TAPAAL: Editor, Simulator and Verifier of Timed-Arc Petri Nets

Joakim Byg, Kenneth Yrke Jørgensen, and Jiří Srba*

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

Abstract. TAPAAL is a new platform independent tool for modelling, simulation and verification of timed-arc Petri nets. TAPAAL provides a stand-alone editor and simulator, while the verification module translates timed-arc Petri net models into networks of timed automata and uses the UPPAAL engine for the automatic analysis.
We report on the status of the first release of TAPAAL (available at `www.tapaal.net`), on its new modelling features and we demonstrate the efficiency and modelling capabilities of the tool on a few examples.

1 Introduction

Petri net is a popular mathematical model of discrete distributed systems introduced in 1962 by Carl Adam Petri in his PhD thesis. Since then numerous extensions of the basic place/transition model were studied and supported by a number of academic as well as industrial tools [8]. Many recent works consider various extensions of the Petri net model with time features that can be associated to places, transitions, arcs or tokens. A recent overview aiming at a comparison of the different time dependent models (including timed automata) is given in [15].

In the TAPAAL tool we consider *Timed-Arc Petri Nets* (TAPN) [4, 7] where an age (a real number) is associated with each token in a net and time intervals are placed on arcs in order to restrict the ages of tokens that can be used for firing a transition. The advantages of this model are an intuitive semantics and the decidability of a number of problems like coverability and boundedness (for references see [15]). On the other hand, the impossibility to describe urgent behaviour limited its modelling power and wider applicability. TAPAAL extends the TAPN model with new features of *invariants* and *transport arcs* in order to model urgent behaviour and transportation of tokens without resetting their age.

TAPAAL has an intuitive modelling environment for editing and simulation of TAPN. It also provides a verification module with automatic checking of bounded TAPN models against safety and liveness requirements via a translation to networks of timed automata and then using the UPPAAL [16] engine as a back-end for verification.

* Author was partially supported by Ministry of Education of the Czech Republic, project No. MSM 0021622419.

The connection between bounded TAPN and timed automata was studied in [13, 14, 5] and while theoretically satisfactory, the translations described in these papers are not suitable for a tool implementation as they either cause an exponential blow-up in the size or create a new parallel component with a fresh local clock for *each place* in the net. As UPPAAL performance becomes significantly slower with the growing number of parallel processes and clocks, the verification of larger nets with little or no concurrent behaviour (few tokens in the net) becomes intractable.

In TAPAAL we suggest a novel translation technique where a fresh parallel component (with a local clock) is created only for *each token* in the net. The proposed translation also transforms safety and liveness properties (EF, AG, EG, AF) into equivalent UPPAAL queries. One of the main advantages of this translation approach is the possibility to use *active clock reduction* and *symmetry reduction* techniques recently implemented in UPPAAL. As a result the verifiable size of models increases by orders of magnitude.

To the best of our knowledge, TAPAAL is the first publicly available tool which offers modelling, simulation and verification of timed-arc Petri nets with continuous time. There is only one related tool prototype mentioned in [1] where the authors discuss a coverability algorithm for general (unbounded) nets, though without any urgent behaviour. Time features (time stamps) connected to tokens can be modelled also in Coloured Petri Nets using CPN Tools [10], however, time passing is represented here using a global clock rather than the local ones as in TAPN, only discrete time semantics is implemented in CPN Tools and the analysis can be nondeterministic as the time stamps are in some situations ignored during the state-space construction.

2 TAPAAL Framework

TAPAAL offers an editor, a simulator and a verifier for TAPN. It is written in Java 6.0 using Java Swing for the GUI components and it is so available for the majority of existing platforms.

TAPAAL's graphical *editor* features all necessary elements for a creation of TAPN models, including invariants on places and transport arcs. The user interface supports, among others, a select/move feature for moving a selected subnet of the model as well as an undo/redo buttons allowing the user to move backward and forward in the history during a creation of larger models. Constructed nets and queries are saved in an interchangeable XML format using the Petri Net Markup Language (PNML) [12] further extended with TAPAAL specific timing features. An important aspect of the graphical editor is that it disallows to enter syntactically incorrect nets and hence no syntax checks are necessary before calling further TAPAAL modules.

The *simulator* part of TAPAAL allows one to inspect the behaviour of a TAPN by graphically simulating the effects of time delays and transition firings. When firing a transition the user can either manually select the concrete tokens that will be used for the firing or simply allow the simulator to automatically select the tokens based on some predefined strategy: youngest, oldest or random. The simulator also allows the user to step back and forth in the simulated trace, which makes it easier to investigate alternative net behaviours.

TAPAAL's *verification* module enables us to check safety and liveness queries in the constructed net. Queries are created using a graphical query dialog, completely eliminating the possibility of introducing syntactical errors and offering an intuitive and easy to use query formulation mechanism. The TAPAAL query language is a subset of the CTL logic comprising EF, AG, EG and AF temporal operators[1], however, several TCTL properties can be verified by encoding them into the net. The actual verification is done via translating TAPN models into networks of timed automata and by using the model checker UPPAAL. The verification calls to UPPAAL are seamlessly integrated inside the TAPAAL environment and the returned error traces (if any) are displayed in TAPAAL's simulator. For safety questions concrete traces are displayed whenever the command-line UPPAAL engine can output them, otherwise the user is offered an untimed trace and can in the simulation mode experiment with suitable time delays in order to realize the displayed trace in the net. A number of verification/trace options found in UPPAAL are also available in TAPAAL, including a symmetry reduction option which often provides orders of magnitude improvement with respect to verification time, though at the expense of disallowing trace options (a current limitation of UPPAAL). Finally, it is possible to check whether the constructed net is k-bounded or not, for any given k. The tool provides a suitable under-approximation of the net behaviour in case the net is unbounded.

The TAPAAL code consists of two parts: the editor and simulator, extending the Platform Independent Petri net Editor project PIPE version 2.5 [9], which is licensed under the Open Software License 3.0, and a framework for translating TAPN models into UPPAAL, licensed under the BSD License.

3 Experiments

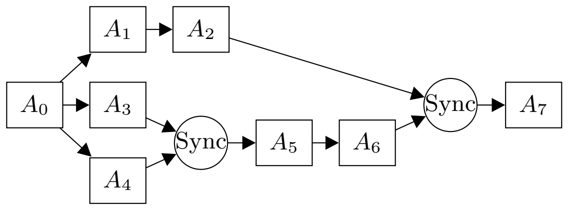
We shall now report on a few experiments investigating the modelling capabilities and verification efficiency of TAPAAL. All examples are included in the TAPAAL distribution and can be directly downloaded also from `www.tapaal.net`.

3.1 Workflow Processes with Deadlines

Workflow processes provide a classical case study suitable for modelling in different variants of Petri nets. A recent focus is, among others, on the addition of timing aspects into the automated analysis process. Gonzalez del Foyo and Silva consider in [6] workflow diagrams extended with task durations and the latest execution deadline of each task. They provide a translation into Time Petri Nets (TPN), where clocks are associated with each transition in the net, and use the tool TINA [3] to analyze schedulability questions. An example of a workflow process (taken from [6]) is illustrated in Fig. 1. The translation described in [6]

[1] At the moment the EG and AF queries are supported only for nets with transitions that do not contain more than two input and two output places.

Task	Duration	Deadline
A_0	5	5
A_1	4	9
A_2	4	15
A_3	2	9
A_4	2	8
A_5	3	13
A_6	3	18
A_7	2	25



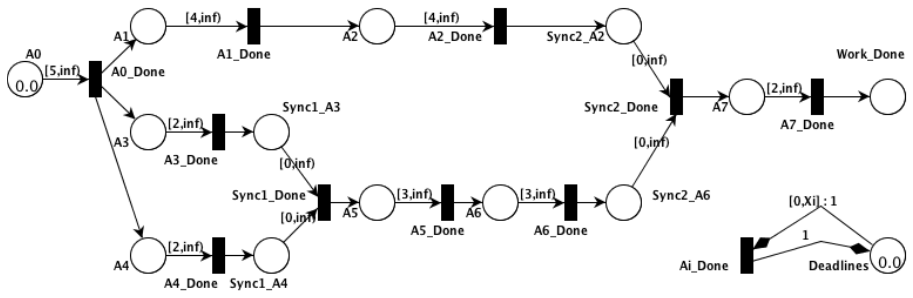


Fig. 1. A simple workflow diagram and its timed-arc Petri net model.

relies on preprocessing of the workflow so that the individual (relative) deadlines for each task must be computed before a TPN model can be constructed.

In our model of extended timed-arc Petri nets a more direct translation without any preprocessing can be given. See Fig. 1 for a TAPAAL Petri net model resulting from the translation of the workflow example. Every transition with the naming schema Ai_done corresponds to the finalisation of the execution of the task A_i. The duration constraints are encoded directly into the net and the global deadlines are handled by adding a fresh place called Deadlines, containing one token (initially of age 0). The latest execution deadline X_i of a task A_i (where $0 \leq i \leq 7$) is then ensured by adding (for each i) a pair of transport arcs constrained with the time interval $[0, X_i]$ between the place Deadlines and the corresponding transition Ai_done. A schematic illustration is given in Fig. 1. Notice that transport arcs have different arrow tips than the standard arcs and are annotated with the label 1 to denote which arcs are paired into a route for the age-preserving token transportation (the annotations are relevant only in the presence of multiple transport arcs connected to a single transition). As the age of the token in the place Deadlines is never reset, it is guaranteed that the latest execution deadlines of all tasks are met.

The workflow example was verified in TAPAAL against the query EF (Work_Done = 1) and by selecting the fastest trace option the tool returned in 0.1s the following scheduling of task executions together with the necessary time delays: 5, A0_Done, 2, A3_Done, A4_Done, Sync1_Done, 2, A1_Done, 1, A5_Done, 3, A2_Done, A6_Done, Sync2_Done, 2, A7_Done.

3.2 Fischer's Protocol and Alternating Bit Protocol

Fischer's protocol [11] for mutual exclusion and alternating bit protocol [2] for network communication via lossy communication medium are well-known and scalable examples used for a tool performance testing. In our experiments we managed to verify Fischer's protocol (with a TAPN model taken from [1]) for 200 processes (each with its own clock) within 29 minutes, while an equivalent timed automaton model of the protocol provided in the UPPAAL distribution verified 200 processes in 2 hours and 22 minutes. The experiment showed a speed-up of 205% for 100 processes, 293% for 150 processes and 393% for 200 processes. The explanation to this seemingly surprising phenomenon is that the translated timed automata model of the protocol contains on one hand more discrete states than the native UPPAAL model (about twice as many), but on the other hand the zones that are tested for inclusion are smaller. As a result, the verification times for the TAPAAL produced automata are significantly faster.

Correctness of the alternating bit protocol was verified for up to 50 messages (each message has its own time-stamp) currently present in the protocol in less than two hours, while a native UPPAAL model verification took more than a day without any result. The speed-up was even more significant than in the case of Fischer's protocol: for 15 messages UPPAAL used 136 seconds and TAPAAL 7.3 seconds, for 17 messages UPPAAL needed 32 minutes and TAPAAL 13.7 seconds. There was also a difference in the tool performance depending on what kind of verification options were used, demonstrating a similar pattern of behaviour as in the case of Fischer's protocol (models with more discrete states and smaller zones verify faster).

4 Conclusion

TAPAAL offers a graphical environment for editing, simulation and verification of timed-arc Petri nets and the introduction of novel elements like invariants on places and transport arcs provides useful features particularly suitable for modelling of workflow processes, time sensitive communication protocols and other systems. The tool shows a promising performance in verification of safety and liveness properties.

The future development will focus on incorporating C-like functions and data structures into tokens in the net, on extending the firing policies with urgency and priorities, and on generalizing the query language. The aim is also to provide concrete error traces for liveness properties, rather than only the abstract or untimed ones (a current limitation of UPPAAL). We also plan to extend the model with cost, probability and game semantics and then use the corresponding UPPAAL branches (CORA, PROB and TIGA) for verification.

Acknowledgments. We would like to thank the UPPAAL team at Aalborg University and in particular Alexandre David for numerous discussions on the topic.

References

[1] P.A. Abdulla and A. Nylén. Timed Petri nets and BQOs. In *Proceedings of the 22nd International Conference on Application and Theory of Petri Nets (ICATPN'01)*, volume 2075 of *LNCS*, pages 53–70. Springer-Verlag, 2001.

[2] K.A. Bartlett, R.A. Scantlebury, and P.T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Commun. ACM*, 12(5):260–261, 1969.

[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 *Proceedings of the IFIP WG 6.1 Tenth International Symposium on Protocol Specification, Testing and Verification (Ottawa 1990)*, pages 1–14, 1990.

[5] P. Bouyer, S. Haddad, and P.-A. Reynier. Timed Petri nets and timed automata: On the discriminating power of Zeno sequences. *Information and Computation*, 206(1):73–107, 2008.

[6] Pedro M. Gonzalez del Foyo and José Reinaldo Silva. Using time Petri nets for modelling and verification of timed constrained workflow systems. In *ABCM Symposium Series in Mechatronics*, volume 3 of *ABCM*, pages 471–478. ABCM - Brazilian Society of Mechanical Sciences and Engineering, 2008.

[7] H.M. Hanisch. Analysis of place/transition nets with timed-arcs and its application to batch process control. In *Proceedings of the 14th International Conference on Application and Theory of Petri Nets (ICATPN'93)*, volume 691 of *LNCS*, pages 282–299, 1993.

[8] F. Heitmann, D. Moldt, K.H. Mortensen, and H. Rölke. Petri nets tools database quick overview. `http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html`.

[9] Platform Independent Petri net Editor 2.5. `http://pipe2.sourceforge.net`.

[10] Kurt Jensen, Lars Kristensen, and Lisa Wells. Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 9(3):213–254, 2007.

[11] L. Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems*, 5(1):1–11, 1987.

[12] Petri Net Markup Language. `http://www2.informatik.hu-berlin.de/top/pnml`.

[13] J. Sifakis and S. Yovine. Compositional specification of timed systems. In *Proceedings of the 13th Annual Symposim on Theoretical Aspects of Computer Science (STACS'96)*, volume 1046 of *LNCS*, pages 347–359. Springer-Verlag, 1996.

[14] J. Srba. Timed-arc Petri nets vs. networks of timed automata. In *Proceedings of the 26th International Conference on Application and Theory of Petri Nets (ICATPN 2005)*, volume 3536 of *LNCS*, pages 385–402. Springer-Verlag, 2005.

[15] J. Srba. Comparing the expressiveness of timed automata and timed extensions of Petri nets. In *6th Int. Conf. on Formal Modelling and Analysis of Timed Systems (FORMATS'08)*, volume 5215 of *LNCS*, pages 15–32. Springer, 2008.

[16] UPPAAL. `www.uppaal.com`.