Time Petri Nets Analysis with TINA

Bernard Berthomieu and François Vernadat
LAAS / CNRS
7 avenue du Colonel Roche, 31077 Toulouse, France
Email: {Bernard.Berthomieu|Francois.Vernadat}@laas.fr
Telephone: +33/(0) 5 61 33 63 00
Fax: +33/(0) 5 61 33 64 11

***Abstract*— Beside the usual graphic editing and simulation facilities, the software tool `Tina` may build a number of state space abstractions for Petri nets or Time Petri nets, preserving certain classes of properties. For Petri nets, these abstractions help preventing combinatorial explosion and rely on so-called partial order techniques such as covering steps and/or persistent sets. For Time Petri nets, that have in general infinite state spaces, they provide finite representation of their behavior, in terms of state class graphs.**

I. Introduction

`Tina` (TIme Petri Net Analyzer, http://www.laas.fr/tina) is a software environment to edit and analyze Petri Nets and Time Petri Nets. This paper overviews its capabilities, architecture, and main applications. More details can be found in [1].

Time Petri nets [2] are one of the most widely used model for the specification and verification of real-time systems. They extend Petri nets with temporal intervals associated with transitions, specifying firing delay ranges for the transitions.

In addition to the usual editing and analysis facilities of similar environments, `Tina` offers various abstract state space constructions that preserve specific classes of properties of the state spaces of nets, like absence of deadlocks, linear time temporal properties, or bisimilarity. For untimed systems, abstract state spaces helps to prevent combinatorial explosion. For timed systems, abstractions are mandatory as their state spaces are typically infinite, `Tina` implements various abstractions based on state classes.

Tina accepts input in graphical or textual formats, including $PNML$ (an XML based exchange format for Petri nets). Transition system outputs can be produced in a number of textual or binary formats, for external checkers.

II. Constructions

a) Classical methods: A first group of tools provided by `Tina` implement "classical" constructions and methods for Petri nets: reachability graphs, coverability graphs (determining unbounded places) and structural analysis (invariants).

b) Partial order abstractions: A second group of tools implement so-called "partial-order" reduction techniques, aimed at preventing combinatorial explosion due to representation of parallelism by interleaving. Two families of methods are provided. The first, based on "stubborn", or "persistent", sets, consists, under certain conditions, of exploring only one path among all equivalent possible paths w.r.t. the class of properties to be preserved. The second implements the "covering steps" technique, in which one fires "steps", or sets of independent transitions fired simultaneously, rather than simple transitions. These techniques preserve deadlock freeness, and, under certain conditions, the linear structure of state spaces. Combining these two techniques yields the method of persistent steps [3].

c) State class graphs: The last group of constructions concerns Time Petri nets. The state spaces of Time Petri nets are in general infinite, state space abstractions for TPN's preserving various classes of properties can be computed in terms of so-called state classes [4] [5] [6]. State classes group possibly infinite sets of states, represented by a marking and a polyhedron capturing temporal information.

Different state class constructions are available, preserving different families of properties of the state space. The well known *State class graph* construction of [4] [5] preserves markings of the TPN and all its properties one can express in linear time temporal logics like LTL. Two more recent constructions, explained in [6], are also supported: the *Strong state classes graph* preserving the states (a state associates a marking with time information for the enabled transitions) and LTL properties, and the *Atomic state class graph*, preserving states and bisimilarity with the state graph.

Realtime properties, like those expressed in logic $TCTL$ are checked using the standard technique of observers. The technique is applicable to a large class of realtime properties and can be used to analyze most of the "timeliness" requirements found in practice. An alternative is provided by path analysis, for which `Tina` provides a dedicated tool, `plan`, able to computes all firing schedules over some firing sequence.

III. Model-Checking

`Tina` can present its results in a variety of formats, understood by model checkers like `MEC` [7], a μ-calculus formula checker, or behavior equivalence checkers like `Bcg`, part of the $CADP$ toolset [8]. In addition, several model-checkers are being developed specifically for `Tina`. The first available, `selt`, is a model-checker for an enriched version of $State/Event - LTL$ [9], a linear time temporal logic supporting both state and transition properties. The logic is rich enough to encode marking invariants like $G(p1 + p3 >= 3)$. For the properties found false, a timed counter example is computed and can be replayed by the simulator.

IV. ARCHITECTURE

The functional architecture of `Tina` is shown Figure 1. The kernel of `Tina` is the exploration engine, parameterized by the class of properties to be preserved.

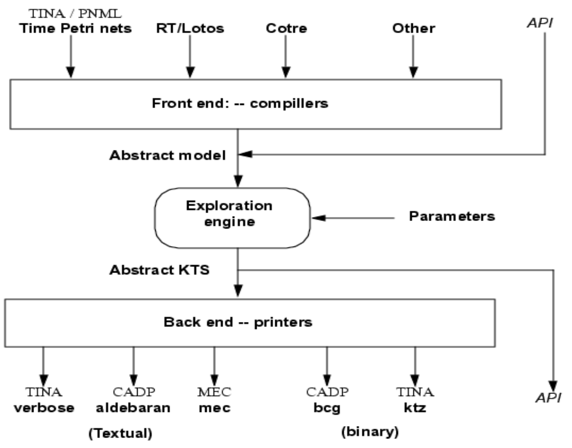
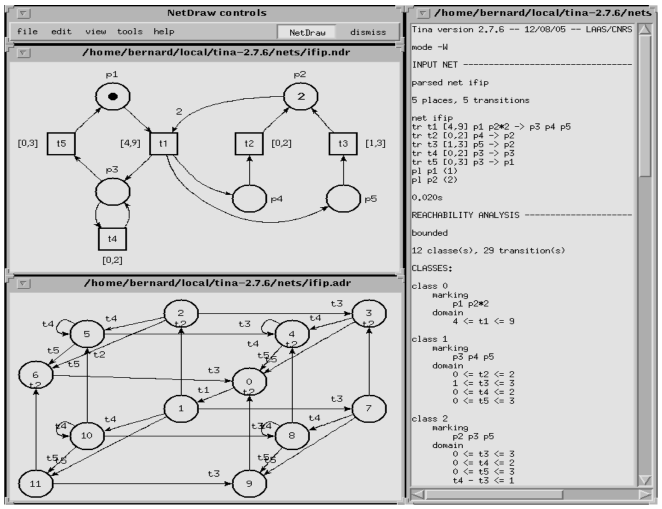


Fig. 1. TINA Architecture

The front-ends convert models into internal representation (abstract timed transition systems). An abstract API is available, so that users wishing to use `Tina` for analyzing their specific models can develop their own front-end. That API abstractly implements a class of "Time Predicate/Action nets", adding to TPN's the capabilities of manipulating data.

Similarly, several back-ends convert the Kripke transition systems obtained as the result of the various constructions into physical representations readable by the proprietary or external model checkers and transition system analyzers.

A screen snapshot of a typical `Tina` session is shown in Figure 2, showing a TPN being edited and its state class graph in verbose and graphical representations.


NetDraw controls
file
edit
view
tools
help
NetDraw
dismiss
/home/bernard/local/tina-2.7.6/nets/ifip.ndr
/home/bernard/local/tina-2.7.6/nets/ifip.adr
/home/bernard/local/tina-2.7.6/nets

```
Tina version 2.7.6 -- 12/08/05 -- LAAS/CNRS

mode -W

INPUT NET ----------------------------------

parsed net ifip

5 places, 5 transitions

net ifip
tr t1 [4,9] p1 p2*2 -> p3 p4 p5
tr t2 [0,2] p4 -> p2
tr t3 [1,3] p5 -> p2
tr t4 [0,2] p3 -> p3
tr t5 [0,3] p3 -> p1
pl p1 (1)
pl p2 (2)

0.020s

REACHABILITY ANALYSIS ---------------------

bounded

12 classe(s), 29 transition(s)

CLASSES:

class 0
    marking
        p1 p2*2
    domain
        4 <= t1 <= 9

class 1
    marking
        p3 p4 p5
    domain
        0 <= t2 <= 2
        1 <= t3 <= 3
        0 <= t4 <= 2
        0 <= t5 <= 3

class 2
    marking
        p2 p3 p5
    domain
        0 <= t3 <= 3
        0 <= t4 <= 2
        0 <= t5 <= 3
        t4 - t3 <= 1
```

Fig. 2. TINA Screen snapshot

V. PROJECTS

The Tina toolbox is currently used in several industrial projects such as TOPCASED (Toolkit in OPen-source for Critical Application & SystEms Development, `http://www.topcased.org`) and OpenEmbeDD. TOPCASED proposes to build a toolkit for the development of real time embedded systems, from specification to implementation, including formal verification, in an integrated "Model Driven Engineering" approach. OpenEmbeDD is a national RNTL project also promoting an MDE approach, addressing the different aspects of the design process of real time embedded software. SPICES - an ITEA project - focuses on AADL (Architecture and Analysis Description Language) descriptions.

VI. PROSPECTIVE

A number of developments are ongoing for `Tina`, concerning new tools, new front-ends, and new back-ends. A μ-calculus model checker is in progress, as well as a version of `selt` operating "on-the-fly". New front-ends are scheduled, notably for $RT - Lotos$ and a realtime language developed within project TOPCASED.

Addition to Time Petri nets of suspension/resumption of actions, of a great value for modeling scheduled real-time systems, has been investigated by several authors, notably [10], [11]. A major extension of `Tina` is being experimented that will support such features, based on the "Stopwatch Time Petri Nets" described in [12].

REFERENCES

[1] B. Berthomieu, P.-O. Ribet, and F. Vernadat, "The tool TINA – construction of abstract state spaces for Petri nets and time Petri nets," *Int. J. of Production Research*, vol. 42, no. 14, pp. 2741–2756, 15 July 2004.

[2] P. M. Merlin and D. J. Farber, "Recoverability of communication protocols: Implications of a theoretical study." *IEEE Tr. Comm.*, vol. 24, no. 9, pp. 1036–1043, Sept. 1976.

[3] P.-O. Ribet, F. Vernadat, and B. Berthomieu, "On combining the persistent sets method with the covering steps graph method," in *Proc. of FORTE 2002, Springer LNCS 2529*, 2002, pp. 344–359.

[4] B. Berthomieu and M. Menasche, "An enumerative approach for analyzing time Petri nets." *IFIP Congress Series*, vol. 9, pp. 41–46, 1983.

[5] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets." *IEEE Trans. on Software Engineering*, vol. 17, no. 3, pp. 259–273, 1991.

[6] B. Berthomieu and F. Vernadat, "State class constructions for branching analysis of time Petri nets," in *Proc. Tools and Algorithms for the Construction and Analysis of Systems, Springer LNCS 2619*, 2003.

[7] A. Arnold, *Systmes de transitions finis et smantique des processus communicants.* Masson, Paris, 1974.

[8] J.-C. Fernandez, H. Garavel, R. Mateescu, L. Mounier, and M. Sighireanu, "Cadp, a protocol validation and verification toolbox," in *8th Conf. Computer-Aided Verification, Springer LNCS 1102*, 1996, pp. 437–440.

[9] S. Chaki, M. E, Clarke, J. Ouaknine, N. Sharygina, and N. Sinha, "State/event-based software model checking," in *4th Int. Conf. on Integrated Formal Methods, Springer LNCS 2999*, 2004, pp. 128–147.

[10] G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario, "Time state space analysis of real-time preemptive systems," *IEEE Trans. on Software Engineering*, vol. 30, no. 2, pp. 97–111, February 2004.

[11] D. Lime and O. H. Roux, "Expressiveness and analysis of scheduling extended time Petri nets," in *5th IFAC Int. Conf. on Fieldbus Systems and their Applications.* Elsevier Science, July 2003.

[12] B. Berthomieu, D. Lime, O. Roux, and F. Vernadat, "Reachability problems and abstract state spaces for time Petri nets with stopwatches." *Journal of Discrete Event Dynamic Systems*, 2007 (to appear).