Model checking of steady-state rewards using bounding aggregations

Hind Castel
GET/INT/SAMOVAR
INT
9,rue Charles Fourier
91011 Evry Cedex, France
hind.castel@int-evry.fr

Lynda Mokdad
Lamsade Laboratory
Université de Paris Dauphine
Place du Maréchal de Lattre de Tassigny
75775 cedex 16, France
lynda.mokdad@lamsade.dauphine.fr

Nihal Pekergin
LACL Laboratory
Université Paris Est
61, av. du Général de Gaulle
94010 Créteil Cedex, France
nihal.pekergin@univ-paris12.fr

Abstract—**This paper presents an algorithm based on stochastic comparisons in order to check formulas with rewards on multidimensional Continuous Time Markov Chains (CTMC). These formulas are expressed in Continuous Stochastic Logic (CSL) which includes means to express transient, steady-state and path performance measures. However, computation of transient and steady state distribution are limited to relatively small sizes because of the state space explosion problem.**

We propose a model checking algorithm based on aggregated bounding Markov processes in order to perform the verification on the bounds values instead of the exact one. The stochastic comparison has been largely applied in performance evaluation however the state space is generally assumed to be totally ordered which induces less accurate bounds for multidimensional Markov processes. We use the increasing set theory and the comparison by mapping functions in order to derive bounds on reduced state spaces. The relevance of the proposed checking algorithm is the possibility of a parametric aggregation scheme in order to improve the accuracy of the bounds and in the same time the precision of the checking, but in return with an increasing of the complexity.

I. Introduction

Continuous Time Markov Chain (CTMC) is the most useful mathematical model underlying many formalisms for the description and the quantitative analysis of stochastic discrete-event dynamic systems. Once a CTMC has been generated, the next step is to compute performance measures such as throughput, loss probabilities, etc. Continuous Stochastic Logic CSL can be used to express constraints on such measures and model checking techniques are applied to automated analysis of these constraints. The CSL [1] was developed as a stochastic extension of the branching-time Temporal Logic (CTL) [5]. Stochastic model checking has been extended to models with some rewards on states and transitions in which logic formalisms as Probabilistic Reward Computational Tree Logics (PRCTL) and Stochastic Reward Logic (CSRL) [9] are used for performance and dependability applications.

We propose to check the rewards based formulas of stochastic models by applying stochastic comparison approach. To check these formulas, transient or steady-state distribution of underling Markov chain must be computed. However, these models are usually represented by multidimensional processes with very large state spaces. As a result, quantitative analysis to check formulas, is difficult if there is no specific solution form (product form solutions, ...). Since exact performance measures can only be obtained using numerical methods [18] with small sizes, it is important to develop new powerful mathematical tools for large state systems.

Related works. In this paper, we propose a model checking algorithm in order to verify reward formulas on aggregated bounding Continuous Time Markov Chain (CTMC). In [15], a stochastic comparison based method is proposed to check state formulas defined over Discrete Time Markov Chain (DTMC) rewards. The authors have assumed a total order, and generate the aggregated Markov chains using the LIMSUB algorithm [8], based on lumpability constraints. In the present paper, we use an algorithm generating bounding aggregated Markov chains presented in [3], [4]. It is based only on a partial order which induces less constraints then a total order, and so more accurate bounds for multidimensional Markov processes. Also, we propose a parametric aggregation scheme in order to improve the accuracy of bounds and also the precision of the checking.

In [1,17], model checking algorithms for CTMC are proposed using lumping equivalence in order to reduce Markov chains state spaces. The notion of state equivalence relation also called stochastic bisimulation is introduced in order to build a reduced state space called *quotient*. The relation is based on the required conditions that must have equivalent states as equality of state labelling, rewards, exit rates. The advantage is the verification of CSL logic formulas on a reduced state space.

Compared with this study, the algorithm which we propose does not suppose lumping equivalence, and so it is easier to aggregate the state space, but in return the verification is made on bounding values instead of exact ones which does not let to decide for all cases.

This paper is organized as follows. Next, we present the CSL logic, focusing on formulas with rewards for performance study. In the section after, we present the stochastic comparison method on multidimensional state spaces in order to generate aggregated bounding Markov chains. In section IV, we apply the algorithm to the model checking of large state Markov chains. We use the increasing set theory in order to generate bounds on performance measures with rewards. The section V is devoted to checking if loss probabilities in a tandem queueing network is included or not in an interval. Different parametric aggregation schemes have been proposed in order to show the trade off between improving the precision of the checking, and increasing the state space. As a conclusion, we resume advantages and limits of our approach. Finally, we resume in an appendix the stochastic ordering theory used in this paper.

II. Model checking Markov chains

Continuous Stochastic Logic (CSL) is an extension of Computation Tree Logic (CTL) with two probabilistic operators that refer to steady-state and transient behaviors of the underlying system.
CSL is a branching-time temporal logic with state and path formulas. The states formulas are interpreted over states of a CTMC, whereas the path formulas are interpreted over paths in CTMC [1]. In order to express the time span of a certain path, the path operators until ($\mathcal{U}$) and next ($\mathcal{X}$) are extended with a parameter that specifies a time interval [1].
The CSL logic has been extended to continuous stochastic reward logic (CSRL) to specify performability measures over Markov reward models (MRMs). It contains operators that refer to the stationary and transient behaviour of the considered systems. To specify performability measures as logical formulas over MRMs, it is assumed that each state is labelled with so-called atomic propositions. Atomic propositions identify specific situations the system may be in, such as "buffer full", "buffer empty" or "variable X is positive" [9].

The MRM is a tuple $\mathcal{M} = (E, R, \rho)$ where E is a finite set of states, $R : E \times E \longrightarrow \mathcal{R}_{\geq 0}$ is the rate matrix, and $\rho : E \longrightarrow \mathcal{R}_{\geq 0}$ is a reward structure that assigns to each state s a reward $\rho(s)$, also called cost. The MRM has a fixed initial distribution α satisfying $\sum_{s \in E} \alpha_s = 1$, so the MRM starts in state s with probability α_s.
CSRL allows one to specify properties over states and over paths. A path is an alternating sequence $s_0\ t_0\ s_1\ t_1 \dots$ where s_i is a state of the MRM and $t_i > 0$ is the sojourn time in state s_i. The accumulated reward for a finite path of length n is given by $\sum_{i=0}^{n-1} t_i \rho(i)$. Let I and J be intervals on the real line, p a probability and $\lhd$, a comparison operator, such as $>$ or $<$. The syntax of CSRL is defined by the following grammar [9]:

State-formulas:

$$\phi ::= true \mid a \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \mathcal{P}_{\lhd p(\phi)}$$

Path-formulas:

$$\phi ::= (\phi\ \mathcal{U}_J^I \phi) \mid \mathcal{X}_J^I(\phi)$$

For the state formulas $\neg$, $\vee$ and $\wedge$ the meanings are as usual. For the formula $\mathcal{P}_{\lhd p(\phi)}$, it is valid in state s if the probability measure of the set of paths starting in s and satisfying path formula ϕ meets the bound $\lhd p$. The path-operators next $\mathcal{X}$ and until $\mathcal{U}$ are used with two intervals. Interval I can be considered as a timing constraint whereas J represents a bound for the cumulative reward. In the following, we give examples for the both path-operators.

- $\mathcal{P}_{\leq 0.2}(\mathcal{X}_{(0,\infty)}^{\leq 5} s)$ which means that with a probability at most 0.2, a transition can be made to s-states at time $t \in [0, 5]$ such that the accumulated reward until t lies in $(0, \infty)$.
- $\mathcal{P}_{\leq 0.2}(a\mathcal{U}_{(0,\infty)}^{\leq 5} b))$ which means that b-state can be reached with probability at least 0.2 at time $t \in [0, 5]$ along a-states such that the accumulated reward until t lies in $(0, \infty)$.

In this paper, we consider steady-state reward formulas as given in [1] for discrete time Markov chains:

Steady-state Reward-formulas:

$$\varepsilon_I(\phi) \mid \mathcal{L}_{\lhd p}(\phi)$$

These formulas are inspired from performance measures of CTMC with rewards.

- The formula $\varepsilon_I(\phi)$ is the long-run expected reward for ϕ-states. If the steady-state exists, $\varepsilon_I(\phi)$ is satisfied if:

$$\varepsilon_I(\phi) \text{ is satisfied iff } \sum_{s' \models \phi} \pi(s')\rho(s') \in I$$

- The formula $\mathcal{L}_{\lhd p}(\phi)$ asserts that the steady state probability to be in ϕ-states meets the bound $\lhd p$.

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

III. Stochastic comparison of multidimensional Markov processes

The stochastic comparison is a mathematical tool which allows to compute bounds on transient distributions and the stationary distribution of a Markov process. In fact, if the underlying Markov process does not have a specific solution form like a product-form or matrix-geometric solutions, etc. the computation of stationary probability distributions becomes difficult or intractable for multidimensional large state spaces.

By means of the stochastic comparison method, it is possible to overcome this problem by reducing the size of the state space of the underlying Markov process. Stochastic comparison is based on the definition of stochastic orderings between random variables, probability measures, or stochastic processes. In this paper, we use the stochastic comparison by mapping functions in order to reduce the state space size. We have summarized in the appendix the main concepts of stochastic comparisons of multidimensional Markov processes used in this paper.

Let $X(t)$ be a large state space Markov process defined on a multidimensional and preordered (not necessarily a totally ordered) state space E, with an infinitesimal generator matrix Q. We suppose that the process is irreducible so the stationary distribution Π exists, but it is very difficult to compute (not a specific solution form, and a large state space size).

The study of this system consists in the checking of the formula $\varepsilon_I(\phi)$, which is true if :

$$R(\phi) \in I$$

where :

$$R(\phi) = \sum_{s \models \phi} \Pi(s)\rho(s)$$

so $\varepsilon_I(\phi)$ is satisfied if the long-run expected reward rate per time-unit for ϕ-states meets the bound of $I = [I_{min}, I_{max}]$. If there is a closed-form for Π then it will be easy to compute and the verification can be easiliy done . Otherwise, as the state space size increases exponentially with the number of components, then it will be very difficult to verify $\varepsilon_I(\phi)$. We propose to define aggregated bounding Markov processes [3,4], in order to derive an aggregated upper (resp. lower) bounding Markov process, from which we can compute the stationary distribution Π^u (resp. Π^l) defined on a smaller state space. We verify the formula $\varepsilon_I(\phi)$ by computing an upper bound (resp. a lower bound) $R^u(\phi)$ (resp. $R^l(\phi)$) to $R(\phi)$, and testing if they are included in I.

We use the algorithm generating aggregated bounding Markov processes [3]. The goal of the algorithm is to derive :

- The Markov process $\{X^u(t), t \geq 0\}$ representing the upper bound computed from $\{X(t), t \geq 0\}$, and the many to one mapping function $g^u : E \to S^u$, $S^u \subset E$.
- The Markov process $\{X^l(t), t \geq 0\}$ representing the lower bound, computed from $\{X(t), t \geq 0\}$, and the mapping function $g^l : E \to S^l$, $S^l \subset E$.

In [3], we have proved using the stochastic ordering theory described in the appendix that the algorithm generates aggregated Markov processes bounds for the exact process. For the upper bound, we have the comparison by the mapping function g^u :

$$\{g^u(X(t)), t \geq 0\} \preceq_{st} \{X^u(t), t \geq 0\}$$

and for the lower bound, by the mapping function g^l :

$$\{X^l(t), t \geq 0\} \preceq_{st} \{g^l(X(t)), t \geq 0\}$$

As results of the comparison of the processes, we have the comparison of transient and stationary probability distributions. Only the second one is used in this paper, it is explained in the next section.

A. Probability distributions comparison

We consider here only the stationary case. From the proposition 1 in the appendix, we have the following relations :

$$\sum_{x|g^u(x)\in\Gamma} \Pi[x]f(x) \leq \sum_{x\in\Gamma} \Pi^u[x]f(x), \forall\Gamma \in \Phi_{st}(S^u) \quad (1)$$

and :

$$\sum_{x\in\Gamma} \Pi^l[x]f(x) \leq \sum_{x|g^l(x)\in\Gamma} \Pi[x]f(x) \forall\Gamma \in \Phi_{st}(S^l) \quad (2)$$

where $f : E \to \mathbb{R}_{>0}$, is an increasing function.

IV. Model checking on aggregated bounding CTMC

In this section we use the stochastic comparisons and precisely the algorithm generating aggregated Markov processes in order to check steady-state reawrd formulas.

A. States formulas bounds

We focus on the state formula with rewards $\varepsilon_I(\phi)$, and we want to verify if it is true or not. We need to compute $R(\phi)$ given by :

$$R(\phi) = \sum_{s\in E_{yes}} \Pi[s]\rho(s)$$

where $E_{yes} = \{s \in E | s \models \phi\}$, and $\rho : E \to \mathbb{R}_{>0}$ is not necessary an increasing function. For multidimensional processes, it is difficult to compute Π if there is no product form, because the state space size increases exponentially. So we propose to compute bounds to $R(\phi)$ in order to verify $\varepsilon_I(\phi)$. These bounds are computed from aggregated bounding Markov processes generated from the exact Markov process. We apply the algorithm generating aggregated bounding Markov processes [3] in order to derive bounding aggregated Markov processes.

As some states of E_{yes} will be aggregated, then the computation of the bounds will be made on the states of $g^u(E_{yes})$ for the upper bound, and $g^l(E_{yes})$ for the lower bound. Although E_{yes} is not an increasing set, we could have the aggregated sets $g^u(E_{yes})$ and $g^l(E_{yes})$ defined as increasing sets. In the example 1 of the appendix, $E_{yes} = \{(0,1)\}$ is not an increasing set, but $g(E_{yes}) = \{(1,1)\}$ is an increasing set. We suppose that $g^u(E_{yes})$ and $g^l(E_{yes})$ are increasing sets which could be very useful for the stochastic comparison of the stationary distributions.

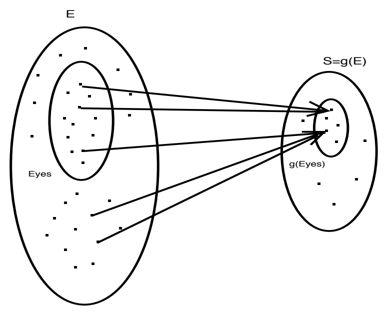

Fig. 1. Aggregation of E_{yes} : the general case

From proposition 2 of the appendix, and Fig.1 where g is replaced by g^u, we have :

$$\sum_{x\in E_{yes}} \Pi[x] \leq \sum_{x|g^u(x)\in g^u(E_{yes})} \Pi[x]$$

If we use the reward function ρ, and if we define an increasing function ρ_1, representing an upper bound to ρ, then we have the following inequality :

$$R(\phi) = \sum_{x \in E_{yes}} \rho(x)\Pi[x] \leq \sum_{x | g^u(x) \in g^u(E_{yes})} \rho_1(x)\Pi[x]$$

From equation 1 in section III-A , as $g^u(E_{yes})$ is an increasing set of $\Phi_{st}(S^u)$, then we obtain :

$$\sum_{x | g^u(x) \in g^u(E_{yes})} \rho_1(x)\Pi[x] \leq R^u(\phi) = \sum_{x \in g^u(E_{yes})} \rho_1(x)\Pi^u[x]$$

So we deduce that

$$R(\phi) \leq R^u(\phi)$$

We study now the lower value case. From proposition 2 in the appendix, and figure Fig.1, where we replace g by g^l, we have :

$$\sum_{x \in E_{yes}} \Pi[x] \leq \sum_{g^l(x) \in g^l(E_{yes})} \Pi[x]$$

It is clear that the inequality sense is inadequate with the definition of a lower bound. We suppose the particular case where the states of $g^l(E_{yes})$ are the mappings of states which are only in E_{yes}, and not outside E_{yes}, as it is represented in Fig.2. This assumption will be possible if we can define a mapping function g^l from the set E_{yes} , verifying this condition.

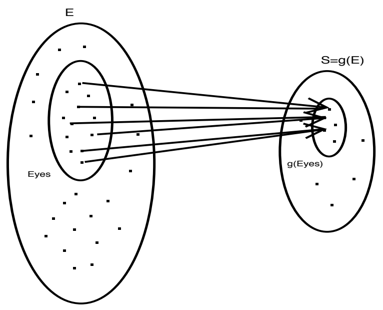

Fig. 2. Aggregation of E_{yes} : a particular case

It is equivalent to say that :

$$\nexists x \mid x \notin E_{yes},\ and\ g^l(x) \in g^l(E_{yes})$$

In this case, we have :

$$\sum_{x \in E_{yes}} \Pi[x] = \sum_{x | g^l(x) \in g^l(E_{yes})} \Pi[x]$$

If we introduce the reward functions, and if we denote by ρ_2 an increasing reward function representing a lower bound for ρ, then we have :

$$R(\phi) = \sum_{x \in E_{yes}} \Pi[x]\rho(x) \geq \sum_{x | g^l(x) \in g^l(E_{yes})} \Pi[x]\rho_2(x)$$

From equation 2 in section III-A as $g^l(E_{yes})$ is an increasing set, we have :

$$\sum_{x | g^l(x) \in g^l(E_{yes})} \Pi[x]\rho_2(x) \geq R^l(\phi) = \sum_{x \in g^l(E_{yes})} \Pi^l[x]\rho_2(x)$$

So we obtain :

$$R(\phi) \geq R^l(\phi)$$

B. Checking state formulas

From the computation of $R^u(\phi)$ and $R^l(\phi)$, we can verify if $\varepsilon_I(\phi)$ is true or not. We have several cases :

1) $R^u(\phi) \leq I_{max}$ and $R^l(\phi) \geq I_{min}$ so as $R^l(\phi) \leq R(\phi) \leq R^u(\phi)$, then we can say that $\varepsilon_I(\phi)$ is true.
2) $R^u(\phi) \geq I_{max}$ or $R^l(\phi) \leq I_{min}$, then we can't conclude if $\varepsilon_I(\phi)$ is true or not, we can try another aggregation scheme, more precise, defined on a larger state space size in order to improve the quality of the bounds and to be perhaps in the case (1).
3) $R^u(\phi) \leq I_{min}$ or $R^l(\phi) \geq I^{max}$ then $\varepsilon_I(\phi)$ is false.

From these inequalities, we define three boolean variables :

- cond1=$R^u(\phi) \leq I_{max}$ and $R^l(\phi) \geq I_{min}$,
- cond2=$R^u(\phi) \geq I_{max}$ or $R^l(\phi) \leq I_{min}$, and
- cond3=$R^u(\phi) \leq I_{min}$ or $R^l(\phi) \geq I^{max}$.

Here we give the procedure BoundCheck which check if $\varepsilon_I(\phi)$ is true or not. As input parameters, the function takes the infinitesimal generator Q, the interval I, and the set E_{yes}. The result of the checking is contained in the boolean variable Check, which equals "yes" if cond2 is verified, "no" if it is cond3. In the checking we use the aggregated bounding process in order to derive the bounds used for the verification. As we have explained previously, if $g^u(E_{yes})$ and $g^l(E_{yes})$ are increasing sets, then we can compare the stationary distributions, otherwise, we modify E_{yes} so as we obtain increasing sets. The parametric aggregation is interesting for the verification : we begin with small sizes of aggregated Markov processes, and step by step we enlarge the process in order to improve the verification. In the procedure, we use a variable stop in order to stop this process. So if stop='n', we continue to improve the bounds, otherwise we stop and if it is cond2 that it is verified, then the variable Check is such that Check=unknown.

```
PROCEDURE BoundCheck(IN Q,I,Eyes,OUT Check)

BEGIN

    -Build aggregated bounding
    Markov processes

 WHILE (cond2) AND (stop='n')
  THEN BEGIN

  - Build a larger Markov process in order
     to improve the quality of the bounds.
```

```
      END
   IF (cond1) THEN
          Check=yes
   ELSE IF (cond2) THEN
          Check=unknown
   ELSE IF (cond3)
        THEN Check=false

END
```

V. Application and numerical results

The system understudy represents a path in a network defined as a series of network nodes (switches, routers) where transits only one flow of packets. We suppose that the leftmost node has the index 1, and indexes increase in the path until node n. This system can be represented by n finite capacity queues in tandem (see Figure 3).

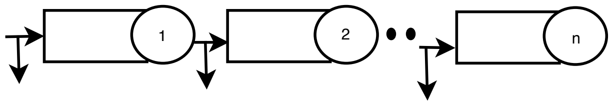

Fig. 3. Tandem queueing network

External arrivals occur only in queue 1, after the flow transits in queues $2,\dots,n$ if there is enough place in each queue. We suppose that arrivals are Poisson in queue 1 with rate λ. Each queue i has an Exponential service time with rate μ_i, and a finite capacity B_i. After a service in queue i, the customer transits to the next queue $i+1$ if there is enough place, otherwise the customer is lost. This system is represented by a Markov process $\{X(t),\ t \geq 0\}$ on $E = \{0,\dots,B_1\}\times\dots\times\{0,\dots,B_i\}\times\dots\times\{0,\dots,B_n\}$. Each state $x \in E$ is represented by a vector: $x = (x_1,\dots,x_i,\dots,x_n)$. where x_i is the number of customers waiting in queue i. We suppose that the stationary distribution denoted Π exists. We introduce the atomic proposition related to the state of the ith buffer : *ith-full* is valid if the ith buffer is full : $x_i = B_i$. So in this case, $E_{yes} = \{x \in E | x_i = B_i\}$. The goal of this study is to verify if the state formula $\varepsilon_I(\textit{ith-full})$ is true or not, which means if the long-run loss probabilities in buffer i is in the interval I. So $\varepsilon_I(\textit{ith-full})$ is true if

$$R(\textit{ith-full}) = \sum_{x \in E_{yes}} \Pi(x)\rho(x) \tag{3}$$

is included in I.Where $\rho(x) = 1$ if $x_i = B_i$ and otherwise 0 is an increasing function.

The numerical resolution of $\{X(t),\ t \geq 0\}$ in order to compute Π is very difficult or intractable : there is no product-form, and the number of states increases exponentially with the number of components. We propose to apply the algorithm generating aggregated bounding Markov processes in order to verify if

$$R^u(\textit{ith-full}) = \sum_{x \in g^u(E_{yes})} \Pi^u(x)\rho(x)$$

and

$$R^l(\textit{ith-full}) = \sum_{x \in g^l(E_{yes})} \Pi^l(x)\rho(x)$$

are included in I.

Δ	λ_i	$\varepsilon_{[10^{-10},10^{-1}]}(\textit{fourth-full})$	
		Aggregation 1	Aggregation 2
10	50	unknown	unknown
10	60	yes	unknown
10	70	yes	unknown
10	80	yes	yes
10	90	yes	yes
15	50	yes	unknown
15	60	yes	unknown
15	70	yes	yes
15	80	yes	yes
15	90	yes	yes

TABLE I
$\varepsilon_{[10^{-10},10^{-1}]}(\textit{fourth-full})$ WITH AGGREGATION 1 AND 2 AND $B_i = 20$

We give numerical results for the model with the following assumptions. Four buffers in tandem, service rates μ_i in each queue is $100Mb$, the packet size is 512 bytes. λ varies from 50 Mb/s to 90 Mb/s, and we take $B_i = B = 20$, and $B_i = B = 40$.

We propose to evaluate the state formula $\varepsilon_I(\textit{fourth-full})$ in order to verify if the packet loss probability of the fourth buffer is included in the interval I or not. This formula is verified from the computation of $R^u(\textit{fourth-full})$ and $R^l(\textit{fourth-full})$. Two aggregation schemes are proposed in order to define the aggregated bounding Markov processes [3,4] : *Aggregation1* which is a "fine aggregation" and *Aggregation2* a "coarse aggregation". In tables III and IV sizes of aggregated processes are given.

A. State formula verification

We verify the formula $\varepsilon_I(\textit{fourth-full})$ in two different intervals I, by computing $R^u(\textit{fourth-full})$ and $R^l(\textit{fourth-full})$. In table I, we suppose that $I = [10^{-10}, 10^{-1}]$, and we give the results of the verification for the two aggregation schemes. We have verified that for the two aggregation schemes, we are in the case that $g^u(E_{yes})$ and $g^l(E_{yes})$ are increasing sets, and for the lower bound 2 is verified. As we have explained before, Aggregation1 is defined so as to provide more precise bounds values then Aggregation2. In tables III and IV, we can see that Markov chains sizes are larger with Aggregation1 then Aggregation2, and when Δ increases, the sizes increases also.

From table I, we can see that $\varepsilon_{[10^{-10},10^{-1}]}(\textit{fourth-full})$ is in some cases unknown with Aggregation2, but true with Aggregation1, because the bounds are more precise. Another remark is that for an aggregation scheme, when Δ increases, the precision improvement makes the state formula transits from an unknown response (Aggregation1, $\Delta = 10$, $\lambda_i = 50$) to a positive response (Aggregation1, $\Delta = 15$, $\lambda_i = 50$).

For table II, given the same assumptions, we reduce the interval I which is now equal to $I = [10^{-6}, 10^{-2}]$. Clearly, we can see globally that the verification is more often unknown then in the precedent case. Clearly, we couldn't conclude

Δ	λ_i	$\varepsilon_{[10^{-6},10^{-2}]}$(*fourth-full*)	
		Aggregation 1	Aggregation 2
10	50	unknown	unknown
10	60	unknown	unknown
10	70	yes	unknown
10	80	yes	unknown
10	90	yes	unknown
15	50	unknown	unknown
15	60	yes	unknown
15	70	yes	unknown
15	80	yes	unknown
15	90	yes	unknown

TABLE II
$\varepsilon_{[10^{-6},10^{-2}]}$(*fourth-full*) : WITH AGGREGATIONS 1, 2 AND $B_i = 20$

Exact	Aggregation1		Aggregation2	
	Δ=10	Δ=15	$\Delta = 10$	Δ=15
194481	158071	191751	61051	140091

TABLE III
STATE SPACE SIZE OF AGGREGATED MARKOV PROCESSES FOR B=20

with Aggregation2, but the response can be obtained from Aggregation1, and with the increasing of Δ. The same remarks can be deduced from V and VI when we increase buffer sizes.

VI. CONCLUSION

In this paper, we have proposed a checker algorithm based on stochastic comparisons of multidimensional Markov chains. This algorithm has been applied to check state formulas with rewards defined on steady-state distribution, with CSRL logic. Quantitative analysis of large systems can be intractable if there is no specific solution as product forms. To overcome the state space explosion problem, we have proposed to use stochastic comparisons by mapping functions in order to define aggregated bounding Markov processes. We have defined an algorithm which verify the state formulas using the upper and lower bounds generated by aggregated Markov processes. We have proposed a parametric aggregation in order to have a tradeoff between the size of the aggregated Markov chains and the accuracy of the bounds. This aggregation allows the checker algorithm to refine its verification by becoming closer to the exact model.

Exact	Aggregation1		Aggregation2	
	Δ=15	Δ=20	Δ=15	Δ=20
2825761	1587311	2296141	438311	884101

TABLE IV
STATE SPACE SIZE OF AGGREGATED MARKOV PROCESSES FOR B=40

Δ	λ_i	$\varepsilon_{[10^{-13},10^{-1}]}$(*fourth-full*)	
		Aggregation 1	Aggregation 2
15	50	unknown	unknown
15	60	yes	unknown
15	70	yes	unknown
15	80	yes	yes
15	90	yes	yes
20	50	unknown	unknown
20	60	unknown	unknown
20	70	yes	yes
20	80	yes	yes
20	90	yes	yes

TABLE V
$\varepsilon_{[10^{-13},10^{-1}]}$(*fourth-full*) FOR AGGREGATIONS 1,2, AND $B_i = 40$

Δ	λ_i	$\varepsilon_{[10^{-12},10^{-2}]}$(*fourth-full*)	
		Aggregation 1	Aggregation 2
15	50	unknown	unknown
15	60	unknown	unknown
15	70	unknown	unknown
15	80	yes	unknown
15	90	unknown	unknown
20	50	unknown	unknown
20	60	unknown	unknown
20	70	yes	unknown
20	80	yes	unknown
20	90	unknown	unknown

TABLE VI
$\varepsilon_{[10^{-12},10^{-2}]}$(*fourth-full*) WITH AGGREGATIONS 1, 2 AND $B_i = 40$

VII. REFERENCES

[1] S. Andova, H. Hermanns, J.P. Katoen, ”Discrete-time rewards model-checked”, Formats2003, LNCS 2791, 2003.

[2] C. Baier, B. Haverkort, H. Hermanns and J.P. Katoen, ”Model-Checking Algorithms for Continuous Markov Chains”, IEEE Transactions on Software Engineering, Vol. 29, NO. 6, June 2003.

[3] M. Ben Mamoun, N. Pekergin and S. Younès, ”Model Checking of Continuous Time Markov Chains by Closed-Form Bounding Distributions”, Qest’06.

[4] H.Castel, L.Mokdad, N.Pekergin, ”Aggregated bounding Markov processes applied to the analysis of tandem queues”, Second International Conference on Performance Evaluation Methodologies and Tools, ACM Sigmetrics, ValueTools 2007, October 23-25, 2007, Nantes, France.

[5] H.Castel, L.Mokdad, N.Pekergin, ”Stochastic bounds applied to the end to end QoS in communication systems”, 15th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2007), October 24-26 2007, Bogazici University Istanbul, published by the IEEE Computer Society

[6] Clarke EM, Emerson A, Sistla A,P ,”Automatic verification of finite-state concurrent systems using temporal logic spacifications”, ACM Trans. on Programming Languages and Systems 8 (2) : 244-263, 1986

[7] M.Doisy, ”Comparaison de processus Markoviens”, PHD thesis, Univ. de Pau et des pays de l’Adour 92.

[8]J. M. Fourneau, N. Pekergin, ”An algorithmic approach to stochastic bounds”, in Performance Evaluation of Complex Systems: Techniques and Tools, LNCS 2459, 2002.

[9] J.M Fourneau, M. Lecoz, F. Quessette, ”Algorithms for irreducible and lumpable strong stochastic bound”, Linear Algebra and its Applications 386 (2004) 167-185, 2004.

[10] B.Haverkort,L.Cloth, H.Hermanns,J-P Katoen, C. Baiers, "Model checking performability properties" Proc. of IEEE DSN'02, 2002.

[11] T.Lindvall, "Lectures on the coupling method", Wiley series in Probability and Mathematical statistics, 1992.

[12] T. Lindvall, "Stochastic monotonicities in Jackson queueing networks", Prob. in the Engineering and Informational Sciences 11, 1997, 1-9.

[13] W. Massey, "Stochastic orderings for Markov processes on partially ordered spaces" Mathematics of Operations Research, Vol.12, N2, May 1987.

[14] H.G. Perros, "Queueing networks with blocking, exact and approximate solutions", Oxford university press, 1994.

[15] N. Pekergin, "Stochastic performance bounds by state space reduction", Performance evaluation, 36-37, (1-17), 1999.

[16] N.Pekergin, S.Younes, "Stochastic model checking with stochastic comparison", EPEW05, LNCS 3670, (109-124), 2005.

[17] D. Stoyan, "Comparison methods for queues and other stochastics models", J. Wiley and son, 1976.

[18] J.Sproston, S.Donatelli, "Backward bisimulation in Markov chain model checking" , IEEE Transactions on Software Engineering, Vol. 32, NO.8, August 2006.

[19] W.J. Stewart, "An Introduction to the Numerical Solution of Markov Chains", Princeton, 1993.

APPENDIX

A. Stochastic orderings theory

We present some theorems and definitions about stochastic orderings used for the generation of the aggregated bounding Markov chains.

Two formalisms can be used for the definition of a stochastic ordering : increasing functions [6,16] or increasing sets [12].

The $\preceq_{st}$ ordering is the most known stochastic ordering, it is equivalent to the sample path ordering (see Strassen's theorem [16]. Stochastic orderings are defined only on discrete and countable state space E, where a binary relation $\preceq$ is defined at least as a preorder [16]:

We consider two random variables X and Y defined on E, and their probability measures given respectively by the probability vectors p and q where $p[i] = Prob(X = i)$, $\forall i \in E$ (resp. $q[i] = Prob(Y = i)$, $\forall i \in E$). The $\preceq_{st}$ ordering can be defined using increasing functions as follows [16]:

Definition 1: $X \preceq_{st} Y \Leftrightarrow E[(f(X))] \preceq E[(f(Y))]$ $\forall f : E \rightarrow \mathbb{R}$, $\preceq$-increasing whenever the expectations exist.
Different methods are associated to the $\preceq_{st}$ ordering: the coupling [16], [10], and the increasing set theory [12].

We focus in this paper only on the $\preceq_{st}$ ordering. The key idea is to define a stochastic ordering from a family of increasing sets [12]. Let $\Gamma \subseteq E$, we denote by

$$\Gamma \uparrow = \{y \in E \mid y \succeq x, x \in \Gamma\}$$

Definition 2: Γ is called an increasing set if and only if $\Gamma = \Gamma \uparrow$

Let $\phi_{st}(E)$ the family of increasing sets which induces the $\preceq_{st}$ ordering:

$$\phi_{st}(E) = \{\text{all increasing sets on } E\}$$

The $\preceq_{st}$ ordering theorem using the increasing set theory [12] states as follows:

Theorem 1:

$$X \preceq_{st} Y \Leftrightarrow p \preceq_{st} q \Leftrightarrow p(\Gamma) \leq q(\Gamma), \forall \Gamma \epsilon \phi_{st}(E)$$

where

$$p(\Gamma) = \sum_{x \in \Gamma} p(x)$$

We present now the comparison of stochastic processes. Let $\{X(t), t \geq 0\}$ and $\{Y(t), t \geq 0\}$ stochastic processes defined on E.

Definition 3: We say that $\{X(t), t \geq 0\} \preceq_{st} \{Y(t), t \geq 0\}$

$$\text{if } \; X(t) \preceq_{st} Y(t), \forall t \geq 0$$

As the stochastic comparison of processes is defined as the stochastic comparison at any time of the processes, then it is also equivalent to :

$$E(f(X(t)) \leq E(f(Y(t)), \forall t \geq 0$$

$\forall f : E \rightarrow \mathbb{R}$, $\preceq$-increasing whenever the expectations exist.

Different methods can be used to define a stochastic ordering : increasing sets and coupling [12,16,10].

Next, we suppose that the Markov processes are not defined on the same state spaces. In this case, we can compare them on a common state space using mapping functions. The relevance of this technique is to reduce the state space of the Markov chains in order to define bounding aggregated Markov chains. Let :

- $X(t)$ a Markov process defined on E, we suppose that the stationary distribution Π_1 exists.
- $Y(t)$ a Markov process defined on $S \subset E$), we suppose that the stationary distribution Π_2 exists.
- g be a many to one mapping from E to S.

The stochastic comparison of these processes by mapping functions is defined as follows [6]:

Definition 4: We say that

$$\{g(X(t)), t \geq 0\} \preceq_{st} \{Y(t), t \geq 0\}$$

$$\text{if } \; g(X(t)) \preceq_{st} Y(t), \forall t \geq 0$$

If the state space $S \subset E$, then this theorem allows to compare the process $X(t)$ with the process $Y(t)$ defined on a smaller state space.

Proposition 1: If

$$\{g(X(t)), t \geq 0\} \preceq_{st} \{Y(t), t \geq 0\}$$

then :

$$\sum_{g(x)\in\Gamma} \Pi_1(x)\rho(x) \leq \sum_{x\in\Gamma} \Pi_2[x]\rho(x), \forall \Gamma \in \Phi_{st}(S)$$

where $\rho : E \rightarrow \mathbb{R}_{>0}$, an increasing function.

B. *Propositions, theorems proofs*

Proposition 2:

$$\sum_{x\in E_{yes}} \Pi[x] \leq \sum_{x|g(x)\in g(E_{yes})} \Pi[x]$$

Proof

$$\begin{aligned}\sum_{x|g(x)\in g(E_{yes})} \Pi[x] &= \sum_{x\in E_{yes}|g(x)\in g(E_{yes})} \Pi[x] \\ &+ \sum_{x\notin E_{yes}|g(x)\in g(E_{yes})} \Pi[x]\end{aligned}$$

as $\forall x \in E_{yes}, g(x) \in g(E_{yes})$, then clearly we have :

$$\sum_{x\in E_{yes}|g(x)\in g(E_{yes})} \Pi[x] = \sum_{x\in E_{yes}} \Pi[x]$$

and so we deduce that :

$$\sum_{x\in E_{yes}} \Pi[x] \leq \sum_{x|g(x)\in g(E_{yes})} \Pi[x]$$

Here we give a simple example about this proposition.

Example 1: Let $E = \{(0,0),(1,0),(0,1),(1,1)\}$, with the preorder component by component. The mapping $g : E \rightarrow S$ where $s \subset E$ is such that : $g(0,0) = (0,0), g(0,1) = g(1,0) = g(1,1) = (1,1)$. So $S = \{(0,0),(1,1)\}$.

For $E_{yes} = \{(0,1)\}$, then $g(E_{yes}) = \{(1,1)\}$, which is an increasing set, and $\sum_{x\in E_{yes}} \Pi[x] = \Pi[(0,1)] \leq \sum_{x\in E|g(x)\in g(E_{yes})} \Pi[x] = \Pi[(0,1)] + \Pi[(1,0)] + \Pi[(1,1)]$