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" [5], is a compositional modelling language that combines expressive and powerful syntax-level features with a formal semantics in terms of stochastic timed automata (STA). STA span a rich spectrum of semantic models, supporting continuous and discrete probability distributions as well as nondeterminism. Well-known submodels of STA are probabilistic timed automata (PTA) [13], timed automata (TA) [1], and generalised semi-Markov processes (GSMP) [10]. 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, 17] to architectural dependability models [4] and industrial production scheduling [14]. The principle idea behind the formalism and its supporting tools is to provide a *single-formalism, multi-solution* approach to modelling and analysis, using existing analysis engines and algorithms where available to avoid unnecessary reimplementations.

Started in 2008, the MODEST TOOLSET constitutes the second generation [6] of tools with this philosophy and currently consists of (i) mcpta [11, 12], which enables model checking of networks of PTA using the PRISM [16] probabilistic

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

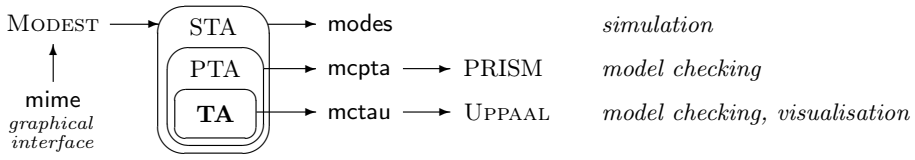


Fig. 1. How the new mctau tool fits in the MODEST TOOLSET

model checker in the background, (ii) modes [4, 11], 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] (see Figure 1 for a toolset overview). 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.

2 Bridging the Gap

A connection between MODEST and UPPAAL had been planned for a long time [6], but several fundamental differences 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* [7]). 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 not every deadline can be transformed into a single invariant (and vice-versa), we have recently shown how to transform all practically relevant deadlines into invariants [11], 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 label that are part of their shared alphabet, while UPPAAL 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 that allows multiple receivers (`a?`) for one sender (`a!`).

Although it is possible, with some effort, to encode binary using only multi-way synchronisation [15], we know of no way to do the opposite without introducing additional intermediate states. We thus resolved this discrepancy 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 [9] or PRISM.

3 The mctau Tool

mctau, at its core, performs the translation of Modest models to the XML-based input language of Uppaal, including Modest features such as user-defined functions and data types. 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 appears 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 to generate useable Uppaal models.

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 (discarding 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 also replaces probabilistic properties by a set of purely timed ones that can determine whether the probability is zero, one, or somewhere within that range.

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 from within the graphical interface of mime.

Tool availability. The Modest Toolset 1.3.4, which includes mctau, and Uppaal 4.1.5 are freely available for academic users at `www.modestchecker.net` (reviewers: see Appendix C) and `www.uppaal.org`, respectively.

4 Evaluation

mctau is able to analyse (the nondeterministic overapproximations of) the three original mcpta PTA case studies [12], models unchanged. In all cases where mctau

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

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 [12]). This shows how mctau can be of great help in debugging and sanity checking of probabilistic models.

The BRP model has also been studied as a pure TA model before [8] 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 model is included in the MODEST TOOLSET download. We also compared the performance of mctau and mcpta (using the digital clocks engine[3]) on a non-probabilistic version of the original MODEST BRP model. Table 2 summarises the results[4]; 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, multi-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 gaps between the two formalisms to be overcome. The implemented bridge spans a practically disturbing gap between CCS and

[3] 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.

[4] 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.

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.

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: SFM-RT 2004. pp. 200–236. No. 3185 in LNCS, Springer (September 2004)
4. Bogdoll, J., Ferrer Fioriti, L.M., Hartmanns, A., Hermanns, H.: Partial order methods for statistical model checking and simulation. In: FMOODS/FORTE. LNCS, vol. 6722, pp. 59–74. Springer (2011)
5. 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)
6. Bohnenkamp, H.C., Hermanns, H., Katoen, J.P.: MoTor: The Modest tool environment. In: TACAS. LNCS, vol. 4424, pp. 500–504. Springer (2007)
7. Bornot, S., Sifakis, J.: An algebraic framework for urgency. Inf. Comput. 163(1), 172–202 (2000)
8. D'Argenio, P.R., Katoen, J.P., Ruys, T.C., Tretmans, J.: The bounded retransmission protocol must be on time! In: TACAS. LNCS, vol. 1217. Springer (1997)
9. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: CADP 2010: A toolbox for the construction and analysis of distributed processes. In: TACAS. LNCS, vol. 6605, pp. 372–387. Springer (2011)
10. Haas, P.J., Shedler, G.S.: Regenerative generalized semi-Markov processes. Communications in Statistics. Stochastic Models 3(3), 409–438 (1987)
11. Hartmanns, A.: Model-checking and simulation for stochastic timed systems. In: FMCO. LNCS, vol. 6957, pp. 372–391. Springer (December 2010)
12. Hartmanns, A., Hermanns, H.: A Modest approach to checking probabilistic timed automata. In: QEST. pp. 187–196. IEEE Computer Society (2009)
13. Kwiatkowska, M.Z., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theor. Comput. Sci. 282(1), 101–150 (2002)
14. 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)
15. Norman, G., Palamidessi, C., Parker, D., Wu, P.: Model checking the probabilistic pi-calculus. In: QEST. pp. 169–178. IEEE Computer Society (2007)
16. Parker, D.: Implementation of Symbolic Model Checking for Probabilistic Systems. Ph.D. thesis, University of Birmingham (2002)
17. Yue, H., Bohnenkamp, H.C., Kampschulte, M., Katoen, J.P.: Analysing and improving energy efficiency of distributed slotted Aloha. In: NEW2AN. LNCS, vol. 6869, pp. 197–208. Springer (2011)

Appendix

This appendix contains our plan for a potential oral presentation of the tool paper, a screenshot of the mime GUI with mctau in use, information about tool availability including an anonymous download link, and a selection of additional material for the interested reviewer.

Please note that the appendix contains a separate bibliography; reference numbers are thus distinct from and overlapping with those from the main paper.

A Presentation Plan

Our planned presentation consists of two main parts:

1. *Introduction/Bridging the gap:* A brief introduction to timed automata, probabilistic timed automata and MODEST followed by an overview of the MODEST/UPPAAL discrepancies and how we resolve them.
2. *Example/demonstration:* A live demonstration of analysing the model of the bounded retransmission protocol with mctau, including model debugging with export to UPPAAL and a comparison with the results obtained from mcpta.

A.1 Introduction/Bridging the Gap

We plan to start our presentation with a general and very brief introduction to (probabilistic) timed systems and the MODEST language. The former already includes the timed automata (TA) formalism, and we thus do not need to explicitly present the UPPAAL language since it consists for the most part of the graphical representation of these automata. We then highlight the discrepancies between the UPPAAL input formalism and MODEST as outlined in Section 2 and how mctau overcomes these. This can be succinctly accomplished based on a table similar to Table 3.

A.2 Example/Demonstration

The second part of our presentation is based on the probabilistic timed model of the bounded retransmission protocol (BRP) that was already mentioned in Section 4. We do not intend to show the full code of the MODEST BRP model during our presentation. It is included in the MODEST TOOLSET download for review, though. Instead, we will try to weave crucial aspects of the model into the beginning of the presentation already (this possibility is one reason to use the BRP case, see below for details) and merely highlight very specific parts relevant to the properties under study in this second part of our presentation.

After a slide-based introduction of the protocol and the properties that will be model-checked, including an intuition of the results we should expect for a working protocol, we switch to mime (see Appendix B for a screenshot taken after analysing a BRP model in mime) to continue in a "tool demonstration"

Table 3. Tabular summary of the language discrepancies and their resolution

	UPPAAL	MODEST	Resolution
Time constraints:	invariants	deadlines	transformation [4]
Assignments:	atomic	sequential	extended UPPAAL
Synchronisation:	multi-way	binary + broadcast	extended both

style. The mime user interface is fully scalable so that we can magnify the entire interface to make it readable for the audience throughout the demonstration.

Our demonstration starts by pointing out a few crucial aspects of the BRP model. The model that we load into mime will contain incorrect values for some of the timing-related constants. After pointing out the meaning of these constants, we use mctau for model checking, yielding results that run contrary to the intuition that we explained before and clearly show that the protocol does not work properly. We then proceed to use mctau's export mode to obtain input for UPPAAL, which we load into the UPPAAL GUI to find counterexamples for the affected properties that show what is wrong with the model.

After correcting the model, model checking with mctau will give us results that are in line with our expectations. We finally end the demonstration by showing that these results are in line with those obtained from mcpta (as in Table 1); if time allows, we can actually run mcpta to compute the true probabilities, showing that this takes noticeably longer than using mctau on the same model.

Going back to slides, the presentation itself ends with a summary of the contributions of mctau and our extensions to UPPAAL and MODEST to (probabilistic) timed model checking in general, in particular concerning the aspect of further tool connections.

A.3 Reasons for Demonstrating the BRP

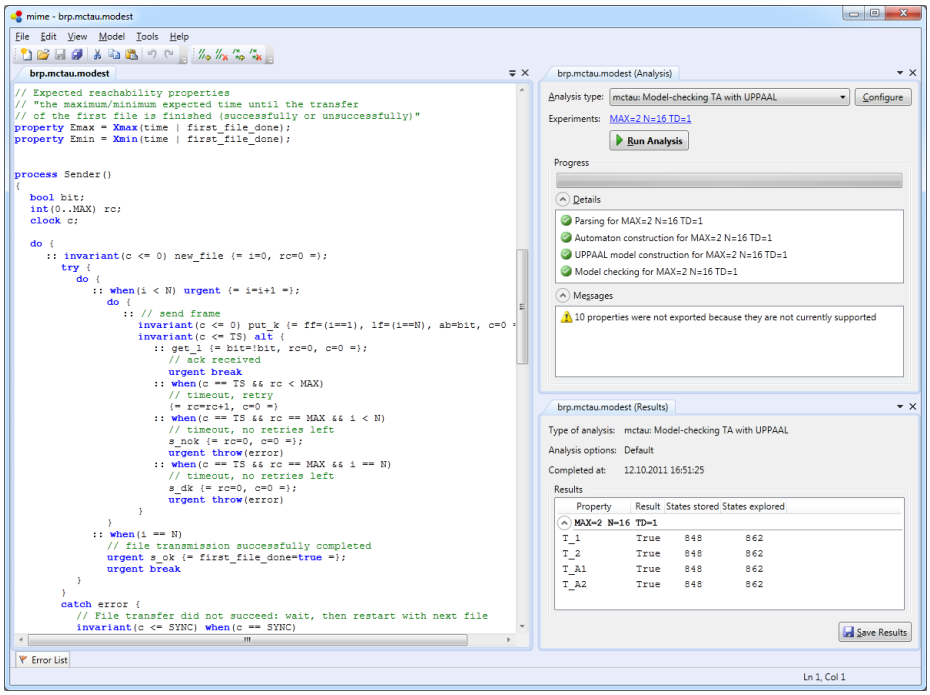
We chose the BRP case study for the demonstration part for several reasons:

- The protocol concept is easy to explain, seeing that it is the classic alternating-bit protocol, a staple of computer networks lectures, with the addition of a bound on the number of retransmissions.
- It has been studied and analysed with a wide range of different approaches over the last two decades, making it relevant and also more probable that the audience is familiar with it.
- The model has three parameters: N, the number of "chunks" that are to be transferred; MAX, the bound on the number of retransmissions; and TD, the maximum transmission delay. These parameters make it possible to achieve almost any desired size for the underlying state space, ranging from small instances like $(N, MAX, TD) = (16, 2, 1)$ well-suited for a demonstration to larger ones that can be used to emphasise the difference in runtime between timed and probabilistic timed model checking.

– The constraints on the timing parameters obtained in [2] are a good case for the necessity of timed model checking. In particular, they hold for the nondeterministic overapproximation of the PTA model, highlighting the usefulness of TA model checking even for PTA models.
– The MODEST code for this model is easily modified to use either deadlines or invariants and binary or multi-way synchronisation, allowing us to present the bridging work that we have done in action.
– As mentioned, parts of this example, in particular code and automata for a lossy channel with (nondeterministic) transmission delay, can already be introduced at the very beginning of the presentation to illustrate MODEST and TA.

B Screenshot

The following screenshot shows the mime graphical interface. The BRP model from [5] had been loaded and the purely timed properties T_1, T_2, T_{A1} and T_{A2} have been analysed with mctau. Using mcpta or modes instead of mctau for this model merely requires selecting that engine in the drop-down box on the upper right and running another analysis. The results could then be displayed alongside those obtained from mctau, or exported to disk. More screenshots as well as examples and documentation are available on the website (see next page).



C Tool Availability

The regular download of the MODEST TOOLSET on `www.modestchecker.net` requires a name and e-mail address. To allow an anonymous review, version 1.3.4 of the MODEST TOOLSET, including the models mentioned in this paper, is available for reviewers without login at

`http://www.modestchecker.net/v134spin/`

Both the MODEST TOOLSET and UPPAAL are cross-platform; on non-Windows systems, the MODEST TOOLSET requires a very recent version of the Mono runtime[5] ($\geq$ 2.**10**.1) and mime is currently not available.

D Additional Material

This section contains more detailed explanations for some of the aspects of mctau and our extensions to MODEST and UPPAAL that cannot be part of the paper proper due to the space restrictions. They are included here for the reviewers' convenience.

D.1 Time constraints: Deadlines

The usefulness as well as the particularities of deadlines as used by MODEST are explained in detail in [4]. As an example for the relationship between deadlines and invariants, consider a location l with invariant $c \leq 3$ (where c is a clock variable). The invariant implies that 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.

The invariants that we cannot transform into deadlines are mainly equality comparison deadlines like $c = 3$ and equivalents. Their semantics (as deadlines) is not as obvious as it may seem: If a location affected by deadline $c = 3$ is entered when $c \leq 3$, time can pass until c == 3, then the location must be left. However, if such a location is entered when $c > 3$, the deadline has no effect on the progress of time. A suitable invariant would thus have to prevent the crossing of $c = 3$ from below without restricting points of time beyond that. This is not possible with a single invariant. A solution that adds additional states would be possible (split all incoming edges depending on whether $c \leq 3$ or $c > 3$), but introduces an open clock constraint (a problem for mcpta/digital clocks) and additional locations, which we want to avoid in order not to make the state-space explosion problem even worse. We have not seen any practical use for equality comparison deadlines so far.

Another transformation from timed automata with deadlines to UPPAAL timed automata is presented in [3]. The set of deadlines allowed is restricted compared to the ones that we support in MODEST. The transformation also

[5] `http://www.mono-project.org/`

makes use of features specific to UPPAAL such as committed and urgent locations, which our transformation [4] does not rely on.

D.2 Synchronisation

As an example for the different synchronisation modes, consider three processes P_1, P_2 and P_3 in parallel composition:

CSP/MODEST-style: If $\alpha(P_1) = \{a, b\}$, $\alpha(P_2) = \{b, c\}$ and $\alpha(P_2) = \varnothing$, then P_1 (P_2) is free to take action a (c), but P_1 and P_2 must synchronise to take action b.

CCS-style binary: If P_1 can perform `a!`, P_2 can perform `a?` and P_3 can perform `a?` and action `a` is restricted (i.e. forced to synchronise), then either P_1 and P_2 or P_1 and P_3 will synchronise on `a`.

Broadcast: In the same setting as for binary synchronisation, all three processes will synchronise. If $\alpha(P_3)$ contained `a`, but `a?` were not enabled, P_1 and P_2 could still synchronise on `a` without P_3.

The addition of binary and broadcast synchronisation to MODEST is not needed to export MODEST models to UPPAAL. However, it allows the MODEST modeller to use these synchronisation modes native to UPPAAL. Binary and broadcast communication are very natural in some settings, and make it possible to arrive at more concise models in such cases, so we are happy to have them included and supported by the exporting mechanism now.

One reason why we have chosen the practical way of just implementing the missing modes to resolve the synchronisation mode discrepancy is that we know of no way to transform CSP-style multi-way synchronisation involving more than one automaton into binary synchronisation without adding additional states. This is because every binary synchronisation step cannot involve more than two automata. Adding states (locations) in the automata will reinforce the state-space explosion problem. Any encoding we can think of will split edges and hence increase the size at least linear in the size of the original component automata. When interleaving them, the increase becomes multiplicative, at least. But even so, it is as yet unclear (to us) whether locations can be added in a semantically sound way, in light of unwanted interleavings between automata running in parallel, even if features such as UPPAAL's committed locations are used.

Finally, our list of tools to connect to in the future is very incomplete. It would still be incomplete if we also added LTSA [6] as an additional CSP-style tool and the Edinburgh CWB [7] and CWB-NC [1] as additional examples for CCS-style tools.

D.3 Graph layout

The automatic graph layout performed by `mctau` is based on the Graph# library[6], which we adapted for timed automata will all their location and edge

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

```
action put, get;
process Channel() {
   clock c;
   put {= c = 0 =};
   invariant(c <= 2) alt {
      :: get
      :: tau
   }; Channel() }
```

⇒

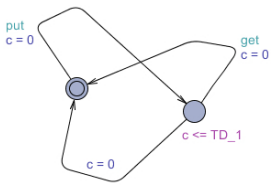


Fig. 2. A MODEST process and its UPPAAL automaton

labels. Figure 2 shows a simple communication channel in MODEST and the UPPAAL automaton generated by mctau based on the *LinLog* layout algorithm.

D.4 Overapproximation of Probabilistic Choices

To illustrate the process of determining whether a probabilistic property in a PTA is surely 0 or 1, consider property $P_{\max}(\Diamond\, e)$. This property determines the maximum probability (over all schedulers) of eventually reaching a state satisfying expression e. After replacing probabilistic by nondeterministic choices, this property 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 $[0, 1]$.

References

1. Cleaveland, R., Sims, S.: The NCSU Concurrency Workbench. In: CAV. LNCS, vol. 1102, pp. 394–397. Springer (1996)
2. D'Argenio, P.R., Katoen, J.P., Ruys, T.C., Tretmans, J.: The bounded retransmission protocol must be on time! In: TACAS. LNCS, vol. 1217. Springer (1997)
3. Gómez, R.: A compositional translation of timed automata with deadlines to uppaal timed automata. In: FORMATS. LNCS, vol. 5813, pp. 179–194. Springer (2009)
4. Hartmanns, A.: Model-checking and simulation for stochastic timed systems. In: FMCO. LNCS, vol. 6957, pp. 372–391. Springer (December 2010)
5. Hartmanns, A., Hermanns, H.: A Modest approach to checking probabilistic timed automata. In: QEST. pp. 187–196. IEEE Computer Society (2009)
6. Magee, J., Kramer, J.: Concurrency – state models and Java programs (2nd ed.). Wiley (2006)
7. Stevens, P.: A practical introduction to games, infinity and the Edinburgh Concurrency Workbench. In: FIW. pp. 35–36. IOS Press (2005)