TAPAAL 2.0: Integrated Development Environment for Timed-Arc Petri Nets*

Alexandre David, Lasse Jacobsen, Morten Jacobsen, Kenneth Yrke Jørgensen, Mikael H. Møller, and Jiří Srba

Department of Computer Science, Aalborg University,
Selma Lagerlöfs Vej 300, 9220 Aalborg Øst, Denmark

Abstract. TAPAAL 2.0 is a platform-independent modelling, simulation and verification tool for extended timed-arc Petri nets. The tool supports component-based modelling and offers an automated verification of the EF, AG, EG and AF fragments of TCTL via translations to Uppaal timed automata and via its own dedicated verification engine. After more than three years of active development with a main focus on usability aspects and on the efficiency of the verification algorithms, we present the new version of TAPAAL 2.0 that has by now reached its maturity and offers the first publicly available tool supporting the analysis and verification of timed-arc Petri nets.

1 Introduction

Timed-arc Petri nets (TAPN) are a particular time extension of the classical Petri net model where the time information is attached to the tokens in the net, representing their ages, and arcs from places to transition contain time intervals that restrict the ages of tokens which are moved along the arcs. The model of TAPN was first studied by Bolognesi, Lucidi, Trigila and Hanisch [4, 9] and it has proved particularly suitable for modelling of manufacturing systems, workflow management systems and other applications [1, 2, 14–16].

We present TAPAAL 2.0, an open source and platform-independent tool (available from `www.tapaal.net`) that allows users to edit, simulate and verify extended TAPN models in a component-based fashion through shared interfaces. There exist several other verification tools for timed models, like Uppaal [19] for networks of timed automata, and Romeo [8] and Tina [3] for time Petri nets (where time intervals are associated with transitions, as opposed to tokens like in the TAPN model). However, all these models are rather different (and complementary) and even though translations among them are possible [18], their suitability from the modeller's point of view depends on the application domain. For example, time Petri nets use an a priory fixed number of clocks (one for each transition) while TAPN allow for dynamic creation of tokens that carry their own local clocks. Some criticism on the classical TAPNs mentions the lack of modelling features for ensuring urgent behaviour and lack of read-arcs.

* The paper was supported by VKR Center of Excellence MT-LAB.

In the extended TAPN model used in TAPAAL, the weak points are dealt with by enforcing urgency via age invariants, modelling read-arcs in a more general setting via transport arcs and introducing other features, like inhibitor arcs and components, facilitating a more convenient modelling.

We are not aware of other tools for the analysis of TAPN, except for two tool prototypes implementing a backward coverability algorithm based on the better-quasi-ordering technique [2] and a forward reachability algorithm presented in [1]. Both implementations consider only the basic TAPN model. The former tool allows to verify solely coverability queries (remarkably also for unbounded nets) while the latter one may not terminate as the reachability questions are in general undecidable. Neither of the tools supports a GUI interface and they do not seem to be maintained.

TAPAAL 1.1 was presented in [6] as a tool providing a TAPN editor, simulator and translator to Uppaal timed automata. The two translations implemented in TAPAAL 1.1 preserve only safety properties but have showed the potential of the translation approach. In the present version of TAPAAL 2.0, the tool now has a completely new and dedicated verification engine (implemented in `C++`) and two novel translations preserving liveness. In addition, the GUI and the modelling features were significantly extended; notably we now support component-based model development, constants, inhibitor arcs, advanced query creation dialog, batch processing engine and numerous other features. The theory behind the tool has been published in [17, 5, 11, 12].

2 Tool Description

The architecture of the tool is outlined in Figure 1. The GUI of the tool has originally been developed as an open source project PIPE 2.5 [10] but since its TAPAAL branch in 2008, it has been significantly extended with new features like a component-based editor that allows to describe component interfaces via shared places and transitions. The composed net can be simulated in a timed simulator, displaying traces with concrete delays, or verified either via TAPAAL's own engine or via automatic translations to the Uppaal engine. Verification can be initiated from a user-friendly query dialog or as a batch job.

Verification Options. TAPAAL 2.0 adds two new translations from extended TAPNs to networks of timed automata that use the broadcast communication feature of Uppaal [11] and contrary to the previous two translations that rely on handshake synchronization (already present in TAPAAL 1.1), they do not produce additional deadlocks and hence allow us to verify also liveness properties. The two new translations can additionally handle all the new extended features, including inhibitor arcs. The main extension of TAPAAL 2.0 is its own dedicated engine implementing an efficient forward reachability algorithm on abstract markings (ages of tokens represented via zones) while applying on-the-fly active clock reduction (resizing of zones to contain only clocks of the active tokens) and other optimizations. The monotonicity property (more tokens added

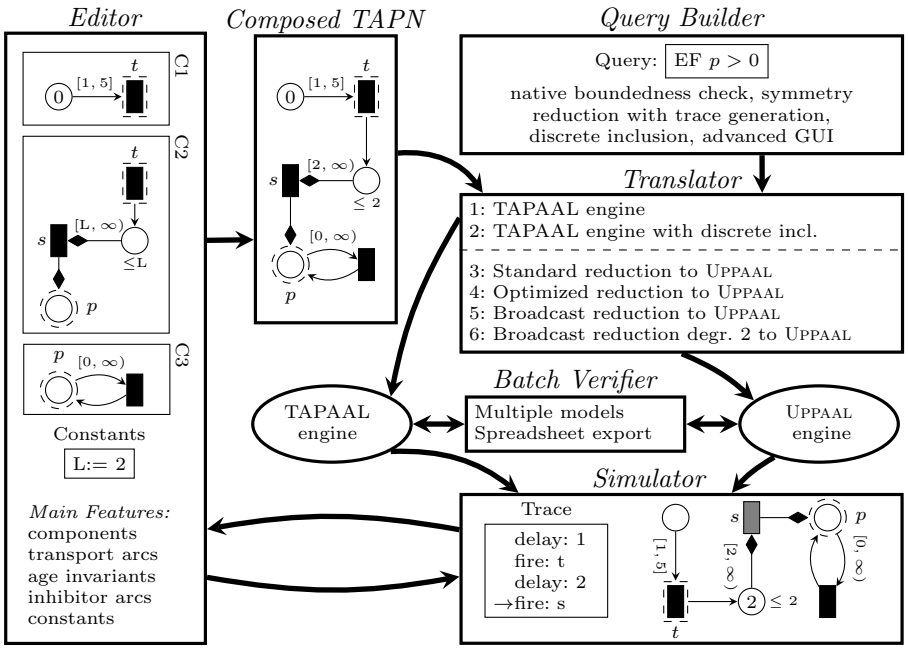

Fig. 1. Architecture of TAPAAL 2.0

to the net cause only more behaviour) that holds for the basic TAPNs, but breaks for the extended TAPNs due to the features like age invariants and inhibitor arcs, can be used to speed-up the verification. From the static analysis of the net, we define a novel ordering on markings of the net so that monotonicity is preserved even for the extended TAPN model and we exploit this in the reachability algorithm, still providing exact verification answers but often with considerable speed-up as demonstrated in Section 3. This technique, called *discrete inclusion*, can be further optimized by a manual intervention of the modeller. The verification engine also implements an automatic symmetry reduction technique and returns executable traces even if symmetry is activated (unlike e.g. Uppaal or TAPAAL 1.1). Finally, the new engine implements a k-boundedness check of a given net. Even if the net is unbounded, the verification up to k tokens in the net is possible, providing a suitable under-approximation of the net behaviour.

Management of Tool Development. To facilitate easy collaboration between the TAPAAL tool contributors, we utilize `launchpad.net/tapaal`, a free tool-chain for collaboration in open source projects. Among others, all software bugs found in TAPAAL are registered and tracked using launchpad's bug management system. To this day, more than 20 individuals have contributed to the development of TAPAAL, working on more than 200 registered bugs and features, during over ten official releases of TAPAAL.

	Uppaal engine				TAPAAL translations				TAPAAL engine			
	original		improved		original		improved		original		improved	
#	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes
3	0.1	<0.1	<0.1	<0.1	0.4	0.2	<0.1	<0.1	0.2	<0.1	<0.1	<0.1
4	0.4	<0.1	<0.1	<0.1	16.8	0.3	<0.1	0.1	2.8	<0.1	<0.1	<0.1
5	5.3	0.1	<0.1	<0.1	—	0.6	<0.1	0.1	89.1	0.2	<0.1	<0.1
6	220.5	0.2	<0.1	<0.1	—	1.8	<0.1	0.1	—	0.9	<0.1	<0.1
7	—	1.1	0.1	<0.1	—	14.5	0.1	0.1	—	6.3	<0.1	<0.1
8	—	3.6	0.5	<0.1	—	104.8	0.1	0.1	—	48.9	<0.1	<0.1
9	—	20.7	3.1	<0.1	—	—	0.1	0.1	—	—	<0.1	<0.1
10	—	143.6	23.2	<0.1	—	—	0.1	0.1	—	—	<0.1	<0.1
11	—	—	148.0	<0.1	—	—	0.1	0.1	—	—	<0.1	<0.1
40	—	—	—	0.9	—	—	0.6	0.7	—	—	4.1	0.6
80	—	—	—	22.8	—	—	11.1	12.7	—	—	158.9	11.0
120	—	—	—	159.8	—	—	73.9	84.8	—	—	—	68.3
160	—	—	—	—	—	—	293.8	—	—	—	—	262.3

Fig. 2. Scheduling Feasibility of MPEG-2 Encoder (time in seconds)

3 Experiments

We present two new case studies in order to argue for the efficiency and applicability of the tool. More experimental results can be found e.g. in [5, 12] and several TAPN models are available within the tool (under File/Example nets). All the models used in the following experiments can be obtained from the tool homepage (section Download). The experiments were carried out on a MacBook Pro equipped with a 2.7GHz Intel Core i7 and 8GB of RAM with a 300 seconds time limit. We used the 64-bit versions of TAPAAL 2.0.2 and Uppaal 4.1.4.

MPEG-2 Case Study. We model the MPEG-2 algorithm that encodes a group of frames on a multiprocessor architecture. The algorithm treats one initial I-frame, a number of B-frames (we parameterize our model on this number), and a final P-frame. The TAPN model was taken directly from [14]. We recreated the Uppaal model from the descriptions in [7] since their original Uppaal model was not available anymore. The results are in Figure 2. The columns called *original* list the verification times for the model described in [14]; in the *improved* variant we employed several additional modelling optimizations (both in the timed automata and the TAPN model) via the use of invariants and symmetry reduction (features not available to the authors of [14]). The query asks whether the encoding can be performed within a given time bound (that we vary). In positive cases (*yes* columns) we used DFS, otherwise (*no* columns) we used BFS. This allows us to see how good the tools are to find a trace to a reachable state or to explore the whole state-space. The discrete inclusion technique does not improve the performance of the TAPAAL engine in this particular case.

Lynch-Shavit Protocol. The second case study is a timed-based mutual exclusion algorithm by Lynch and Shavit [13]. Both the TAPN and timed automata models

#	Uppaal engine	TAPAAL translations	TAPAAL engine	TAPAAL inclusion	TAPAAL M-incl.	TAPAAL M*-incl.
15	0.2	0.4	0.9	0.2	0.2	0.2
25	2.4	3.0	8.5	1.1	0.9	0.9
35	14.6	15.8	42.7	4.6	4.00	3.9
45	62.6	57.1	153.3	14.3	11.3	10.5
55	190.8	164.7	—	38.8	27.2	25.0
65	—	—	—	106.5	57.7	52.7
75	—	—	—	262.4	113.2	100.5
85	—	—	—	—	203.2	178.6
95	—	—	—	—	—	299.2

Fig. 3. Lynch-Shavit Protocol for Mutual Exclusion (time in seconds)

were taken from [1]. The column called *inclusion* refers to the generic application of the discrete inclusion technique, *M-incl.* refers to a manual optimization of the technique and *M*-incl.* shows a possible best performance of the technique (optimized by a brute force search with an automatic script). The results are presented in Figure 3 for a different number of processes participating in the protocol. The discrete inclusion technique is not an over-approximation of the behaviour and provides conclusive answers applied to any model, not only the protocol in this case study.

TAPAAL's performance is convincing in comparison with state-of-the-art model checkers like Uppaal and, in several cases, it provides a considerably faster verification. Due to space limitation, we present only two case studies but we also observed similar performance improvements for other models. In particular, if the net structure allows for more tokens in the same place, the generic discrete inclusion technique often gives a significant speed up and it can be further manually tuned up. Moreover, if symmetry reduction is applicable, it is often better exploited in the net models (where its detection is automatic on contrary to the user defined one in Uppaal models) and our translations create networks of timed automata that are significantly faster to verify than the native Uppaal models. Last but not least, TAPAAL allows for simulation of concrete error traces (even in case of symmetry reduction) while many other timed automata and Petri net tools display only the abstract ones, which makes them difficult to understand for the end-users.

4 Conclusion

TAPAAL 2.0 is an open source, platform-independent modelling and verification tool for extended timed-arc Petri nets. The tool has reached its maturity both in the GUI aspects as well as in the actual verification performance. The tool is becoming increasingly popular as documented by the total number of 2089 downloads (calculated in October 2011), out of which more than 650 downloads took part since April 2011.

References

1. P.A. Abdulla, J. Deneux, P. Mahata, and A. Nylén. Using forward reachability analysis for verification of timed Petri nets. *Nordic J. of Computing*, 14:1–42, 2007.
2. P.A. Abdulla and A. Nylén. Timed Petri nets and BQOs. In *Proc. of ICATPN'01*, volume 2075 of *LNCS*, pages 53–70. Springer, 2001.
3. B. Berthomieu, P-O. Ribet, and F. Vernadat. The tool TINA — construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 42(14):2741–2756, 2004.
4. T. Bolognesi, F. Lucidi, and S. Trigila. From timed Petri nets to timed LOTOS. In *Proc. of the IFIP WG 6.1 10th International Symposium on Protocol Specification, Testing and Verification*, pages 1–14. North-Holland, Amsterdam, 1990.
5. J. Byg, K.Y. Jørgensen, and J. Srba. An efficient translation of timed-arc Petri nets to networks of timed automata. In *Proc. of ICFEM'09*, volume 5885 of *LNCS*, pages 698–716. Springer, 2009.
6. J. Byg, K.Y. Jørgensen, and J. Srba. TAPAAL: Editor, simulator and verifier of timed-arc Petri nets. In *Proc. of ATVA'09*, volume 5799 of *LNCS*, pages 84–89. Springer, 2009.
7. M.E. Cambronero, A.P. Ravn, and V. Valero. Using UPPAAL to analyze an mpeg-2 algorithm. In *Proc. of VII Workshop Brasileiro de Tempo Real*, pages 73–82, 2005.
8. G. Gardey, D. Lime, M. Magnin, and O.H. Roux. Romeo: A tool for analyzing time Petri nets. In *Proc. of CAV'05*, volume 3576 of *LNCS*, pages 418–423. Springer, 2005.
9. H.M. Hanisch. Analysis of place/transition nets with timed-arcs and its application to batch process control. In *Proc. of ICATPN'93*, volume 691 of *LNCS*, pages 282–299. Springer, 1993.
10. Platform Independent Petri net Editor 2.5. `http://pipe2.sourceforge.net`.
11. L. Jacobsen, M. Jacobsen, M.H. Møller, and J. Srba. A framework for relating timed transition systems and preserving TCTL model checking. In *Proc. of EPEW'10*, volume 6342 of *LNCS*, pages 83–98. Springer, 2010.
12. L. Jacobsen, M. Jacobsen, M.H. Møller, and J. Srba. Verification of timed-arc Petri nets. In *Proc. of SOFSEM'11*, volume 6543 of *LNCS*, pages 46–72. Springer, 2011.
13. N. Lynch and N. Shavit. Timing-based mutual exclusion. In *Proceedings of the 13th IEEE Real-Time Systems Symposium*, pages 2–11, 1992.
14. F.L. Pelayo, F. Cuartero, V. Valero, H. Macia, and M.L. Pelayo. Applying timed-arc Petri nets to improve the performance of the MPEG-2 encoding algorithm. In *Proc. of MMM'04*, pages 49–56. IEEE, 2004.
15. V.V. Ruiz, J.J. Pardo, and F. Cuartero. Translating TPAL specifications into timed-arc Petri nets. In *Proc. of ICATPN'02*, volume 2360 of *LNCS*, pages 414–433. Springer, 2002.
16. V.V. Ruiz, F.L. Pelayo, F. Cuartero, and D. Cazorla. Specification and analysis of the MPEG-2 video encoder with timed-arc Petri nets. *ENTCS*, 66(2), 2002.
17. J. Srba. Timed-arc Petri nets vs. networks of timed automata. In *Proc. of ICATPN'05)*, volume 3536 of *LNCS*, pages 385–402. Springer, 2005.
18. J. Srba. Comparing the expressiveness of timed automata and timed extensions of Petri nets. In *Proc. of FORMATS'08*, volume 5215 of *LNCS*, pages 15–32. Springer, 2008.
19. UPPAAL. `http://uppaal.org`.

Appendix: TAPAAL 2.0 Mini Tutorial

The screenshot in Figure 4 shows TAPAAL 2.0 during the simulation of a simple model of a sender and receiver over a lossy medium. TAPAAL models are extensions of the well-known Petri Net models, where tokens now carry a real value (timestamp, or age) and input arcs are annotated with time intervals. In our example, we have three tokens of ages 3.0, 3.0 and 5.0, respectively. In order to fire a transition (drawn as a rectangle), we need at least one token in every input place (drawn as a circle) connected by an arc to the transition, and the age of the token must moreover belong to the interval on the arc.

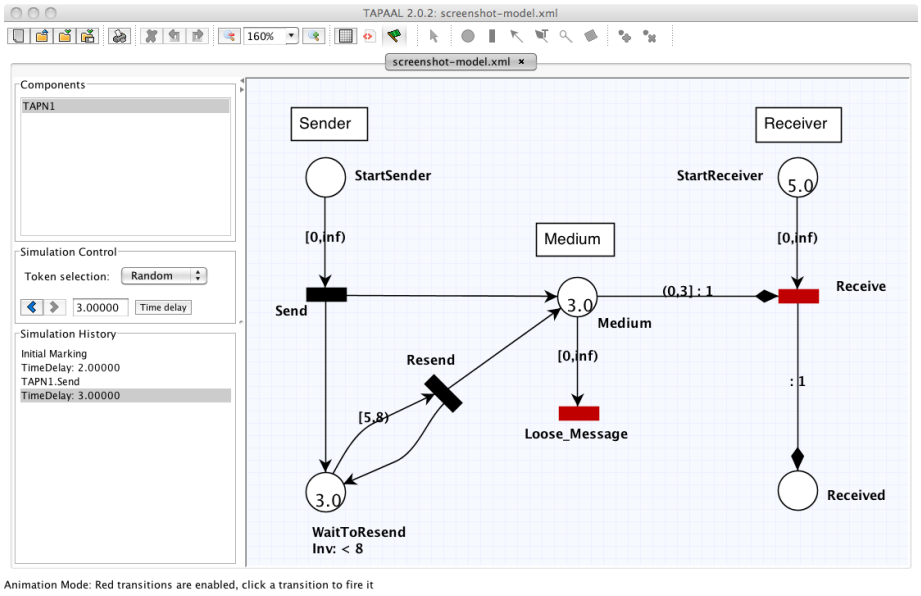

Fig. 4. A screenshot of TAPAAL 2.0.2 during simulation

This means, referring again to Figure 4, that the transitions `Receive` and `Loose_Message` can fire in the current marking; the choice of which one will fire is done in a nondeterministic manner. When a transition is fired, all input arcs consume one token of a suitable age from each of the input places and the output arcs produce new tokens to the output places. The net in our example contains two types of arcs; arcs with arrow tips are called normal arcs and arcs with diamond tips are transport arcs. Tokens produced by normal arcs into the output places become new (fresh) tokens with initial ages 0, while tokens produced by transport arcs preserve the age of the token that is "transported" along the corresponding pair of transport arcs. Hence, for example, firing the transition `Receive` will consume the tokens in places `Medium` and `StartReceiver`, and create one token in the place `Received` of age 3. TAPAAL additionally supports

inhibitors arcs (not shown in the screenshot), which are special input arcs that disable transitions from firing whenever there is at least one token in the input place.

Another type of behaviour of the net is a time delay. Here all tokens grow older with the same speed, provided that none of the tokens breaks any age invariants associated to the places, like the one < 8 in the place `WaitToResend`. This means that the net, in its current marking, cannot delay for 5 time units or more (otherwise the token's age 8 in the `WaitToResend` place breaks the invariant) and the transition `Resend` becomes urgent. Hence invariants can be used to enforce urgency in the net behaviour.

Finally, when we switch to the editor view (by pressing the small green flag that switches between the editor and the simulator), we can create a new query asking whether it is possible to place at least one token into the place `Received` by writing the logical formula `EF TAPN1.Received >= 1`. The EF temporal quantifier asks whether there is some reachable marking that satisfies that the place `Received` in the component `TAPN1` (we have only one component in our model) contains at least one token.

The user is encouraged to consult our wikipage `wiki.tapaal.net` for a more thorough introduction into the tool and when first running the tool, it is advisable to open the intro-example from File/Example nets in order to learn the basics about TAPAAL 2.0.