Time for Statistical Model Checking of real-time systems [⋆]

Alexandre David[1], Kim G. Larsen[1], Axel Legay[2], Marius Mikučionis[1], Zheng Wang[3]

[1] Computer Science, Aalborg University, Denmark
[2] INRIA/IRISA, Rennes Cedex, France
[3] Shanghai Key Laboratory of Trustworthy Computing, Software Engineering Institute, East China Normal University, China

Abstract. We propose the first tool for solving complex (some undecidable) problems of timed systems by using Statistical Model Checking (SMC). The tool monitors several runs of the system, and then relies on statistical algorithms to get an estimate of the correctness of the entire design. Contrary to other existing toolsets, ours relies on i) a natural stochastic semantics for networks of timed systems, ii) an engine capable to solve problems that are beyond the scope of classical model checkers, and iii) a friendly user interface.

1 Context

Timed model checking (TMC) is a technique used to prove the absence of bugs in systems whose behaviors depend on real or discrete time constraints. The approach has been implemented in several tools [4,2,5] capable of handling case studies of industrial size. Unfortunately, many applications are still out of scope of TMC. This is due to the complexity of the timed behaviors, which can even make the problem undecidable.

In a recent work [8], we presented Constant Slope Timed Automata (CSTA), that are timed systems in that clocks may have different rates (even potentially negative) in different locations. Such automata are as expressive as linear hybrid automata or priced timed automata, but the addition of features such as input and output modalities allows us to specify complex problems in an elegant manner. Unfortunately most of such problems are either undecidable or too complex to be solved with classical model checking approaches. In [8], we proposed to estimate undecidable problems by using Statistical Model Checking (SMC) [13,9]. SMC consists of monitoring some runs of the system and then uses a statistical algorithm to obtain an estimate for the system. Such simulation-based techniques were applied in other contexts where they outperformed classical model checking techniques with an order of magnitude [13,14,1].

[⋆] Work partially supported by VKR Centre of Excellence – MT-LAB, an "Action de Recherche Collaborative" ARC (TP)I, the Foundation for Abroad Studies from East China Normal University, and NFSC No. 90818024.

To apply SMC on CSTA, we had to define a stochastic semantics on their behaviors. This was done in a very natural manner by adding distributions on the delay before a transition is taken. Those distributions are uniform if the delays are bounded, and exponential otherwise. The semantics then establishes a race between the components and selects the smallest delay. One of its major advantages is that the composition of several timed systems remains a pure stochastic system, not a Markov Decision Process. The latter is needed to apply SMC[4].

In this paper, we report on an implementation of our work within the Uppaal toolset [2]. One of the major differences with classical Uppaal is the introduction of a new user interface that allows to specify CSTAs with respect to a stochastic semantics — such semantics is naturally needed to apply SMC. Another contribution is the implementation of several versions of the sequential hypothesis testing algorithm of Wald [12]. Contrary to other implementations of SMC [13,10,7], we also consider those tests that can compare two probabilities without computing them. Finally, contrary to other SMC-based tools, our tool comes with a wide range of functionalities that allow the user to visualize the results in the form of, e.g., probability distributions, evolution of the number of runs with time bounds, or computation of expected values.

Related work. Related work includes the very rich framework of stochastic timed systems of MoDeST [3]. Here, however, general hybrid variables are not considered and parallel composition do not yield fully stochastic models. For the notion of probabilistic hybrid systems considered in [11] the choice of time is resolved non-deterministically rather than stochastically as in our case and as required by SMC.

2 The Toolset

User Interface. Our extension supports the rich modeling constructs of Uppaal with additions specific to CTSA. We add a rational expression attached to locations to define the exponential rates for choosing (unbounded) delays stochastically. We add branching edges and associated weights for the probabilistic extension as shown in Fig. 1. We also generalize rates on clocks to be expressions that take value over integers (even negative) compared to just 0 and 1 for Uppaal.

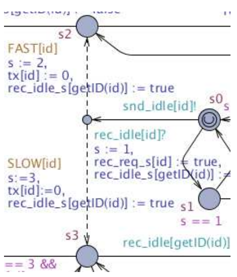


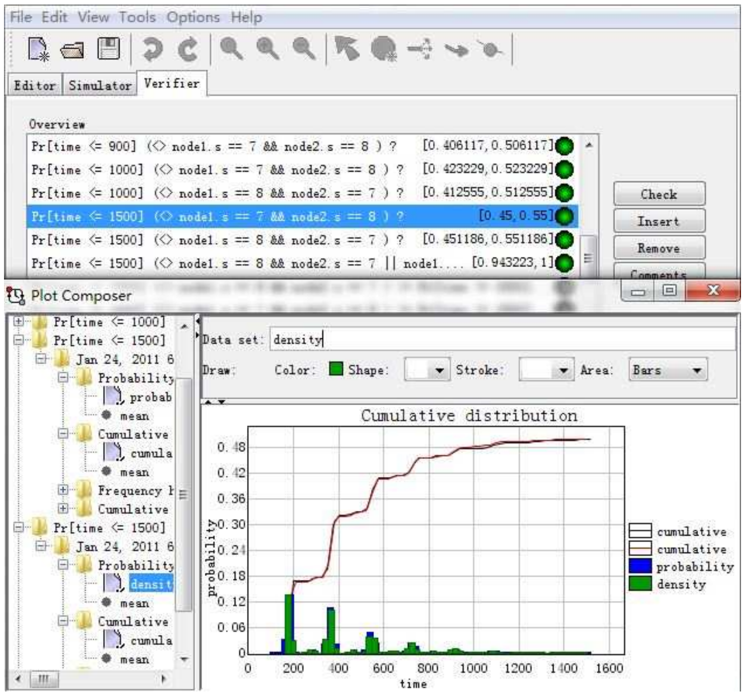
Fig. 1: Branching edges (from firewire case-study).

The verifier shows the estimated intervals of probabilities and provides a plot composer to visualize and compare different results. Figure 2 shows a screenshot with the verifier (above) and the plot composer (below). The verifier provides additional results in a form of plotted data which are accessible via a

[4] One could try to use classical heuristics for removing non determinism, but they are generally not easily applicable to timed systems.

popup-menu by right-clicking over the property. Any plot can be exported to a number of graphical formats and data saved in a textual format.

In addition, a custom plot can be created in plot composer accessible via Tools menu. On the left side of the plot composer, the data sets are grouped and displayed in a tree structure. Any data set can be selected for the composite plot by double clicking on its node and the same way de-selected. The details of the plot can be customized on the right side above the plot.



```
Pr[time <= 900] (<> node1.s == 7 && node2.s == 8 ) ?    [0.406117,0.506117]
Pr[time <= 1000] (<> node1.s == 7 && node2.s == 8 ) ?   [0.423229,0.523229]
Pr[time <= 1000] (<> node1.s == 8 && node2.s == 7 ) ?   [0.412555,0.512555]
Pr[time <= 1500] (<> node1.s == 7 && node2.s == 8 ) ?             [0.45,0.55]
Pr[time <= 1500] (<> node1.s == 8 && node2.s == 7 ) ?   [0.451186,0.551186]
Pr[time <= 1500] (<> node1.s == 8 && node2.s == 7 || node1....  [0.943223,1]
```

Data set: density
Draw: Color: 🟩 Shape: Stroke: Area: Bars

Fig. 2: The verifier shows the results of the different probabilistic queries and the plot composer shows probability distributions.

Simulation-based Engine. Any SMC implementation is divided into three parts. First, one needs an algorithm capable of generating random runs, in our case according to our stochastic semantics. Second, those runs have to be monitored with respect to some property. Finally one needs a statistic algorithm to get a general confidence on the results. We implement a new engine to generate such runs that works on states with discrete clock valuations, which makes the computations cheaper compared to equivalent symbolic operations. The runs are bounded by either time, cost, or a number of discrete steps. The engine monitors the generated run with respect to a set of properties that are being checked. Currently the tool is capable of monitoring some properties written in cost-constrained temporal logic ("can I reach a from b with a cost less than 5?"), but in the future the monitoring procedure could be generalized. Runs are stopped when such properties hold. To do this, the algorithm is able to compute the relative upper bound of an invariant from a given state or compute the delay needed to satisfy a guard or more generally predicates in our properties.

Properties are evaluated on bounded runs by `time`. In fact `time` may be replaced by `steps` or by a clock, or by a cost constraint. The bound is a constant value. The expressions `expr` are state predicates. Our tool can answer of the three following questions:

– A qualititive check: `Pr[time<=bound](<> expr) >= p`.
– A quantitative check: `Pr[time<=bound](<> expr) ?`.
– A comparison check
`Pr[time1<=bound1](<> expr1) >= Pr[time2<=bound2](<> expr2)`.

The first formula is to check if the probability of satisfying some property is at least `p`. The second one estimates this probability within an interval. The last one compares two probabilities without evaluating them and gives the result for all bounds up to `bound1` if both bounds are the same. All these checks rely on some statistic algorithm to estimate the correctness by observing runs of the system. The qualitative check is an extension of the sequential hypothesis testing of Wald, the quantitative checks estimates the probability with a Monte-Carlo based approach, finally the comparison check is an extension of sequential hypothesis testing that allows to compare probabilities without computing them. The algorithms are precise up to a certain value that can be chosen by the user.

3 Case Studies

We present two case-studies to highlight the features of our tool. More can be found in *http://www.cs.aau.dk/~adavid/smc/*, where we handle jobshop scheduling problem and show that our tool performs better than Prism.

3.1 Firewire Protocol

We consider the IEEE 1394 High Performance Serial Bus ("FireWire" for short) that is used to transport video and audio signals on a network of multimedia devices. The protocol has two modes, one *fast* and one *slow* mode for the nodes. The model defines weights to enter these modes as shown in Fig. 1.

This is a leader election protocol that we model with two nodes. We compute the probability for node 1 to become the root (or leader) within different time bounds. We use variable s to denote the state of a node. At initialization, every node is in the *contention* state $s = 0$. After a sequence of steps, a node will enter the *root* state $s = 7$ and the other node will enter the *child* state $s = 8$. The query formula is `Pr[time <= 1500] (<> node1.s==7 && node2.s==8) ?`.

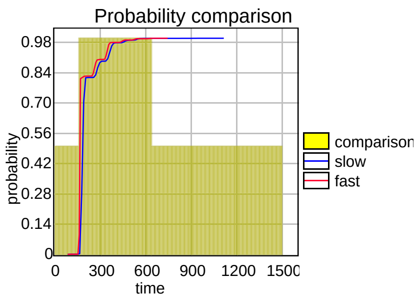


Fig. 3: Probability Comparison

We are also interested in checking whether there is a difference between the probability that a *fast* node becomes the root or the child and the probability that a *slow* node does. We use a probability comparison property for this purpose.

The results in the plot composer of Fig. 2 show that the probability to elect node 1 as the root increases with the time bound. This probability density diagram shows that in most cases, node 1 may become the root after around 200ms. Uppaal confirms that the fastest possible to reach this state is 164ms. The result in Fig. 3 shows that at the beginning the probabilities are indistinguishable, then the *fast* node has higher probability to become a *root* or a *child*, and at the end the probabilities become very close again.

3.2 Bluetooth Protocol

Bluetooth is a wireless telecommunication protocol using frequency-hopping to cope with interference between the devices in the network. A device can be in a scan state where it replies to a request after two time slots (a slot is 0.3125 ms) and goes to a reply state where it waits for a random amount of time before coming back to the scan state. When a device stays in the scan state, it can also enter the sleeping state (2012 time slots) to save energy. We model energy consumption with a clock (called energy) for which we change the rate depending on these states.

We check the following properties:

– We evaluate the probability of replying within 70000 time units:
  `Pr[time<=70000](<> receiver1.Reply) ?`
  The result is between $[0.866977, 0.966977]$.
– We evaluate the probability of letting time pass 70000 time units with a limited energy budget:
  `Pr[energy<=4000] (<> time>=70000) ?`
  The result is between $[0.949153, 1]$.

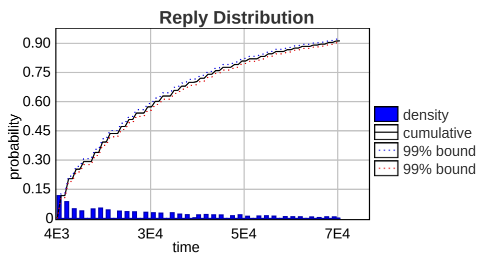
In both cases the tool is able to compute a distribution of the probability over the bound given in argument. Fig. 4(a) shows the cumulative probability of successfully communicating with time. Fig. 4(b) shows that it costs at least $2,400$ energy units. The plot shows how bluetooth consumes energy.

4 Conclusion

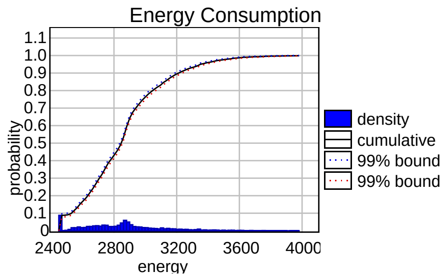
We presented an extension of Uppaal for CSTA. Our tool handles problems that are out of scope of existing tools for SMC and timed stochastic systems. We are currently implementing a Bayesian extension of our work following the theory in [6]. We are also working on an extension that allows to handle nested probabilistic operators and unbounded cost constraints formulas.

References

1. A. Basu, S. Bensalem, M. Bozga, B. Caillaud, B. Delahaye, and A. Legay. Statistical abstraction and model-checking of large heterogeneous systems. In *FORTE*, volume 6117 of *LNCS*, pages 32–46. Springer, 2010.



(a)



(b)

Fig. 4: Probability to reply and energy consumption with 99% confidence.

2. G. Behrmann, A. David, and K. G. Larsen. A tutorial on Uppaal. In M. Bernardo and F. Corradini, editors, *SFM*, ncs(3185), pages 200–236. Springer, 2004.
3. H. Bohnenkamp, P. D'Argenio, H. Hermanns, and J.-P. Katoen. Modest: A compositional modeling formalism for real-time and stochastic systems. Technical Report CTIT 04-46, University of Twente, 2004.
4. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A model-checking tool for real-time systems. In *Proc. 10th Int. Conference on Computer Aided Verification (CAV)*, volume 1427 of *Lecture Notes in Computer Science*, pages 546–550. Springer, 1998.
5. T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond hytech: Hybrid systems analysis using interval numerical methods. In *Proc. 3rd Int. Workshop on Hybrid Systems: Computation and Control (HSCC)*, volume 1790 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 2000.
6. S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A bayesian approach to model checking biological systems. In *CMSB*, volume 5688 of *LNCS*, pages 218–234. Springer, 2009.
7. J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker MRMC. In *Proc. of 6th Int. Conference on the Quantitative Evaluation of Systems (QEST)*, pages 167–176. IEEE Computer Society, 2009.
8. D. Poulsen, A. David, K. G. Larsen, A. Legay, M. Mikucionis, J. V. Vliet, and W. Zheng. Efficient statistical model checking for constant slope timed i/o automata. Technical report, Aalborg University, 2011. Submitted for publication.
9. K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, LNCS 3114, pages 202–215. Springer, 2004.
10. K. Sen, M. Viswanathan, and G. A. Agha. Vesta: A statistical model-checker and analyzer for probabilistic systems. In *QEST*, pages 251–252. IEEE Computer Society, 2005.
11. T. Teige, A. Eggers, and M. Fränzle. Constraint-based analysis of concurrent probabilistic hybrid systems: An application to networked automation systems. *Nonlinear Analysis: Hybrid Systems*, In Press, Corrected Proof:–, 2010.
12. R. Wald. *Sequential Analysis.* Dove Publisher, 2004.
13. H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events.* PhD thesis, Carnegie Mellon, 2005.
14. H. L. S. Younes, M. Z. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.