COSMOS: a Statistical Model Checker for the Hybrid Automata Stochastic Logic

Paolo Ballarini*, Hilal Djafri†, Marie Duflot‡, Serge Haddad† and Nihal Pekergin‡
*INRIA Bretagne Atlantique †LSV - ENS Cachan ‡LACL - UPEC

***Abstract*—This tool paper introduces COSMOS, a statistical model checker for the Hybrid Automata Stochastic Logic (HASL). HASL employs Linear Hybrid Automata (LHA), a generalization of Deterministic Timed Automata (DTA), to describe accepting execution paths of a Discrete Event Stochastic Process (DESP), a class of stochastic models which includes, but is not limited to, Markov chains. As a result HASL verification turns out to be a unifying framework where sophisticated temporal reasoning is naturally blended with elaborate reward-based analysis. COSMOS takes as input a DESP (described in terms of a Generalized Stochastic Petri Net), an LHA and an expression Z representing the quantity to be estimated. It returns a confidence interval estimation of Z. COSMOS is written in C++ and is freely available to the research community.**

I. INTRODUCTION

Probabilistic model checking is concerned with the assessment of probabilistic statements regarding a probabilistic model. In the case of Markov models, efficient (symbolic) *numerical* methods have been developed and popular model checking tools, most notably PRISM [7], take advantage of them. Nevertheless *numerical* model checking suffers from state space explosion thus large models are intractable. On the other hand the *statistical* model checking approach, which does not require storage of a model's state space, has been gaining popularity in recent times. Dedicated statistical model checkers have been developed [5] [1] [4], while tools which were originally designed for numerical verification, such as PRISM and MRMC[3], have been then extended with statistical verification functionalities. Yet, there seem to be at least two important restrictions concerning existing verification tools for probabilistic models: their limited *markovian scope* (despite the fact that many systems cannot realistically be modeled in terms of exponentially distributed activities only) and the restrained expressivity of the logic used to assert the statement to be assessed.

In this paper we introduce COSMOS, a statistical verification tool which targets DESP (i.e. possibly infinite state, not-necessarily Markovian probabilistic models). COSMOS is based on a novel, very expressive logic, named HASL, which we briefly describe next.

II. HYBRID AUTOMATA STOCHASTIC LOGIC

HASL is a stochastic logic for DESP models introduced in [6]. A formula of HASL consists of a Linear Hybrid Automata (LHA) $\mathcal{A}$, characterizing the *accepted paths*, and an expression Z, describing the quantity to be measured. The output of HASL procedure is the estimation of the expectation of Z wrt the paths accepted by $\mathcal{A}$.The distinctive aspect of LHA-based verification is that LHA employ real valued variables (as opposed to Timed Automata's *clocks*), to store relevant statistics of the processed paths. This provides the modeler with a very expressive language to reason about a model, a language through which relevant behaviors can be characterized by any combination of timing conditions, events-occurrence conditions and cost/reward conditions. If x is a data-variable of an LHA, then typical HASL expressions include, for example: $Z \equiv E[last(x)]$ (for estimating the expectation of the last value assumed by x on an accepting path), $Z \equiv E[avg(x)]$ (for estimating the expectation of the average value assumed by x on an accepting path), etc. Note that the assessment of a measure of probability (which is the probabilistic model checking problem) is obtained, in HASL terms, through $Z \equiv E[last(x)]$ where x is binomial random variable set to 1 on acceptance of a path and to 0 on rejection.

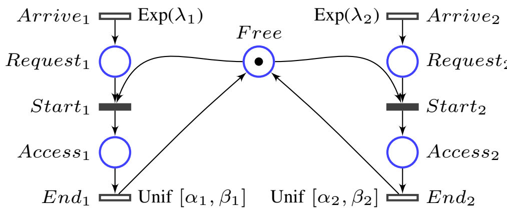

Fig. 1. Infinite-state GSPN model of a shared memory system.

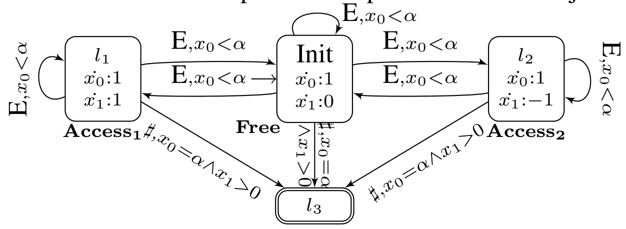
E,$x_0<\alpha$
l_1 $\dot{x_0}$:1 $\dot{x_1}$:1
Init $\dot{x_0}$:1 $\dot{x_1}$:0
l_2 $\dot{x_0}$:1 $\dot{x_1}$:-1
Access₁
Free
Access₂
$\sharp$,$x_0=\alpha \wedge x_1>0$
$\sharp$,$x_0=\alpha$
$x_1 > 0$
l_3

Fig. 2. An LHA for measures related to the difference of memory usage

The LHA in Figure 2 is an example automaton for assessing measures related to the utilization difference between 2 classes of users sharing a mutual exclusive resource (see the DESP model depicted, in Petri-net form, in Figure 1). Such LHA uses a (global) clock variable (x_0) and a variable (x_1) for recording the utilization difference. x_1 grows with rate 1 when the resource is held by class 1 users (location l_1), is decremented with rate -1 when class 2 users hold the resource (l_2) and doesn't change when the resource is free ($Init$). A path is accepted ($l3$) only if at time $x_0 = \alpha$ the resource has been held longer by class 1 users ($x_1 \geq 0$). $Z \equiv E[max(x_1)]$ is an example of relevant HASL expression for this case: it allows

PROBABILITY first queue is full (Tandem Queuing System, bounded queue length $C = 5$)										AVERAGE WAITING-TIME of station 1			
	probability measure			gen. paths gain	runtime (sec)			runtime-gain			waiting-time measure		
T	PRISM-n	PRISM-s	COSMOS	COSMOS	PRISM-s	COS-G++	COS-LLVM	COS-G++	COS-LLVM	T	PRISM-n	PRISM-s	COSMOS
20	0.33574	-0.00137	+0.0001	+35.89%	18.64	51.61	14.63	- 176.87%	+21.51%	10	1.08484	+0.01778	+0.00197
40	0.56931	+0.00061	+0.00126	+29.59%	30.03	89.12	24.59	-196.76%	+18.11%	20	2.49446	+0.0634	+0.01604
80	0.82229	+0.00351	+0.0064	+57.84%	42.54	73.73	20.22	-73.31%	+52.46%	50	6.73012	+0.20522	-0.00962

TABLE I
COMPARING COSMOS VS PRISM: ESTIMATION OF A MEASURE OF PROBABILITY AND OF A MEASURE OF WAITING-TIME

for measuring the (expected value of the) maximum of the utilization time difference (between class 1 and class 2 users) on condition that, at time $t = \alpha$ the difference is positive. Figure 3 depicts the trend of $E[max(x_1)]$ as a function of the time bound α and for different arrival rate (λ_2) for class 2 users. Note that HASL verification is (currently) limited to the assessment of finite-horizon (either time-bounded or event-occurrence bounded) properties of a DESP.

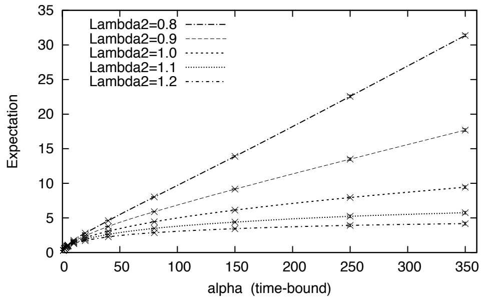

Fig. 3. Maximum of the difference for memory occupation times

III. COSMOS TOOL

COSMOS is implemented in C++ and relies on the BOOST libraries for random number generation functionalities. Currently two versions of the tool are available, one suitable for compilation with the G++ compiler, the other suitable to optimized compilation through the LLVM compiler infrastructure [2].

Inputs. COSMOS takes as input a DESP (represented in terms of a Generalized Stochastic Petri Net (GSPN)[1]), and an LHA, both stored into (formatted) textual files (extension .gspn and .lha respectively). On execution COSMOS prompts for the expression Z to be estimated, plus the estimation parameters: the confidence level ϵ, the interval width δ. It returns the estimated value $\overline{Z}$. The tool is realized according to a *model driven code generation* scheme (Figure 4): the input model $\mathcal{D}$, and the input automaton $\mathcal{A}$ are first processed and tailored C++ code is generated which then yields the executable for the HASL statistical model checker for model $\mathcal{D}$ and automaton $\mathcal{A}$. Such executable is then used to calculate the Z expressions provided as input on execution.

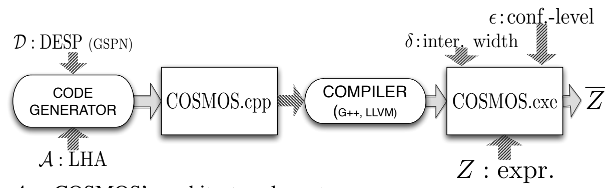

Fig. 4. COSMOS's architecture layout

[1]extended with generally distributed timed-transitions.

Tool evaluation. We performed a number of experiments aimed at evaluating COSMOS both in terms of accuracy and runtime. We considered two popular workbench models available at [7]: a Tandem Queuing System (TQS) and a Cyclic Server Polling System (CSPS). We encoded the models and the corresponding (time-bounded) properties in COSMOS and run experiments both with COSMOS and PRISM. An excerpt of the results is reported in Table I. The leftmost part of the table refers to the evaluation of a measure of probability of the TQS whereas the rightmost part refers to the evaluation of the average waiting-time for one station of the CSPS. To assess the accuracy of COSMOS we compared its output with that produced by PRISM via both its *numerical* engine (PRISM-n) and its *statistical* engine (PRISM-s)[2]: results indicate that COSMOS is comparably accurate to PRISM-s in estimating measures of probability whereas it appears more accurate in estimating non-probability measures (see waiting-time estimations). With respect to runtime COSMOS-LLVM appears (up to twice) faster than PRISM whereas COSMOS-G++ (up to twice) slower than PRISM. When confronting runtimes it should be noted that: simulation of the synchronized $\mathcal{D} \times \mathcal{A}$ process (HASL), is inherently costlier than simulation of a CTMC (CSL); such extra cost is compensated by the fact that COSMOS (which establishes when to stop paths generation at run-time) requires a substantially smaller number of runs than those required (and statically determined) by PRISM.

Development. We plan to evolve COSMOS in several respects, including: (1) accelerating the path generation when faced to difficult acceptance condition via the *rare event* approach; (2) improve the usability by: supporting the execution of batches of experiments (short-term), adding graphical support for definition of GSPN model and LHA (long-term). The (provisional) web-page for COSMOS is http://www.lsv.ens-cachan.fr/~djafri/cosmos/.

REFERENCES

[1] APMC home page: http://sylvain.berbiqui.org/apmc.
[2] LLVM home page. http://llvm.org/.
[3] MRMC home page. http://www.mrmc-tool.org/trac/.
[4] VESTA home page: http://osl.cs.uiuc.edu/ ksen/vesta2/.
[5] YMER home page. http://www.tempastic.org/ymer/.
[6] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin. HASL : an expressive language for statistical verification of stochastic models. In *ValueTOOLS'11*. To appear.
[7] PRISM home page. http://www.prismmodelchecker.org.

[2]experiments executed with confidence-level=99.99%, interval-width=0.01 both for COSMOS and PRISM-s.