Stochastic Model Checking with Stochastic Comparison*

Nihal Pekergin[1,2] and Sana Younès[1]

[1] PRiSM, Université de Versailles Saint-Quentin en Yvelines,
45 Av. des Etats Unis, 78000 Versailles, France
[2] Centre Marin Mersenne, Université Paris 1, 75013 Paris, France
{nih, sayo}@prism.uvsq.fr

Abstract. This paper presents a stochastic comparison based method to check state formulas defined over Discrete Time Markov Reward Models. High-level specifications like stochastic Petri nets, Stochastic Automata Networks, Stochastic Process Algebras have been developed to construct large Markov models. However computation of transient and steady-state distributions are limited to relatively small parameter sizes because of the state space explosion problem. Stochastic comparison technique by which both transient and steady-state bounding distributions can be computed, lets to overcome this problem. On the other hand, bounding techniques are useful in Model Checking, since we check generally formulas to see if they meet some bounds or not. We propose to apply stochastic bounding algorithms to construct bounding distributions and to check formulas through these distributions.

1 Introduction

Model checking has been introduced as an automated technique to verify functional properties of systems expressed in a formal logic like Computational Tree Logic (CTL) [6]. This formalism has been extended with some probabilistic operators to Probabilistic CTL and Continuous Stochastic Logic (CSL)[3]. Stochastic Model Checking is typically based on discrete time or continuous time Markov chains or Markov decision processes. For performance and/or dependability applications, stochastic model checking has been extended to models with some rewards on states and/or transitions in which logic formalisms PRCTL(Probabilistic Reward Computational Tree Logic)[2] and CSRL (Stochastic Reward Logic)[13] are used.

We propose to check the reward based formulas of stochastic models by applying stochastic comparison approach. Indeed, to check these formulas transient or steady-state distribution of the underlying Markov chain must be computed. However the numerical computation of these distributions may be very complex or intractable because of the state space explosion. The stochastic comparison has been shown to be an efficient method to overcome this problem [10]. This

* This work is partially supported by ACI Sécurité SurePath

M. Bravetti et al. (Eds.): EPEW 2005 and WS-FM 2005, LNCS 3670, pp. 109–123, 2005.

method consists in computing bounding distributions rather than the exact distributions by analysing "simpler" bounding chains. Simple bounding models can be constructed by reducing state space size or by imposing some specific structures on bounding chains which let to apply some specific methods like matrix-geometric, product form solution, etc.

The stochastic comparison has been largely used in different areas of applied probability as well as in reliability, performance evaluation, dependability applications [16,18]. There are different stochastic ordering relations and the most known is the strong stochastic ordering ($\leq_{st}$) which yields the comparison of the underlying distributions. Comparison in the sense of this ordering can be established by coupling constructions, by sample-path comparisons or by some analytical methods. However these are generally model oriented techniques. We apply here algorithmic stochastic bounding techniques to construct bounding models in a fully automated manner. Therefore the proposed methodology can be easily integrated to model checkers.

Let us explain the proposed methodology: We are interested in formulas defined as rewards on distributions of a time-homogeneous Discrete Time Markov Chain (DTMC). Thus we need to compute a transient or steady-state distribution. We construct bounding chains by aggregating the states of the original one by means of the algorithm given in [11] which based on the stochastic monotonicity and the comparison of stochastic matrices and the lumpability of Markov chains. The rewards are evaluated through bounding distributions. The state space size may be drastically reduced by aggregation, that will reduce the numerical complexity to compute distributions of the bounding chains. Therefore it is possible to apply numerical methods to compute efficiently the bounding distributions. Obviously, there are some constraints to construct aggregated state space and to order macro-states because of the stochastic ordering constraints and the underlying formula. These issues are discussed in section 4.

The bounding techniques can be applied to efficiently check stochastic models since exact values are not always necessary, and it suffices to show that the underlying formulas meet some bounds. In [7], the bounds on state reachability probabilities of Markov decision processes are computed by abstraction of the underlying model defined on smaller state spaces. If the verification of the considered property cannot be concluded, the abstract model is refined until a verdict to the property can be deduced from the computations. For reward based model checking, generally used rewards such as average population, loss rates, blocking probabilities are defined as non decreasing functions of the transient or steady-state distributions. Thus the stochastic comparison in the sense of the $\leq_{st}$ ordering which is associated to the non decreasing functions can be applied to bound such rewards.

The remaining of the paper is organised as follows: in section 2, we provide a brief introduction of stochastic comparison. In section 3, we present reward based stochastic model checking formalism given in [2]. Section 4 is devoted to the proposed methodology to improve the stochastic reward based model checking. Finally, in section 5, we give some numerical examples to illustrate the proposed methodology.

2 Stochastic Comparison

In this section, we present some preliminaries on the stochastic comparison method and we refer to the books [16,18] for the theoretical issues and different applications of this method.

Definition 1. *Let X and Y be random variables taking values on a totally ordered space S. Then X is said to be less than Y in the strong stochastic sense, ($X \leq_{st} Y$) if and only if $E[f(X)] \leq E[f(Y)]$ for all non decreasing functions $f : S \to R$, whenever the expectations exist.*

Indeed $\leq_{st}$ ordering gives the comparison of the underlying probability distribution functions: $X \leq_{st} Y \leftrightarrow Prob(X > a) \leq Prob(Y > a) \quad \forall a \in S$. Thus it is more probable for Y to take larger values than for X. Since the $\leq_{st}$ ordering yields the comparison of sample-paths, it is also known as sample-path ordering.

We give in the next proposition the $\leq_{st}$ comparison in the case of finite state space $S = \{1, 2, \cdots, n\}$.

Property 1. Let X, Y be random variables taking values on $S = \{1, 2, \cdots, n\}$ and p, q be probability vectors which are respectively denoting distributions of X and Y.

$X \leq_{st} Y \leftrightarrow \sum_i^n p[i] \leq \sum_i^n q[i] \quad \forall i = \{n, n-1, \cdots, 1\}$

The stochastic comparison of random variables has been extended to the comparison of Markov chains.

Definition 2. *Let $\{X(t),\ t > 0\}$ and $\{Y(t),\ t \geq 0\}$ be two DTMC taking values in S. $\{X(t),\ t \geq 0\}$ is said to be less than $\{Y(t),\ t \geq 0\}$ in the strong stochastic sense, that is, $\{X(t),\ t \geq 0\} \leq_{st} \{Y(t),\ t \geq 0\}$ iff $X(t) \leq_{st} Y(t)\ \ \forall t$.*

The comparison of Markov chains yields the comparison of transient distributions at each time, and if the limit distributions exist, we have also the comparison of the steady-state distributions. It is shown that monotonicity and comparability of time-homogeneous DTMC yield sufficient conditions for their stochastic comparison [16].

Theorem 1. *Let $\{X(t), t \geq 0\}$ and $\{Y(t), t \geq 0\}$ be two time-homogeneous DTMC and P and Q be their respective probability transition matrices. $P[i, *]$ indicates row i of matrix P. Then $\{X(t), t > 0\} \leq_{st} \{Y(t), t > 0\}$, if*

- *$X(0) \leq_{st} Y(0)$,*
- *st-monotonicity of at least one of the matrices holds, that is,*

$$either\ P[i, *] \leq_{st} P[i+1, *] \quad or \quad Q[i, *] \leq_{st} Q[i+1, *]$$

- *st-comparability of the matrices holds, that is, $P[i, *] \leq_{st} Q[i, *]\ \ \forall i$.*

In [1] an algorithm based on this theorem is given to construct an optimal st-monotone upper bounding Markov chain. This algorithms takes an irreducible stochastic matrix P as input and returns as output a st-monotone upper bounding matrix, Q, such that, $P \leq_{st} Q$. Indeed, the monotonicity and comparability constraints can be given as in equation 1. Note that inequalities are replaced by equalities to construct optimal bounds.

$$\begin{cases} \sum_{k=j}^{n} Q[1,k] & = \sum_{k=j}^{n} P[1,k] \\ \sum_{k=j}^{n} Q[i+1,k] & = max(\sum_{k=j}^{n} Q[i,k], \sum_{k=j}^{n} P[i+1,k]) \end{cases} \tag{1}$$

In this algorithm that will be called Vincent's algorithm, the construction is done from the last column to the first column and within a column from the first row to the last row. This idea has been extended to devise algorithms to construct st-monotone, bounding stochastic matrices having some specific structures to simplify their numerical analysis [10]. In [5,11] it is shown that ordinary lumpability constraints given in the following Property 2 are consistent with the st-monotonicity. Thus for a given P, it is possible to construct a st-monotone, lumpable, bounding matrix. Let us give the lumpability constraints for discrete time Markov chains.

Property 2. Let Q be the probability transition matrix of an irreducible finite time-homogeneous DTMC, $\mathcal{A} = \{A_1, A_2, \cdots, A_n\}$ be a partition of states. The chain is ordinary lumpable according to partition $\mathcal{A}$, if and only if for all states e and f in the same arbitrary macro state A_i, we have:

$$\sum_{j \in A_k} Q[e,j] = \sum_{j \in A_k} Q[f,j] \quad \forall \ \ macro-state \ \ A_k \in \mathcal{A}$$

The algorithm given in [11] (LIMSUB Algorithm) constructs for a given irreducible, time-homogeneous stochastic matrix P, and partition of states, $\mathcal{A} = \{A_1, \cdots, A_m\}$, a total order relation on $\mathcal{A}$: $A_1 \leq A_2 \leq \cdots \leq A_m$, a st-monotone, lumpable according to $\mathcal{A}$, irreducible upper bounding matrix Q. There are two steps in this algorithm. The first step is based on Vincent's algorithm to satisfy the stochastic monotonicity and comparison constraints (see equations 1) while the second step is to satisfy the lumpability constraints. We explain this algorithm through the following example. Let P be the input matrix and state space be divided into two partitions $A_1 = \{1,2\}$ and $A_2 = \{3,4\}$. Q is the matrix computed from Vincent's algorithm in the first step. Thus Q is $\leq_{st}$ monotone and upper bounding matrix of P. The modified entries are given bolded and supscripts indicate if the probabilities are increased or decreased. In the second step, the sum of probabilities in each macro-state is adjusted to make Q lumpable. Hence Q_{sup} computed from Q is lumpable.

$$P = \left[\begin{array}{cc|cc} 0.2 & 0.2 & 0.2 & 0.4 \\ 0.2 & 0.1 & 0.4 & 0.3 \\ \hline 0.1 & 0.4 & 0.2 & 0.3 \\ 0.1 & 0.1 & 0.4 & 0.4 \end{array}\right] \quad Q = \left[\begin{array}{cc|cc} 0.2 & 0.2 & 0.2 & 0.4 \\ 0.2 & 0.1 & \mathbf{0}.3^{-} & \mathbf{0}.4^{+} \\ \hline 0.1 & \mathbf{0}.2^{-} & \mathbf{0}.3^{+} & \mathbf{0}.4^{+} \\ 0.1 & 0.1 & 0.4 & 0.4 \end{array}\right]$$

$$Q_{sup} = \left[\begin{array}{cc|cc} 0.2 & \mathbf{0}.1^{-} & \mathbf{0}.3^{+} & 0.4 \\ 0.2 & 0.1 & 0.3 & 0.4 \\ \hline 0.1 & \mathbf{0}.1^{-} & \mathbf{0}.4^{+} & 0.4 \\ 0.1 & 0.1 & 0.4 & 0.4 \end{array}\right] \qquad Q_{sup} = \left[\begin{array}{cc} 0.3 & 0.7 \\ 0.2 & 0.8 \end{array}\right]$$

It is also possible to derive lower bounds from the following algorithm by reversing the order of states and then running algorithm LIMSUB on the permuted P. By permuting again the computed upper bounding matrix, we obtain the st-monotone, lower bounding matrix, Q_{inf}. In the sequel, the upper bounding matrix will be denoted by Q_{sup}.

The stochastic comparison approach consists in analysing the bounding matrices Q_{inf} and Q_{sup} to provide bounds on transient distributions and the steady-state distribution of P. Obviously the numerical analysis of the lumpable bounding matrices is much easier than that of P due to the state space reduction.

3 Model Checking with Discrete Time Reward Markov Chains

The underlying system is modelled by a labelled, finite, ergodic (irreducible, aperiodic, positive recurrent) discrete time Markov chain $\mathcal{D} = (S, P, L)$ where S is a finite set of states, $P : S \times S \rightarrow [0, 1]$ is the transition matrix and $L : S \rightarrow 2^{AP}$ is the labelling function which assigns to each state s, the set $L(s)$ of atomic propositions valid in s. AP denotes the finite set of atomic propositions.

For Markov chains, there are two types of state probabilities: transient probabilities where the system is considered at time n. Let $\pi(s, s', n)$ be the probability that the system is in state s' within n steps given the system starts in state s. The steady-state probabilities are the long-run probabilities where the system reaches an equilibrium: $\pi(s, s') = lim_{n\rightarrow\infty}\pi(s, s', n)$ is the steady-state probability of state s'. For ergodic DTMC, $\pi(s, s')$ exists and is independent of the initial state s and that will be noted by $\pi(s')$.

We are interested in the Probabilistic Reward CTL (PRCTL) logic given in [2]. It is indeed the extension of the Probabilistic CTL (PCTL) logic [12] to specify performability measure over Discrete Time Markov Reward Models. In these models a reward (cost) is associated to each state s. Let $\rho : S \rightarrow \mathbf{R}_{\geq 0}$ be the reward assignment function. Every time the system enters (leaves) state s, it incurs reward $\rho(s)$ which can be a constant or a random variable [14]. We give briefly the syntax of the PCTL logic.

Let $n \in \mathbf{N} \cup \{\infty\}$ and I be an interval of real numbers, namely $I \subseteq \mathbf{R}_{\geq 0}$, $p \in [0, 1]$, and $\triangleleft$ a binary comparison operator. The syntax of PRCTL:

$$\phi ::= true \mid a \mid \phi \vee \phi \mid \neg\phi \mid \mathcal{P}_{\triangleleft p}(\phi\, \mathcal{U}_I^J \phi) \mid \mathcal{L}_{\triangleleft p}(\phi)$$

$$\mathcal{I}_I^n(\phi) \mid \mathcal{C}_I^n(\phi) \mid \mathcal{E}_I^n(\phi) \mid \mathcal{E}_I(\phi)$$

The first four operators are classical logic operators, while the fifth and the sixth ones are from the PCTL logic. The path formula $\mathcal{P}_{\triangleleft p}(\phi\, \mathcal{U}_I^J \phi)$ asserts that the

probability for paths starting in s and satisfying $\phi\, \mathcal{U}_I^J \phi$ meets the bound $\lhd p$. The state formula $\mathcal{L}_{\lhd p}(\phi)$ asserts that the steady-state probability to be in ϕ states meets the bound $\lhd p$.

The last four formulas are inspired from performance measures of DTMC with rewards [14] and included in the PRCTL logic [2]. These are all state formulas and defined from transient or steady-state distribution of the underlying Markov chain.

The formula $\mathcal{I}_I^n(\phi)$ is satisfied, if the instantaneous expected reward in ϕ-states (states which satisfy formula ϕ) at the n-th step, starting in state s, meets the bounds of I:

$$\mathcal{I}_I^n(\phi) \text{ is satisfied iff } \sum_{s' \models \phi} \pi(s, s', n)\rho(s') \in I \tag{2}$$

The formula $\mathcal{C}_I^n(\phi)$ is satisfied, if the expected accumulated reward in ϕ-states up to the n-th transition meets the bound of I:

$$\mathcal{C}_I^n(\phi) \text{ is satisfied iff } \sum_{i=0}^{n-1} \sum_{s' \models \phi} \pi(s, s', i)\rho(s') \in I \tag{3}$$

The formula $\mathcal{E}_I^n(\phi)$ is satisfied if the expected reward per unit time in ϕ-states up to the n-th transition meets the bound of I:

$$\mathcal{E}_I^n(\phi) \text{ is satisfied iff } \frac{1}{n} \sum_{i=0}^{n-1} \sum_{s' \models \phi} \pi(s, s', i)\rho(s') \in I \tag{4}$$

The formula $\mathcal{E}_I(\phi)$ is the long-run expected reward per unit-time (reward rate) for ϕ-states which is the limiting case of $\mathcal{E}_I^n(\phi)$. ($\mathcal{E}_I(\phi)$=$\lim_{n\to\infty} \mathcal{E}_I^n(\phi)$). If the steady-state exists, $\mathcal{E}_I(\phi)$ is satisfied if:

$$\mathcal{E}_I(\phi) \text{ is satisfied iff } \sum_{s' \models \phi} \pi(s')\rho(s') \in I \tag{5}$$

The other state operator of the PCTL logic, $\mathcal{L}_{\lhd p}(\phi)$ can be also defined by means of the steady-state distribution and it is satisfied if:

$$\mathcal{L}_{\lhd p}(\phi) \text{ is satisfied iff } \sum_{s' \models \phi} \pi(s') \lhd p \tag{6}$$

4 Model Checking by Stochastic Comparison

In this section we explain the proposed methodology to check reward based stochastic models by applying stochastic comparison method. This methodology is composed of three main steps and the treatment in each step depends on the considered formula ϕ that will be checked:

1. Partition of the state space and ordering of macro-states.
2. Construction of the bounding chains through algorithm LIMSUB (see section 2) and computing transient or steady-state distribution as a function of the the considered formula (see section 3).
3. Checking the underlying formula.

4.1 State Space Partition

We divide state space S into two subset S_{no} and S_{yes} such that S_{yes} contains ϕ-states ie. $S_{yes} = \{s \in S \mid s \models \phi\}$ and S_{no} contains states which do not verify ϕ ie. $S_{no} = \{s \in S \mid s \not\models \phi\}$.

We order state space to have S_{no} followed by S_{yes}. We are especially interested in S_{yes} since the rewards are computed over these states. In performance and dependability applications the size of S_{yes} is small compared to the size of S_{no}. In general the states of S_{no} are aggregated into macro-states to reduce the state space size. However there is no constraint on the ordering of these macro-states and on the rewards assigned to them. But if states of S_{yes} are aggregated, because of the $\leq_{st}$ stochastic ordering, some constraints on the macro-state ordering and on the rewards must be satisfied (figure 1).

Suppose that S_{yes} is divided into k macro-states: $S_{yes} = \{A_1, A_2, \ldots, A_k\}$. The rewards for macro-states are defined as follows:

- to compute upper bounds, $\rho_{sup}(A_i) = \max\{\rho(s), s \in A_i\}$.
- to compute lower bounds, $\rho_{min}(A_i) = \min\{\rho(s), s \in A_i\}$.

Since $\leq_{st}$ stochastic ordering is associated to increasing reward functions (see definition 1), macro-states are ordered according to the increasing rewards. Let us remark that the macro-state ordering may be different for the upper and the lower bounding computations. For the sake of simplicity, we suppose in the sequel that macro-states are ordered as follows: $\rho(A_1) \leq \rho(A_2) \leq \cdots \leq \rho(A_k)$ $\rho \in \{\rho_{sup}, \rho_{inf}\}$.

The atomic propositions of macro-states must be also updated: for each macro-state A_i, $L(A_i) = \cap_{s \in A_i} L(s)$. Let us emphasise that the accuracy of the bounds depends on the aggregation procedure: if the number of macro-states is small, bounds will be less accurate. By increasing the number of macro-states the accuracy can be improved with detriment of the numerical complexity. Thus a trade-off between the accuracy of results and the computation efficiency must be found.

4.2 Construction and Computing of Bounding Chains

Once the state space is partitioned, the bounding chains are constructed through algorithm LIMSUB given in section 2. Recall that the input parameters are the stochastic matrix of the underlying model, P and the partition $\mathcal{A}$=$\{A_1, A_2 \cdots A_m\}$. The upper bounding matrix Q_{sup} is returned as the output of algorithm LIMSUB. The lower bounding matrix Q_{inf} can be constructed by reversing the

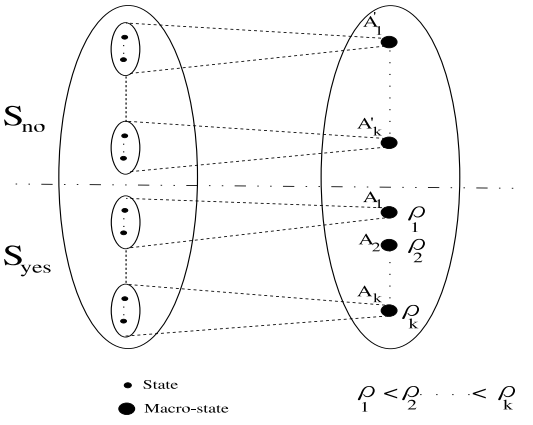

Fig. 1. Partition and ordering of state space

order of macro-states. In this case the inputs of algorithm LIMSUB are P, and the partition $\mathcal{A} = \{A_m, A_{m-1} \cdots A_1\}$. By permutating the output matrix, we obtain the lower bounding matrix, Q_{inf}.

Transient and steady-state distributions of the bounding chains Q_{inf} and Q_{sup} can be efficiently computed by applying conventional numerical methods. We refer to Stewart's book [19] for numerical methods to compute distributions of Markov chains.

We can derive the inequalities on state probabilities. By construction, Q_{sup} and Q_{inf} are st-monotone and $Q_{inf} \leq_{st} P \leq_{st} Q_{sup}$. Thus it follows from theorem 1 that transient distributions (and the steady-state distribution if it exists) of the underlying Markov chains are $\leq_{st}$ comparable. Let $\pi_{bound}(s, A_j, n)$ be the probability that the bounding Markov chain is in macro-state A_j at step n beginning in state s at time 0 and $\pi_{bound}(A_j)$ be the probability that the bounding Markov chain is in macro-state A_j at the steady-state. The following inequalities follow from property 1 of the $\leq_{st}$ ordering.

Property 3. – *Transient state probability bounds:*

$$\sum_{j=i}^{m} \pi_{inf}(s, A_j, n) \leq \sum_{j=i}^{m} \sum_{s' \in A_j} \pi(s, s', n) \leq \sum_{j=i}^{m} \pi_{sup}(s, A_j, n) \quad \forall i \in \{m, m-1, \cdots 1\} \tag{7}$$

– *Steady-state state probability bounds:*

$$\sum_{j=i}^{m} \pi_{inf}(A_j) \leq \sum_{j=i}^{m} \sum_{s' \in A_j} \pi(s') \leq \sum_{j=i}^{m} \pi_{sup}(A_j) \quad \forall i \in \{m, m-1, \cdots 1\} \tag{8}$$

4.3 Checking of State Formulas

In this section, we show how state formulas of the PRCTL logic can be checked through the bounding distributions. Remember that for a given state formula ϕ,

the state space is partitioned and ordered such that the states satisfying ϕ, S_{yes} are greater (after) than that of S_{no}. Let suppose that $\mathcal{A} = \{A_1, \cdots, A_{k-1}, A_k, A_{k+1} \cdots A_m\}$ and $S_{yes} = \{A_k, A_{k+1}, \cdots A_m\}$. Thus by taking $i = k$ in equations 7 and 8, we have the following probability bounds for ϕ states:

$$\sum_{A_i \in S_{yes}} \pi_{inf}(s, A_i, n) \leq \sum_{s' \models \phi} \pi(s, s', n) \leq \sum_{A_i \in S_{yes}} \pi_{sup}(s, A_i, n) \tag{9}$$

$$\sum_{A_i \in S_{yes}} \pi_{inf}(A_i) \leq \sum_{s' \models \phi} \pi(s') \leq \sum_{A_i \in S_{yes}} \pi_{sup}(A_i) \tag{10}$$

In the following proposition, we provide transient and steady-state reward bounds. Recall that rewards of macro-states are defined as follows: $\rho_{sup}(A_i) = max\{\rho(s), s \in A_i\}$ and $\rho_{inf}(A_i) = min\{\rho(s), s \in A_i\}$. And macro-states of S_{yes} are ordered according to increasing rewards: $\rho(A_k) \leq \rho(A_{k+1}) \cdots \leq \rho(A_m),\ \rho \in \{\rho_{inf}, \rho_{sup}\}$. As it has been stated before, the macro-state ordering may be different for the upper and the lower bounding computations. Without loss of generality we take the same ordering in both cases. In fact ϕ is satisfied in all macro-states of S_{yes}, so rewards are computed for all macro-states of S_{yes}.

Proposition 1. *We have the following inequalities on rewards:*

– Transient reward bounds:

$$\sum_{A_i \in S_{yes}} \pi_{inf}(s, A_i, n)\rho_{inf}(A_i) \leq \sum_{s' \models \phi} \pi(s, s', n)\rho(s') \leq \sum_{A_i \in S_{yes}} \pi_{sup}(s, A_i, n)\rho_{sup}(A_i) \tag{11}$$

– Steady-state reward bounds:

$$\sum_{A_i \in S_{yes}} \pi_{inf}(A_i)\rho_{inf}(A_i) \leq \sum_{s' \models \phi} \pi(s')\rho(s') \leq \sum_{A_i \in S_{yes}} \pi_{sup}(A_i)\rho_{sup}(A_i) \tag{12}$$

Proof. By construction, the distributions are $\leq_{st}$ comparable (equations 7, 8). Therefore we have the inequalities between the increasing functionals of these distributions (see definition 1). In fact inequalities 11, 12 are the increasing functionals on these distributions. Reward function of S_{no} states is zero and in the upper bound some rewards are replaced by greater values while they are replaced by smaller values in the lower bound. □

For a given reward formula $R(\phi)$, let $R_{sup}(\phi)$ (resp. $R_{inf}(\phi)$) be the reward on the macro-states of S_{yes} computed through the upper (resp. lower) bounding distribution. The following proposition gives how we can check formula $R(\phi)$ to see if it meets the bound of $I \in [r_{min}, r_{max}]$.

Proposition 2. *1. if $R_{inf}(\phi) \geq r_{min}$ and $R_{sup}(\phi) \leq r_{max}$ then we can conclude that $R(\phi)$ is true*

2. if $R_{inf}(\phi) \geq r_{max}$ or $R_{sup}(\phi) \leq r_{min}$ then we can conclude that $R(\phi)$ is false

3. *otherwise, we cannot conclude if $R(\phi)$ is true or not, through these bounding distributions. We can either modify the aggregation scheme (partition of states) or try to compute exact rewards.*

Proof. We give here the proof by specifying $R(\phi)$ for the sake of simplicity. Let us consider $\mathcal{E}_I(\phi)$ which is satisfied, if $\sum_{s' \models \phi} \pi(s')\rho(s') \in I$. It follows from equation 12 that

$$R_{inf}(\phi) \leq \mathcal{E}_I(\phi) \leq R_{sup}(\phi)$$

Thus, case 1 allows us to conclude that $\mathcal{E}_I(\phi)$ is satisfied:

$$r_{min} \leq R_{inf}(\phi) \leq \mathcal{E}_I(\phi) \leq R_{sup}(\phi) \leq r_{max}$$

Similarly, case 2 lets us to conclude that $\mathcal{E}_I(\phi)$ is not satisfied. Otherwise the rewards computed on bounding distributions do not let us to check $\mathcal{E}_I(\phi)$.

Let us remark that the case of transient reward formulas follows from equation 11. In the same manner formula $\mathcal{L}_{\lhd p}(\phi)$ can be checked by means of equation 10. □

5 Numerical Examples

In this section, we present numerical results computed from the proposed methodology. We consider four finite buffers in tandem where each buffer is a D/D/1/B queue (figure 2). The external arrivals and the services in all stages are independently, identically distributed batch processes with maximum size G. Let p_{ik} be the probability that k packets are served during a slot in stage $i, 1 \leq i \leq 4$ and $0 \leq k \leq G$. Indeed the service in stage i constitutes the arrivals to stage $i+1$. External arrivals are denoted by p_{0k}. At the end of a slot, it is assumed that first the end of services takes place and then the arrived packets are accepted. The packet acceptance mechanism is the rejection: a packet which arrives to a full buffer is lost.

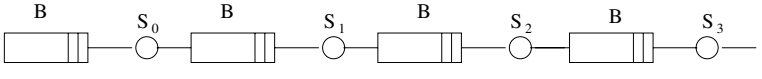

Fig. 2. A tandem queue with four buffers

Let $N_i(t), \quad 1 \leq i \leq 4$ be the number of packets at time t in buffer i. Thus $\{(N_4(t), N_3(t), N_2(t), N_1(t)),\ t \geq 0\}$ is a Discrete Time Markov chain of size $(B+1)^4$. In the sequel, we denote by $s = (n_4, n_3, n_2, n_1)$ a state of this Markov chain. We are interested in packet loss characteristics in buffer 4. Since all earlier stages must be taken into account to compute packet losses in this buffer, we must consider whole Markov chain of $(B+1)^4$ size. Thus the numerical complexity to solve the underlying model increases rapidly with B.

We define the following atomic propositions related to buffer 4:

- *frth-full* is valid if the fourth buffer is full. $S_{yes} = \{s \mid n_4 = B\}$
- *frth-loss* is valid if a packet loss may occur. $S_{yes} = \{s \mid n_4 > B - G\}$

Based on these atomic propositions, we check the following state formulas:

Steady-state formulas:

- $\mathcal{E}_{[0,10^{-7}]}(frth\text{-}loss)$ to check whether the long-run loss rate in buffer 4 is lower than 10^{-7} or not.
- $\mathcal{L}_{\leq 10^{-9}}(frth\text{-}full)$ to check whether the probability that buffer 4 is full in steady-state is less than 10^{-9} or not.

Transient formulas: For all these formulas we suppose that at the beginning all buffers are empty.

- $\mathcal{I}^n_{[0,10^{-9}]}(frth\text{-}loss)$ to check whether the expected packet loss at time n, $n \in \{40, 50\}$, meets the bound of I or not.
- $\mathcal{C}^n_{[0,10^{-9}]}(frth\text{-}loss)$ to check whether the expected cumulated packet loss up to time n, $n \in \{40, 50\}$, meets the bound or not.
- $\mathcal{E}^n_{[0,10^{-9}]}(frth\text{-}loss)$ to check whether the expected packet loss per unit time up to time n, $n \in \{40, 50\}$, meets the bound or not.

We now give the rewards assigned to states to compute these formulas related to packet losses in buffer 4. For a given state $s = (n_4, n_3, n_2, n_1)$,

$$\rho(s) = \sum_{j=0}^{G} \sum_{k=0}^{G} p_{3j} \cdot p_{4k} \cdot (\max(0, n_4 + \min(n_3, j) - k - B)) \tag{13}$$

To check these state formulas, we must compute transient or steady-state distribution of the underlying Markov chain. We check these formulas by solving upper bounding aggregated Markov chains to overcome state-space explosion. First we construct the exact Markov chain by means of evolution equations of the system [9]. In fact we begin by a state and generate all transitions (states) by taking into account the events which can occur in the system and their probabilities.

The second step is to aggregate states to define macro-states. We define macro states regarding to the number of customers in buffer 3 and 4 without considering the number of packets in the first two stages. Thus a macro-state (n_4, n_3) contains all states (n_4, n_3, i, j) $\forall i, j \in [0, B]$. Due to this aggregation procedure, the state space size will be reduced to $(B + 1)^2$. We reorder states using the lexicographic ordering to put together states of macro-states before running LIMSUB algorithm. Moreover states of S_{yes} must be after states of S_{no} and they must be ordered according to increasing rewards because of the $\leq_{st}$ ordering. In the considered example the reward function is largely compatible with the lexicographic ordering, we must reorder only a little number of states (equation 13).

The last step is to solve the upper bounding aggregated Markov chain (Q_{sup}) to compute the bounding distributions. Since it is defined on a reduced state

Table 1. Arrival process

Probabilities		External arrivals	First stage service	Second stage service	Third stage service	Fourth stage service
$\mathcal{PROC}_1$	p_0	0.7	0.3	0.4	0.6	0.2
	p_1	0.2	0.5	0.3	0.2	0.4
	p_2	0.1	0.2	0.3	0.2	0.4
$\mathcal{PROC}_2$	p_0	0.7	0.3	0.4	0.5	0.3
	p_1	0.2	0.5	0.3	0.3	0.3
	p_2	0.1	0.2	0.3	0.4	0.4

Table 2. Results obtained with the arrival process $\mathcal{PROC}_1$

Formulas	B	Exact	Bound	Valid?
$\mathcal{E}_{[0,10^{-9}]}(frth\text{-}loss)$	25	5.9610^{-16}	3.4910^{-11}	yes
	30	-	1.1310^{-10}	yes
$\mathcal{I}^{40}_{[0,10^{-9}]}(frth\text{-}loss)$	25	2.910^{-20}	1.7610^{-11}	yes
	30	-	$5.68\ 10^{-11}$	yes
$\mathcal{I}^{50}_{[0,10^{-9}]}(frth\text{-}loss)$	25	10^{-18}	$2.81\ 10^{-11}$	yes
	30	-	$1.09\ 10^{-10}$	yes
$\mathcal{C}^{40}_{[0,10^{-9}]}(frth\text{-}loss)$	25	6.8910^{-20}	1.310^{-10}	yes
	30	-	$3.67\ 10^{-10}$	yes
$\mathcal{C}^{50}_{[0,10^{-9}]}(frth\text{-}loss)$	25	3.7710^{-18}	3.710^{-10}	yes
	30	-	$1.26\ 10^{-9}$	unknown
$\mathcal{E}^{40}_{[0,10^{-9}]}(frth\text{-}loss)$	25	1.7210^{-21}	3.2710^{-12}	yes
	30	-	$9.17\ 10^{-12}$	yes
$\mathcal{E}^{50}_{[0,10^{-9}]}(frth\text{-}loss)$	25	$7.55\ 10^{-18}$	7.4110^{-12}	yes
	30	-	$2.52\ 10^{-11}$	yes
$\mathcal{L}_{\leq 10^{-9}}(frth\text{-}full)$	25	2.3510^{-15}	1.2310^{-10}	yes
	30	-	1.2110^{-12}	yes

space, this can be done efficiently. We have applied an indirect method (Gauss-Seidel) [19] to compute bounding distributions.

We fix the maximum size of batches, $G = 2$ and consider two different arrival processes, $\mathcal{PROC}_1$ and $\mathcal{PROC}_2$. The probabilities of having i batches, p_i $0 \leq i \leq 2$ for external arrivals and for services in each stage are given in the following table.

In table 2, we give the results computed for arrival process $\mathcal{PROC}_1$. For each formula we give results for $B = 25$ and $B = 30$. However, we could not solve the

Table 3. Results obtained with the arrival process $\mathcal{PROC}_2$

Formulas	B	Exact	Bound	Valid?
$\mathcal{E}_{[0,10^{-7}]}(frth\text{-}loss)$	25	$7.09\ 10^{-14}$	$1.04\ 10^{-7}$	unknown
	30	-	$1.79\ 10^{-8}$	yes
$\mathcal{I}^{40}_{[0,10^{-7}]}(frth\text{-}loss)$	25	$3.4\ 10^{-17}$	$1.8\ 10^{-8}$	yes
	30	-	$3.37\ 10^{-11}$	yes
$\mathcal{I}^{50}_{[0,10^{-7}]}(frth\text{-}loss)$	25	7.910^{-16}	4.410^{-8}	yes
	30	-	1.5510^{-10}	yes
$\mathcal{C}^{40}_{[0,10^{-7}]}(frth\text{-}loss)$	25	$8.54\ 10^{-17}$	$1.08\ 10^{-7}$	unknown
	30	-	1.4210^{-10}	yes
$\mathcal{C}^{50}_{[0,10^{-7}]}(frth\text{-}loss)$	25	$3.3\ 10^{-15}$	$4.31\ 10^{-7}$	unknown
	30	-	1.0610^{-9}	yes
$\mathcal{E}^{40}_{[0,10^{-7}]}(frth\text{-}loss)$	25	2.1310^{-18}	2.710^{-9}	yes
	30	-	$3.56\ 10^{-12}$	yes
$\mathcal{E}^{50}_{[0,10^{-7}]}(frth\text{-}loss)$	25	6.610^{-17}	$8.6\ 10^{-9}$	yes
	30	-	$2.13\ 10^{-11}$	yes
$\mathcal{L}_{\leq 10^{-7}}(frth\text{-}full)$	25	$2.22\ 10^{-13}$	$2.95\ 10^{-7}$	unknown
	30	-	5.0410^{-8}	yes

chain with $B = 30$ because of its size (see table 4). In the last column we give if the formula can be checked through these bounding distributions or not. For this arrival process, most of the formulas can be checked through these bounding distributions. In the last column, *unknown* indicates that we cannot conclude whether the formula is satisfied or not through these bounding distributions.

In table 3, we give the results under arrival process $\mathcal{PROC}_2$. For this arrival process, some formulas cannot be checked through these bounding distributions. We can change the aggregation procedure to have more detailed representation of the underlying system.

The numerical results are computed in an Intel Pentium 4 with CPU 2.8 GHz and 1.5GBytes memory. Let us give computation times for different steps for exact and bounding Markov chains (see table 4). We give in columns *Size*

Table 4. Comparison of original and bounding model sizes

	Exact Markov chain				Bounding Markov chain			
B	Size	Entries	Generation	Resolution	Size	Entries	Generation	Resolution
25	456 976	77 970 677	9 min	16 min	676	13 397	6 min	0.001s
30	923 521	163 169 007	10 min	-	961	19 242	12 min	0.13 s

the state space size, and in columns *Entries* the number of non null entries of the chain. For the exact chain, *Generation* time corresponds to the time for generating the underlying Markov chain, while it corresponds to the reordering of states and the execution of LIMSUB algorithm for the bounding chain.

We can see that the resolution times are drastically reduced for bounding chains due to the state space size reduction. Therefore it will be possible to check large models through bounding distributions. Actually, the underlying matrix is stored in the memory during the computation of the bounding model. However the bounding chain is constructed column by column so it is possible to avoid the storage of whole matrix using Kronecker or MTBDD structures. These issues are under work to be able to check very large models.

6 Conclusions

In this paper we show how algorithmic stochastic bounding techniques can be applied to check state formulas in the PRCTL logic. Indeed we must compute a transient or the steady-state distribution of the underlying DTMC to check state formulas. However the computation of these distributions has high numerical complexity or is intractable because of the well-known state space explosion problem. On the other hand we do not need in general exact values to check these formulas. Therefore bounding techniques are useful in stochastic model checking. We proposed to apply stochastic bounding algorithms to overcome the state space explosion problem. Since bounding models can be constructed in a fully automated manner by means of the bounding algorithms, the proposed methodology can be easily integrated to model checkers.

In this work we are interested only on state formulas, but this approach can be also extended to path formulas. In fact we apply the $\leq_{st}$ stochastic ordering, which is also called as sample-path ordering. Intuitively this means that if two chains are comparable in this stochastic ordering sense, their sample-paths are comparable. We are working on the application of $\leq_{st}$ stochastic ordering to check path formulas.

Acknowledgements. The authors thank Jean-Michel Fourneau for fruitful discussions.

References

1. Abu-Amsha, O., Vincent, J.M.: An algorithm to bound functionals of Markov chains with large state space. In: *4th INFORMS Conference on Telecommunications*, Boca Raton, Florida, (1998)
2. Andova, S., Hermanns, H., Katoen, J.P.: Disrete-time rewards model-checked. In Formal Modelling and Analysis of Timed Systems (FORMATS 2003), Marseille France.
3. Aziz, A., Sanwal, K., Singhal,V. and Brayton R.: Model checking continuous time Markov chains. *ACM Trans. on Comp. Logic*, 1(1), p. 162-170, 2000.

4. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Automated performance and dependability evaluation using Model Checking. *LNCS 2459, Performance evaluation of complex systems: Techniques and Tools*, pp 64-88, 2002.
5. Benmammoun, M., Fourneau, J.M., Pekergin, N., Troubnikoff, A.: An algorithmic and numerical approach to bound the performance of high speed networks. In IEEE MASCOTS 2002, Fort Worth, USA, pp. 375-382.
6. Clarke, E.M., Emerson,A., Sistla, A. P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems* 8(2):244–263, 1986.
7. D'Argenio, P.R. Jeannet, B., Jensen, H.E. and Larsen, K.G. Reduction and Refinement Strategies for Probabilistic Analysis. In Proc *Process Algebra and Probabilistic Methods Performance Modeling and Verification*, Springer-Verlag, 2001.
8. Fourneau, J.M., Pekergin, N., Younès, S.:Improving Stochastic Model Checking with Stochastic Bounds. In SAINT Modelling and Performance Evaluation in Next Generation Internet Workshop, 2005.
9. Fourneau, J.M., Lecoz, M., Pekergin, N., Quessette, F.: An open tool to compute stochastic bounds on steady-state distributions and rewards. In IEEE MASCOTS 2003, pp. 219-225.
10. Fourneau,J.M., Pekergin, N.: An algorithmic approach to stochastic bounds. *LNCS 2459, Performance evaluation of complex systems: Techniques and Tools*, pp 64-88, 2002.
11. Fourneau, J.M., Lecoz, M. and Quessette, F.: Algorithms for irreducible and lumpable strong stochastic bound. *Linear Algebra and its Applications* 386(2004) 167-185, 2004.
12. Hansson,H. and Jonsson B.: A logic for reasoning about time and reliability. In *Form. Asp. of Comp.* **6**: 512-535, 1994.
13. Haverkort, B., Cloth, L., Hermanns, H., Katoen, J.P. and E C. Baier: Model Checking Performability Properties In Proc. *Dependability Systems and NETWORKS (DSN) 2002, IEEE CS Press*, 2002.
14. Kulkarni, V.G.: *M*odeling and Analysis of Stochastic Systems.Chapman& Hall, 1995.
15. Kwiatkowska, M., Norman, G., Parker D.: PRISM: Probabilistic Symbolic Model Checker. In Proc.*TOOLS 2002, volume 2324 of LNCS*, p. 200-204, Springer-Verlag April 2002.
16. Muller, A. and Stoyan, D.: *Comparison Methods for Stochastic Models and Risks*, Wiley , New York,2002.
17. Pekergin N.:Stochastic performance bounds by state space reduction. *Performance Evaluation*, 36-37, pages 1-17, 1999.
18. Shaked, M. and Shantikumar, J.G.: *Stochastic Orders and Their Applications*, Academic Press, San Diago, 1994.
19. Stewart W. J: *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, (1994)