mctau: Bridging the Gap between Modest and UPPAAL[⋆]

Jonathan Bogdoll[2], Alexandre David[1], Arnd Hartmanns[2], and Holger Hermanns[2]

[1] Aalborg University, Department of Computer Science, Aalborg, Denmark
[2] Saarland University – Computer Science, Saarbrücken, Germany

Abstract. Modest is a high-level compositional modelling language for stochastic timed systems with a formal semantics in terms of stochastic timed automata, an overarching formalism of which several well-studied models are special cases. The emphasis of Modest is to make use of existing analysis techniques and tools in a single-formalism, multiple-solution approach. In this paper, we focus on networks of timed automata as supported by Uppaal. We report on extensions made to Modest and Uppaal that allow the transformation of a rich subset of Modest models to Uppaal timed automata and enable connections to further tools and formalisms. We present our Modest-to-Uppaal tool chain mctau, which allows both a fully automated analysis as well as model transformation, and we compare its performance with the existing mcpta tool.

1 Introduction

Modest, the "modelling and description language for stochastic timed systems" [6], is a compositional modelling language that combines expressive and powerful syntax-level features—such as recursive process definitions, loops, arrays, exception handling and user-defined data structures—with a formal semantics in terms of stochastic timed automata (STA). STA span a very rich spectrum of semantic models, supporting continuous and discrete probability distributions as well as nondeterminism. Well-known and extensively studied submodels of STA are probabilistic timed automata (PTA) [15], timed automata (TA) [1], and generalised semi-Markov processes (GSMP) [11]. Most of the submodels are easily identifiable on the syntactic level.

Modest has been used in a wide variety of application studies, ranging from wireless sensor networks [2,18] and communication protocols [13] to architectural dependability models [5], industrial production scheduling [16] and electric power grid management [4]. The principle idea behind the formalism and its supporting tools is to provide a *single-formalism, multiple-solution* approach to modelling and analysis, using existing analysis engines and algorithms where available to avoid unnecessary reimplementations.

⋆ This work has been supported by the European Union FP7-ICT projects Quasimodo, contract no. 214755, and MEALS, contract no. 295261, by the DFG as part of SFB/TR 14 AVACS, and by the DFG/NWO Bilateral Research Program ROCKS.

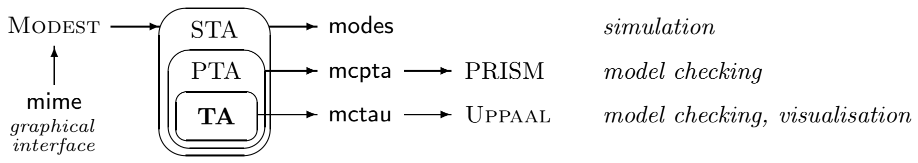

Fig. 1. How the new mctau tool fits in the MODEST TOOLSET

Started in 2008, the MODEST TOOLSET constitutes the second generation [7] of tools with this philosophy, currently including (i) mcpta [12,13], which enables model checking of networks of PTA using the PRISM [14] probabilistic model checker in the background, (ii) modes [5,12], a discrete-event simulator that primarily targets GSMP models but in fact is enhanced to handle certain nondeterministic models in a sound way, and (iii) mime as a graphical user interface that seamlessly integrates the analysis tools into a MODEST source code editor with syntax and error highlighting. The tools are usable and robust.

In this paper, we present a new member of the MODEST TOOLSET family: mctau, providing visualisation and analysis of networks of timed automata. It does so by connecting to the real-time model checker UPPAAL [3]. Although mcpta already includes support for TA as a special case of PTA, we will see that mctau allows a more efficient analysis; in addition to this, the way we bridge several semantic gaps between MODEST and UPPAAL will be of practical use beyond just the mctau tool. Figure 1 provides a toolset overview.

2 Bridging the Gap

A connection between MODEST and UPPAAL had been planned for a long time [7], but several fundamental differences between the two modelling languages have prevented this up to now:

Time Constraints. Constraints on the flow of time are specified as location *invariants* in UPPAAL, while MODEST uses *deadlines* (or *urgency constraints* [8]). As an example, if location l has invariant $c \leq 3$ (where c is a clock variable), time can pass while in l as long as $c \leq 3$ holds. Deadlines, on the other hand, are associated to edges in an automaton, and specify that *some* edge must be taken out of a location once the deadline of *one* outgoing edge becomes satisfied. Invariant $c \leq 3$ can thus be expressed as deadline $c \geq 3$ on some edge leaving l.

Deadlines are more flexible in parallel composition and synchronisation, easily allowing, for example, a synchronising edge to be taken *as soon as possible* in all components. While there are deadlines that cannot be transformed into a single invariant (mainly equality comparison deadlines like $c = 3$ and equivalents) and vice-versa, we have recently shown how to transform all practically relevant deadlines into invariants [12], and this transformation is implemented in mctau.

Assignments. The assignments associated to an edge in UPPAAL are performed sequentially: $x := y, y := x$ will result in x and y both having the same value.

In MODEST, variables are assigned new values in function of their previous ones atomically. $x := y, y := x$ will thus result in swapping x and y. UPPAAL 4.1.5 now implements this semantics as an option (`-M` at the command line and the `Modest` checkbox in the option menu of the graphical interface).

Synchronisation. Both UPPAAL and MODEST support the notion of *parallel composition*, where a number of independent processes run in parallel. However, the synchronisation mechanisms differ fundamentally: MODEST supports a CSP/LOTOS-style *multi-way* synchronisation where processes synchronise on edges with the same action label that is part of the intersection of the action alphabets α of the processes. For example, if $\alpha(P_1) = \{a, b\}$ and $\alpha(P_2) = \{b, c\}$ then P_1 (P_2) is free to take action a (c), but P_1 and P_2 must synchronise to take action b. UPPAAL, on the other hand, provides CCS-style *binary* synchronisation where exactly two processes synchronise on a matching pair of actions (e.g. `a!` and `a?`) and I/O-automata inspired *broadcast* synchronisation where all processes able to perform an `a?` action synchronise with one sender performing `a!`.

Although it is possible, with some effort, to encode binary using only multi-way synchronisation [17], we are not aware of any way to do the opposite in a semantically sound way and without introducing additional intermediate states in ways that would make the state-space explosion problem significantly worse. We thus resolved this discrepancy in a practical manner by adding multi-way synchronisation to UPPAAL 4.1.5 and extending MODEST with broadcast and binary synchronisation. These extensions also open UPPAAL and MODEST for a large number of further tool connections that were previously infeasible such as connecting UPPAAL with CSP-style tools, notably CADP [10] or PRISM.

3 The mctau Tool

mctau is our new addition to the MODEST TOOLSET that, at its core, performs the translation of MODEST models to the XML-based input language of UPPAAL, including advanced features of MODEST such as user-defined functions and data types. It supports all types of properties that are already supported by UPPAAL. mctau is available as a command-line executable and as a fully-integrated analysis engine inside mime. It has two modes of operation:

Export Mode: A `.modest` input file is transformed into a `.xml` file with the automata and a `.q` file with the properties to be analysed. These can be opened in the UPPAAL graphical interface for analysis or further modification.

Analysis Mode: UPPAAL is completely hidden from the user: The model transformation as well as the analysis of the properties, using UPPAAL's command-line `verifyta` executable, is performed by mctau in a fully automated way. This is also the way that mctau is used from within mime.

Since MODEST is a text-based formalism while UPPAAL is based on a graphical automata notation, mctau incorporates a set of graph layout algorithms, based on

```
action put, get;
process Channel() {
   clock c;
   put {= c = 0 =};
   invariant(c <= TD) alt {
      :: get
      :: tau
   }; Channel() }
```

⇒

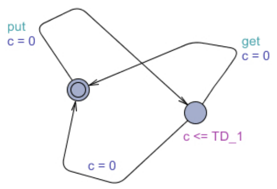

Fig. 2. A Modest process and its Uppaal automaton

the Graph# library[1] and adapted for timed automata with all their location and edge labels, to generate easily useable Uppaal models. Figure 2 shows a simple communication channel in Modest and the Uppaal automaton generated by mctau based on the *LinLog* layout algorithm.

Aside from TA, mctau can also cope with networks of PTA: When given a model with probabilistic branching, mctau generates (and analyses) an overapproximation that is obtained by replacing all probabilistic with nondeterministic branching (making sure to discard branches with probability zero). This neither adds nor removes paths through the model, but all probabilities are lost. Still, it is useful for a fast qualitative analysis. mctau then also replaces probabilistic properties by a set of purely timed ones to determine whether the probability is exactly zero or one. For example, property $P_{\max}(\Diamond\, e)$ to determine the maximum probability (over all schedulers) of eventually reaching a state satisfying expression e is replaced by $\forall\,\Box\,\neg e$ and $\forall\,\Diamond\, e$: If the first property is satisfied, the original probability must be zero; if the second property is satisfied, it must be one; otherwise, it may be any number in the *closed* interval $[0, 1]$.

This handling of PTA models greatly improves the usability and applicability of mctau since it allows the user to write a single model to subsequently use three different tools—mctau, mcpta and modes—with vastly different background technologies, all of that optionally within the graphical interface of mime.

Tool availability. The Modest Toolset, which includes mctau, and Uppaal are both freely available for academic users at `www.modestchecker.net` and `www.uppaal.org`, respectively.

4 Evaluation

mctau is able to analyse (the nondeterministic overapproximations of) the three original mcpta PTA case studies [13], without requiring any changes to the models. In all cases where mctau reports probability 0 or 1, mcpta does so as well. For the BRP model in particular, we see that whenever mctau reports $[0, 1]$, the actual probability as reported by mcpta is in $]0, 1[$, as shown in Table 1 (model parameters (N, MAX, TD) and property names are as in [13]). This shows how mctau can be of great help in model debugging and for sanity checking of probabilistic models.

[1] `http://graphsharp.codeplex.com/`

Table 1. Results of mctau and mcpta for the probabilistic BRP model $(16, 2, 1)$

property	T_{A1}	T_{A2}	P_A	P_B	P_1	P_2	D_{max}
mctau	*true*	*true*	0	0	$[0,1]$	$[0,1]$	$[0,1]$
mcpta	*true*	*true*	0	0	$4.233 \cdot 10^{-4}$	$2.645 \cdot 10^{-5}$	$9.996 \cdot 10^{-1}$

Table 2. Performance of mctau and mcpta on the nonprobabilistic BRP model

		standard properties			time-bounded properties		
tool	model	states	time	memory	states	time	memory
mctau	$(16, 2, 1)$	880	1 s	27 MB	831	1 s	19 MB
(using UPPAAL)	$(64, 5, 4)$	8 317	2 s	30 MB	8 091	1 s	21 MB
mcpta	$(16, 2, 1)$	3 972	2 s	167 MB	170 371	20 s	253 MB
(using PRISM)	$(64, 5, 4)$	304 785	13 s	187 MB	4 914 666	284 s	686 MB

The BRP model has also been studied as a pure TA model before [9] with some properties that had not been transferred to the PTA model. We were able to reconstruct that TA model in MODEST and check all properties with mctau. The corresponding model file is included in the MODEST TOOLSET download. We also compared the performance of mctau and mcpta (using the digital clocks engine[2]) on a nonprobabilistic version of the original MODEST BRP model. Table 2 summarises the results[3]; as expected (since mcpta/PRISM are not designed for nonprobabilistic models), the more specialised tool shows significantly improved performance.

5 Conclusion

We have presented mctau, a tool providing a link between the MODEST and UPPAAL modelling formalisms. The newly established connection opens the door to a powerful tool chain that gives MODEST modellers access to the editor and simulator of UPPAAL and reinforces the single-formalism, multiple-solution approach of MODEST. This approach might one day provide a possible solution to one of the obstacles that, in our experience, new users seeking to apply model-checking in their subject area face: the daunting number of different modelling languages which makes for low flexibility and a steep learning curve.

mctau was only possible because of recent results and implementation efforts that allowed the semantic gap between the two formalisms to be overcome. The implemented bridge spans a practically disturbing gap between CCS and I/O automata on the one side and CSP and LOTOS on the other. The inclusion of multi-way synchronisation in UPPAAL 4.1.5 is a key enabler for further connections with prominent verification tools such as PRISM or CADP.

[2] Use of PRISM's game-based engine was not possible due to its restrictions concerning the use of global variables and the access to other modules' local variables.

[3] Linux VM on Intel Core i5, `/usr/bin/time -v` for time and memory measurement; "states" is the number of zones explored by UPPAAL for mctau and the number of reachable discrete states (including discretised clock valuations) for mcpta.

Uppaal nowadays also contains an efficient statistical model checking engine, which we currently do not make use of since it relies on an entirely new and different semantics for timed automata. An investigation of the relationship between this "stochastic" semantics and Modest is planned as future work.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. TCS 126(2), 183–235 (1994)
2. Baró Graf, H., Hermanns, H., Kulshrestha, J., Peter, J., Vahldiek, A., Vasudevan, A.: A verified wireless safety critical hard real-time design. In: WoWMoM. IEEE (2011)
3. Behrmann, G., David, A., Larsen, K.G.: A Tutorial on Uppaal. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 200–236. Springer, Heidelberg (2004)
4. Berrang, P., Bogdoll, J., Hahn, E.M., Hartmanns, A., Hermanns, H.: Dependability results for power grids with decentralized stabilization strategies. Reports of SFB/TR 14 AVACS 83 (2012) ISSN: 1860-9821, www.avacs.org
5. Bogdoll, J., Ferrer Fioriti, L.M., Hartmanns, A., Hermanns, H.: Partial Order Methods for Statistical Model Checking and Simulation. In: Bruni, R., Dingel, J. (eds.) FORTE 2011 and FMOODS 2011. LNCS, vol. 6722, pp. 59–74. Springer, Heidelberg (2011)
6. Bohnenkamp, H.C., D'Argenio, P.R., Hermanns, H., Katoen, J.-P.: MoDeST: A compositional modeling formalism for hard and softly timed systems. IEEE Transactions on Software Engineering 32(10), 812–830 (2006)
7. Bohnenkamp, H.C., Hermanns, H., Katoen, J.-P.: motor: The modest Tool Environment. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 500–504. Springer, Heidelberg (2007)
8. Bornot, S., Sifakis, J.: An algebraic framework for urgency. Inf. Comput. 163(1), 172–202 (2000)
9. D'Argenio, P.R., Katoen, J.-P., Ruys, T.C., Tretmans, J.: The Bounded Retransmission Protocol Must be on Time! In: Brinksma, E. (ed.) TACAS 1997. LNCS, vol. 1217, pp. 416–431. Springer, Heidelberg (1997)
10. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: CADP 2010: A Toolbox for the Construction and Analysis of Distributed Processes. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS 2011. LNCS, vol. 6605, pp. 372–387. Springer, Heidelberg (2011)
11. Haas, P.J., Shedler, G.S.: Regenerative generalized semi-Markov processes. Communications in Statistics. Stochastic Models 3(3), 409–438 (1987)
12. Hartmanns, A.: Model-Checking and Simulation for Stochastic Timed Systems. In: Aichernig, B.K., de Boer, F.S., Bonsangue, M.M. (eds.) FMCO. LNCS, vol. 6957, pp. 372–391. Springer, Heidelberg (2011)
13. Hartmanns, A., Hermanns, H.: A Modest approach to checking probabilistic timed automata. In: QEST, pp. 187–196. IEEE Computer Society (2009)
14. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
15. Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. TCS 282(1), 101–150 (2002)

16. Mader, A., Bohnenkamp, H.C., Usenko, Y.S., Jansen, D.N., Hurink, J., Hermanns, H.: Synthesis and stochastic assessment of cost-optimal schedules. STTT 12(5), 305–318 (2010)
17. Norman, G., Palamidessi, C., Parker, D., Wu, P.: Model checking the probabilistic pi-calculus. In: QEST, pp. 169–178. IEEE Computer Society (2007)
18. Yue, H., Bohnenkamp, H., Kampschulte, M., Katoen, J.-P.: Analysing and Improving Energy Efficiency of Distributed Slotted Aloha. In: Balandin, S., Koucheryavy, Y., Hu, H. (eds.) NEW2AN 2011 and ruSMART 2011. LNCS, vol. 6869, pp. 197–208. Springer, Heidelberg (2011)