.10^{47}) having threshold probabilities very close to zero. The time per simulation (observation)

θ	Nsamp	$SMCD_1$	$SMCD_2$	α	β
	10^5	Yes	Yes	0.043	0.010
10^{-3}	10^6	Yes	Yes	$0.183.10^{-8}$	$0.341.10^{-11}$
	10^6	No	Yes	0.043	0.010
10^{-4}	10^7	No	Yes	$0.186.10^{-8}$	$0.347.10^{-11}$
	10^7	No	Yes	0.043	0.010
10^{-5}	10^8	No	Yes	$0.187.10^{-8}$	$0.348.10^{-11}$

Table 1. SMC Decision for $S_{<\theta}(\varphi_1)$ and for $S_{<\theta}(\varphi_2)$ with $\delta=\frac{\theta}{5}$

θ	Nsamp	$SMCD_1$	$SMCD_2$	α	β
	10^5	Yes	Yes	0.210	0.114
10^{-3}	10^6	Yes	Yes	$0.129.10^{-2}$	$0.429.10^{-3}$
	10^6	No	Yes	0.210	0.114
10^{-4}	10^7	No	Yes	$0.129.10^{-2}$	$0.429.10^{-3}$
	10^7	No	Yes	0.210	0.114
10^{-5}	10^8	No	Yes	$0.129.10^{-2}$	$0.429.10^{-3}$

Table 2. SMC Decision for $S_{<\theta}(\varphi_1)$ and for $S_{<\theta}(\varphi_2)$ with $\delta=\frac{\theta}{10}$

θ	Nsamp	$SMCD_1$	$SMCD_2$	α	β
	10^6	Yes	Yes	0.396	0.355
10^{-3}	10^7	Yes	Yes	0.163	0.153
	10^8	Yes	Yes	$0.821.10^{-3}$	$0.737.10^{-3}$
	10^7	No	Yes	0.396	0.355
10^{-4}	10^8	No	Yes	0.163	0.153
	10^9	No	Yes	$0.825.10^{-3}$	$0.740.10^{-3}$
	10^8	No	Yes	0.396	0.355
10^{-5}	10^9	No	Yes	0.163	0.153
	10^{10}	No	Yes	$0.826.10^{-3}$	$0.741.10^{-3}$

Table 3. SMC Decision for $S_{<\theta}(\varphi_1)$ and for $S_{<\theta}(\varphi_2)$with $\delta=\frac{\theta}{100}$

is just a few milli-seconds on a standard PC, and it must be multiplied by the number of samples depending on the required correctness bounds. The monotone perfect sampling with statistical decision techniques provide a really interesting alternative for the probabilistic verification of large systems. Since the discrete event systems have in general monotone dynamics this condition is not very restrictive hypothesis for real world models. Even for non monotone systems, functional perfect simulation is very interesting, because the reward space is only $\{0, 1\}$.

In terms of precision, we obtained α between the order of 10^{-8} and 10^{-2} and we obtained β between the order of 10^{-11} and 10^{-2}. Note that, our computed α and β depend on the sample size $Nsamp$, the term $Nsamp.\theta$ and the indifference region parameter δ. Moreover, the sample size is approximately inversely proportional to the squared width of the indifference region $(2.\delta)$, which explains our consideration of $Nsamp{=}Nsamp * 10^2$ in the table 3 in order to get more precision.

Finally, let note that with $N_{max}{=}30$ and $\lambda = 0.75$ we can see that the mean queue length is relatively small, so saturation could be considered as a rare event, which explains our decision $SMCD_2$ about saturation probability of all queues at fourth stage to be always Yes. Our main goal here is to illustrate the feasibility of the proposed approach for very large systems and rare events. We consider here the steady-state operator since it is more difficult to obtain numerical experiences (sample generation is stopped only when steady-state is reached while there 3 conditions to stop for until formula). The sample paths for until formulas can be generated from Algorithm 2 and then can be combined with statistical decision techniques.

6 Conclusion

We propose to apply perfect simulation to genereate sample paths in order to perform statistical model checking of Markov chains. The statistical model checking by Monte Carlo simulation has been already proposed for time bounded until formula. However the steady-state operator and the unbounded until operator can not be (efficiently) checked by this approach because of the problem of detecting steady-state. Perfect simulation is a relatively recent extension of Monte Carlo simulation which allow to sample steady-state without any bias. We propose to do statistical model checking by combining perfect sampling and hypothesis testing. Therefore it will be possible to check steady-state and unbounded time until formulas of temporal logics for Markov chains. The proposed approach will be integrated to the perfect sampler Ψ^2 [?], to be able to model check large performability models.

Acknowledgment The authors thank to Jean-Marc Vincent for fruitful discussions and for his help on Ψ^2 tool.

References

1. A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, Model Checking Continuous Time Markov Chains, *ACM Trans. on Comp. Logic*, 1(1), pp. 162-170, 2000.
2. C. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen, Model-Checking Algorithms for Continuous-Time Markov Chains, *IEEE Trans. Software Eng.* 29(6), pp. 524-541, 2003.
3. P. Glasserman and D. Yao, *Monotone Structure in Discrete-Event Systems*, John Wiley &Sons, 1994.
4. H. Hansson, B. Jonsson, A logic for reasonning about time and reliability, *Formal Aspects Compt.* 6, pp. 512-535, 1994.
5. T. Hérault, R. Lassaigne, F. Magniette and S. Peyronnet, Approximate probabilistic model checking, *VMCAI, LNCS 2937*, pp 73-84, 2004.
6. H. Hermanns, J.P. Katoen, J. Meyer-Kayser, and M. Siegle, A tool for model-checking Markov chains. *In International Journal on Software Tools for Technology Transfer*, 4(2), pp.153-172, 2003.
7. M. Kwiatkowska, G. Norman, and D. Parker, "Prism : Probabilistic symbolic model checker. In Proceedings of PAPM/PROBMIV 2001 Tools Session, September 2001.
8. S. Keshav, An Engineering approach to computer networking, Addison Wesley, 1997.
9. D. Propp, J.and Wilson, Exact sampling with coupled Markov chains and applications to statistical mechanics. *R*andom Structures and Algorithms, 9(1 and 2), pages 223-252, 1996.
10. K. Sen and M. Viswanathan and G. Agha, Statistical model checking of black-box probabilistic systems, *CAV'04, LNCS 3114*, pp. 202-215, 2004.
11. K. Sen, M. Viswanathan, G. Agha, On Statistical Model Checking of Stochastic Systems, *L*NCS 3576, 2005.
12. J.M. Vincent and C. Marchand, On the exact simulation of functionals of stationary Markov chains, *Linear Algebra and its Applications*, 386:285-310, 2004.
13. J.-M. Vincent and J. Vienne, Perfect simulation of index based routing networks, *Performance Evaluation Review*, 34(2):24-25, 2006.
14. J.-M. Vincent and J. Vienne, PSI2 a Software Tool for the Perfect Simulation of Finite Queuing Networks, *In QEST*, Edinburgh, September 2007.
15. H. L. S. Younes and R. G. Simmons, Probabilistic verification of discrete event systems using acceptance sampling, *CAV02*, p. 223-235, 2002.
16. H.L. Younes, M. Kwiatkowska, G. Norman and D. Parker, Numerical vs. statistical probabilistic model checking, *Software Tools for Technology Transfer*, no. 8(3), 216-228, 2006.
17. H. L. S. Younes, Error Control for Probabilistic Model Checking, *VMCAI 2006*, LNCS 3855, p. 142-156, 2006.
18. H. L. S. Younes and R. G. Simmons, Statistical probabilistic model checking with a focus on time-bounded properties, *Information and Computation 204*, no. 9: 1368-1409, 2006.