Romeo: A Parametric Model-Checker for Petri Nets with Stopwatches

Didier LIME[1], Olivier (H.) ROUX[1], Charlotte SEIDNER[1], and Louis-Marie TRAONOUEZ[1]

[1] IRCCyN, CNRS UMR 6597, Nantes, France
{`Didier.Lime` | `Olivier-h.Roux` | `Charlotte.Seidner` | `Louis-Marie.Traonouez`}`@irccyn.ec-nantes.fr`

Abstract. Last time we reported on Romeo, analyses with this tool were mostly based on translations to other tools. This new version provides an integrated TCTL model-checker and has gained in expressivity with the addition of parameters. Although there exists other tools to compute the state-space of stopwatch models, Romeo is the first one that performs TCTL model-checking on stopwatch models. Moreover, it is the first tool that performs TCTL model-checking on timed parametric models. Indeed, Romeo now features an efficient model-checking of time Petri nets using the Uppaal DBM Library, the model-checking of stopwatch Petri nets and parametric stopwatch Petri nets using the Parma Polyhedra Library and a graphical editor and simulator of these models. Furthermore, its audience has increased leading to several industrial contracts. This paper reports on these recent developments of Romeo.

Key words: Time Petri nets, model-checking, stopwatches, parameters, TCTL, tool

1 Introduction

Time Petri nets (*TPNs*) [1] are a classical time extension of Petri nets. They allow an easy representation of real-time systems features such as synchronization and parallelism. State reachability is decidable for bounded *TPNs*, which is sufficient for virtually all practical purposes.

However, it is also often useful to model actions that can be suspended and later resumed. Several extensions of time Petri nets have been proposed to express the preemptive scheduling of tasks, such as *Scheduling-TPNs* [2], inhibitor hyperarc *TPNs* (*ITPNs*) [3] or *Preemptive-TPNs* [4]. All these models belong to the class of *TPNs* extended with stopwatches (*SwPNs*)[5]. Reachability and most other properties of interest are however undecidable for *SwPNs*, even when bounded [5].

Furthermore, the design of a system often benefits from the use of parameters, e.g. when specifications are not yet completely defined. Parametric *TPNs* (*PTPNs*) and parametric *SwPNs* (*PSwPNs*) are parametric extensions of *TPNs* and *SwPNs* that can be used to perform parametric model-checking [6]. The goal is to synthesize constraints on the parameters which helps the system design.

2 Presentation of Romeo

The Romeo tool [1] (available for Linux, MacOSX and Windows platforms) consists of a graphical user interface (GUI) (written in Tcl/Tk), to edit and simulate *TPNs*, and a computation module Mercutio (written in C++), that performs model-checking and state-space computation.

The two other main tools for the analysis of *TPNs* and *SwPNs* are Tina [7] and Oris [8]. Tina has many interesting features, including the computation of a graph preserving CTL (Computation Tree Logic) properties and an off-line full LTL model-checker. Oris also has several unique features of interest among which the most notable is probably the analysis of time Petri nets with stochastic aspects.

Design and simulation The GUI allows Petri nets edition and features three different modes for each supported extension (*TPNs*, *Scheduling-TPNs* or *ITPNs*). In each mode, a parametric extension is available (*PTPNs*, *Scheduling-PTPNs* or *PITPNs*). In these latter extensions, Romeo supports the use of parametric linear expressions in the time bounds of the transitions, and allows to add linear constraints on the parameters to restrict their domain.

The GUI also features an on-line simulator for scenario testing. It proposes to choose between two state-space exploration methods, the zone-based method for *TPNs* or the state-class method for *TPNs* and the other extensions.

On-the-fly model-checking Through the computation module Mercutio, Romeo can perform model-checking of time Petri nets models for quantitative temporal logic formulae (TCTL). We consider a restricted subset of TCTL formulae with no recursion in the formulae. This subset allows efficient on-the-fly model-checking and is sufficient to verify many interesting properties on time models. Reachability properties can be checked with formulae such as $\exists\Diamond_{[a,b]}(p)$ (where $[a,b]$ is a time interval, with b possibly infinite, and p a property on the markings of the net) and safety properties with $\forall\Box_{[a,b]}(p)$. Liveness properties can be checked with $\forall\Diamond_{[a,b]}(p)$ or by using a bounded response property such as $p \leadsto_{[0,b]} q$. It is equivalent to $\forall\Box(p \Rightarrow \forall\Diamond_{[0,b]}(q))$, and thus allows one level of recursion.

This subset allows to implement efficient model-checking algorithms for *TPNs* with the state-class graph [9]. In bounded *TPNs*, the algorithm is based on DBMs (Difference Bounds Matrix) and the implementation uses the Uppaal DBM Library [10] to encode the firing domains. For *SwPNs* and parametric *SwPNs*, the reachability problem is undecidable and as a consequence semi-algorithms are implemented in Romeo. In these models, firing domains are encoded with polyhedra by using the Parma Polyhedra Library [11].

With parametric nets, Romeo can verify parametric TCTL formulae in which the bounds of temporal constraints (a and b in the above examples) can be

[1] Download at: `http://romeo.rts-software.org/`

replaced by parameters. As a result, constraints on the parameters are synthesized in a disjunction of polyhedra which represents the values of the parameters for which the formula is verified.

State-space computation Romeo implements two state-space computation methods, the state-class graph and the zone-based graph.

Romeo computes the state-class graph (SCG) that preserves LTL properties [12]. The algorithm is based on DBMs for bounded *TPNs* and the semi-algorithm is based on polyhedra for *PTPNs*. For *SwPNs*, exact computation semi-algorithms are implemented by using either polyhedra or both polyhedra and DBMs. Finally an overapproximating semi-algorithm is also available for *SwPNs* and uses DBMs.

For *TPNs*, Romeo generates a zone-based graph [13] that preserves markings and converges by inclusion.

Finally, translations from *TPNs* to Timed Automata are available through the state-class automaton or by a structural translation [14]. *SwPNs* can be translated into Stopwatch Automata by the overapproximation method introduced in [15].

3 New functionalities

Since the last paper about Romeo [16], several enhancements have been made to the tool.

Regarding expressiveness, we have added the support for several classical special arcs: read arcs (resp. logical inhibitor arcs) test the number of tokens in a place, without consuming them, for the relation "greater than" (resp. "less than"). Reset arcs empty a place of all its token (regardless of their number) after the firing of a transition.

Furthermore, in addition to the scheduling extension, stopwatches can now be defined using time inhibitor arcs [3].

Regarding verification, we now have a model-checker for TCTL properties for *TPNs*, *SwPNs* and their parametric counterparts. It allows to work with Romeo without the help of other tools.

This is, to our knowledge, the first TCTL model-checker for (bounded) time Petri nets but also the first one for stopwatch models.

Furthermore, as already mentioned, Romeo now implements a new framework for the design and verification of parametric time Petri nets as described in [6]. The associated parametric model-checker is also, to our knowledge, the first one to perform TCTL parametric model-checking on timed parametric models.

4 Conclusion

Romeo is one of the three main tools on time and stopwatch Petri nets. It performs state-space computation and simulation based on both the state-class method and the zone-based method. Moreover, it performs on-the-fly TCTL model-checking of time, stopwatch and parametric Petri nets. Its implementation is very efficient thanks to the use of two robust libraries, namely the Uppaal DBM Library and the Parma Polyhedra Library.

ROMEO has many industrial users such as *DGA*, *SODIUS*, *Dassault Aviation* and *EADS* leading to several industrial contracts and partnerships.

References

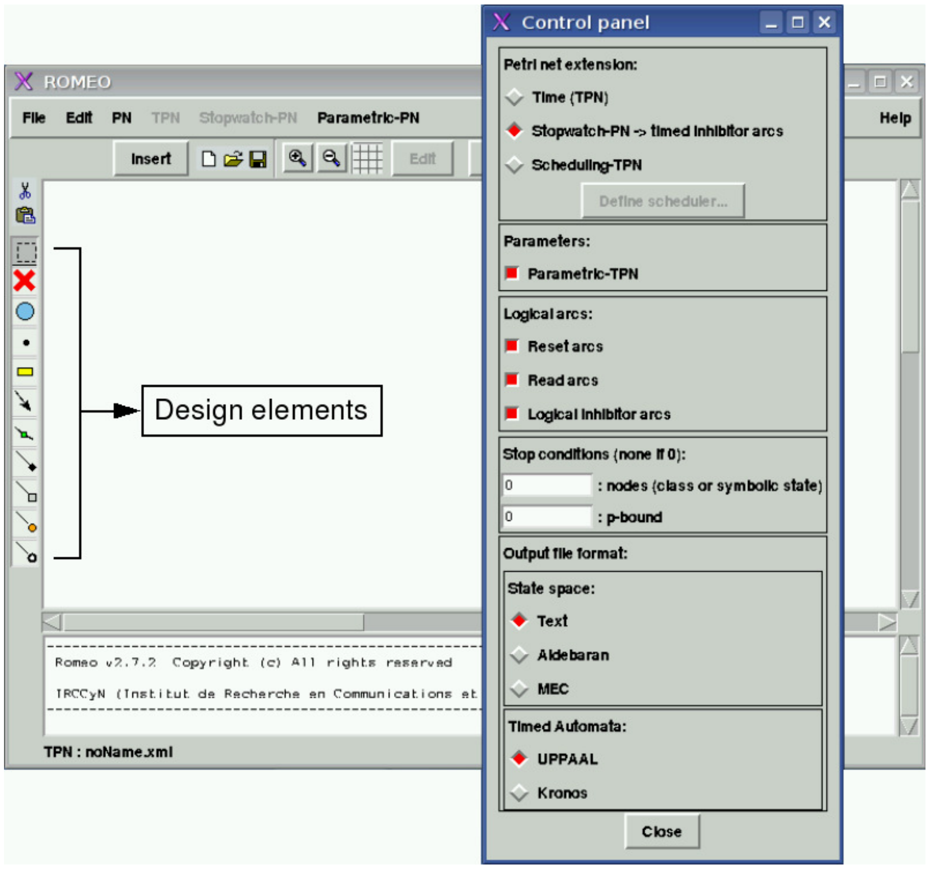
1. Merlin, P.: A study of the recoverability of computing systems. PhD thesis, Department of Information and Computer Science, Univ. of California, Irvine (1974)
2. Roux, O., Déplanche, A.M.: A t-time Petri net extension for real time-task scheduling modeling. European Journal of Automation (JESA) **36**(7) (2002) 973–987
3. Roux, O.H., Lime, D.: Time Petri nets with inhibitor hyperarcs. Formal semantics and state space computation. In: ICATPN'04. Volume 3099 of LNCS., Bologna, Italy, Springer (June 2004) 371–390
4. Bucci, G., Fedeli, A., Sassoli, L., Vicario, E.: Time state space analysis of real-time preemptive systems. IEEE trans. on Soft. Eng. **30**(2) (February 2004) 97–111
5. Berthomieu, B., Lime, D., Roux, O.H., Vernadat, F.: Reachability problems and abstract state spaces for time Petri nets with stopwatches. Discrete Event Dynamic Systems **17**(2) (2007) 133–158
6. Traonouez, L.M., Lime, D., Roux, O.H.: Parametric model-checking of time Petri nets with stopwatches using the state-class graph. In: FORMATS'08. Volume 5215 of LNCS., Saint-Malo, France, Springer (September 2008) 280–294
7. Berthomieu, B., Ribet, P.O., Vernadat, F.: The tool tina – construction of abstract state spaces for Petri nets and time Petri nets. Int. Journal of Production Research **42**(4) (July 2004) Tool available at http://www.laas.fr/tina/.
8. G.Bucci, L.Sassoli, E.Vicario: Oris: A tool for state-space analysis of real-time preemptive systems. In: QEST'04. (September 2004)
9. Hadjidj, R., Boucheneb, H.: On-the-fly TCTL model checking for time Petri nets using state class graphs. In: ACSD, IEEE Computer Society (2006) 111–122
10. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. International Journal on Software Tools for Technology Transfer **1**(1–2) (Oct 1997) 134–152
11. Bagnara, R., Hill, P.M., Zaffanella, E.: The Parma Polyhedra Library. Quaderno 457, Dipartimento di Matematica, Università di Parma, Italy (2006)
12. Berthomieu, B., Diaz, M.: Modeling and verification of time dependent systems using time Petri nets. IEEE trans. on Soft. Eng. **17**(3) (1991) 259–273
13. Gardey, G., Roux, O.H., Roux, O.F.: State space computation and analysis of time Petri nets. Theory and Practice of Logic Programming (TPLP) **6**(3) (2006) 301–320 Copyright Cambridge Press.
14. Cassez, F., Roux, O.H.: Structural translation from time Petri nets to timed automata. In: AVoCS'04. ENTCS, London (UK), Elsevier (September 2004)
15. Lime, D., Roux, O.H.: A translation based method for the timed analysis of scheduling extended time Petri nets. In: The 25th IEEE RTSS'04, Lisbon, Portugal (December 2004)
16. Gardey, G., Lime, D., Magnin, M., Roux, O.H.: Roméo: A tool for analyzing time Petri nets. In: CAV'05. Volume 3576 of LNCS., Edinburgh, Scotland, UK, Springer (July 2005)

Appendix

Presentation of Romeo in an application

We illustrate the features of Romeo in a scheduling problem taken from [4]. We consider a system of three tasks: $task_1$ and $task_3$ are periodic, $task_2$ is sporadic. The periods are expressed in function of a time parameter a and are respectively a, $2.a$ and $3.a$ for the tasks 1, 2 and 3. The system has fixed priorities between the tasks: $task_1$ has the greatest priority, then $task_2$ and then $task_3$.

We design a *PTPN* with time inhibitor arcs model of this system in Romeo. The graphical user interface of the tool is presented in Figure 1. We choose in the control panel the type of net we want to edit, and we add the elements of the net (places, transitions, arcs).



Petri net extension:
Time (TPN)
[✓ Stopwatch-PN -> timed inhibitor arcs]
Scheduling-TPN

Parameters:
[✓ Parametric-TPN]

Logical arcs:
[✓ Reset arcs]
[✓ Read arcs]
[✓ Logical inhibitor arcs]

Stop conditions (none if 0):
0 : nodes (class or symbolic state)
0 : p-bound

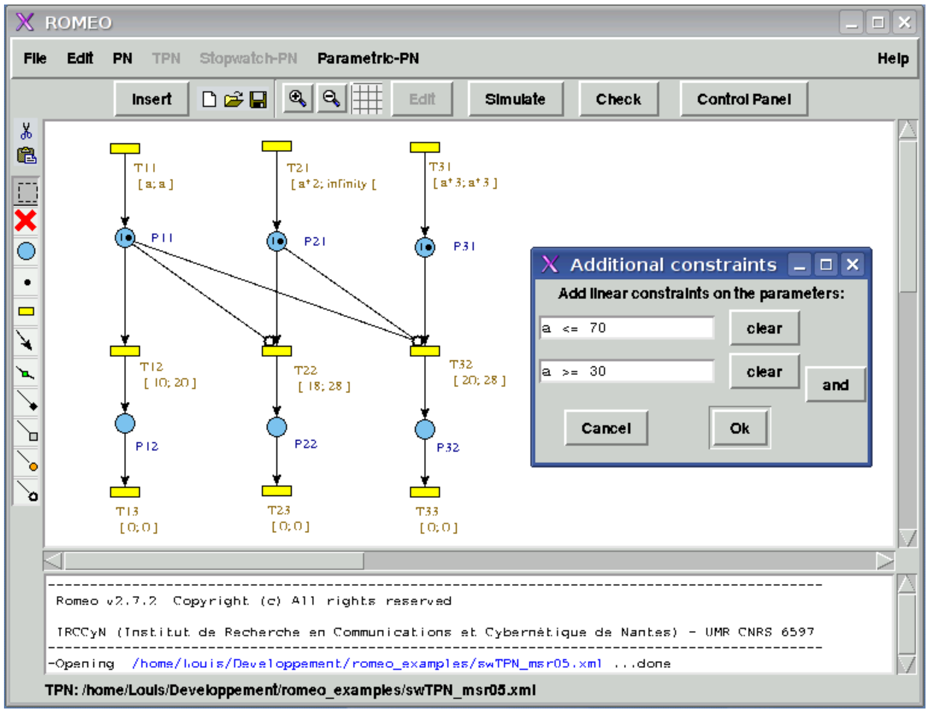
Output file format:
State space:
[✓ Text]
Aldebaran
MEC
Timed Automata:
[✓ UPPAAL]
Kronos

```
Romeo v2.7.2  Copyright (c) All rights reserved
IRCCyN (Institut de Recherche en Communications et
```

TPN : noName.xml

Fig. 1. GUI of Romeo with the control panel

We obtain the model presented in Figure 2, in which the inhibitors arcs are drawn with a circle end. We add constraints on the parameter to restrict its domain so that $30 \leq a \leq 70$.



Add linear constraints on the parameters:

a <= 70

a >= 30

```
------------------------------------------------------------------------------------------
 Romeo v2.7.2  Copyright (c) All rights reserved

 IRCCyN (Institut de Recherche en Communications et Cybernétique de Nantes) - UMR CNRS 6597
------------------------------------------------------------------------------------------
-Opening  /home/louis/Developpement/romeo_examples/swTPN_msr05.xml ...done
```

TPN: /home/Louis/Developpement/romeo_examples/swTPN_msr05.xml

Fig. 2. *PITPN* model of the system with constraints on the parameters

The simulator of Romeo presented in figure 3 allows to test scenarios for an early verification of some properties. Firable transitions are in green, whereas the one only enabled are in brown.

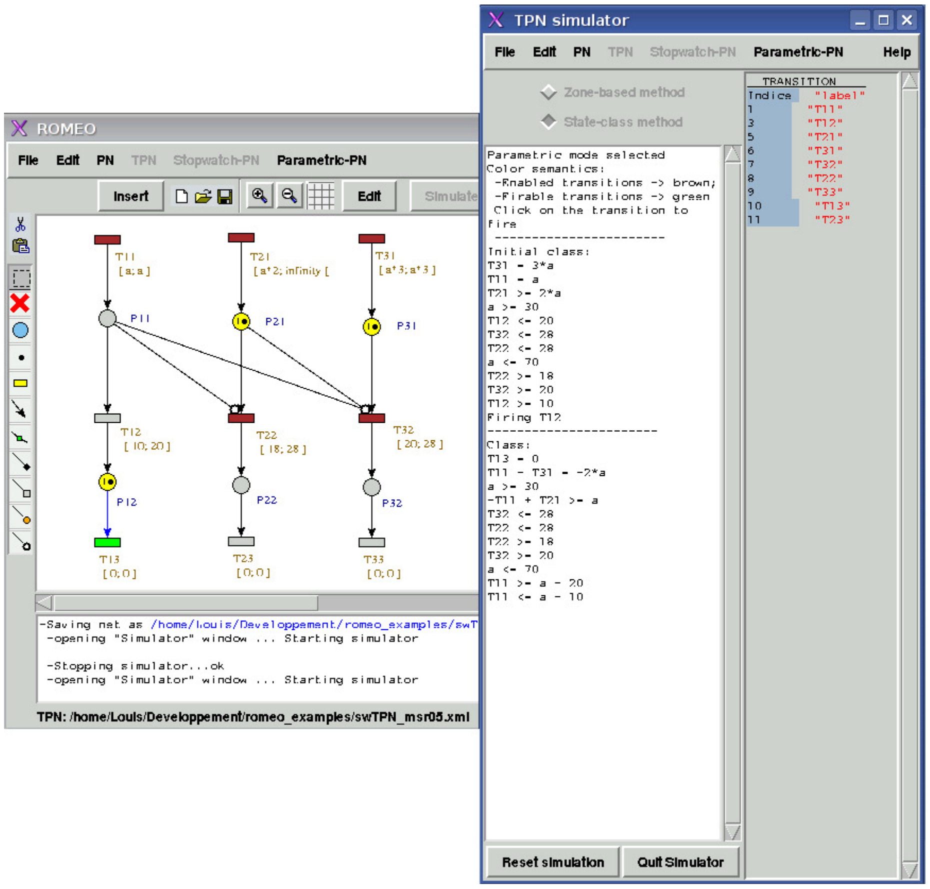
Then, we can perform model-checking on the model. The interesting problems on this system first concern the schedulability of the three tasks, which is expressed by the property that the *PITPN* model is safe (i.e. $1-$bounded). We can verify this property in Romeo with a TCTL formula:

$$\forall P_i,\ \forall\Box_{[0,\infty[}(M(P_i) \leq 1)$$

The result of the parametric model-checking is $a > 48$, as shown in Figure 4.

We can add this new constraint in Romeo and verify new properties on the system which is now schedulable. For example, we can compute the worst case response time (WCRT) of $task_3$ with the parametric TCTL formulae:

$$M(P_{31}) > 0 \leadsto_{[0,b]} M(P_{32}) > 0$$



```
-Saving net as /home/Louis/Developpement/romeo_examples/swT
 -opening "Simulator" window ... Starting simulator

 -Stopping simulator...ok
 -opening "Simulator" window ... Starting simulator
```

TPN: /home/Louis/Developpement/romeo_examples/swTPN_msr05.xml

```
Parametric mode selected
Color semantics:
 -Enabled transitions -> brown;
 -Firable transitions -> green
 Click on the transition to
fire
 ------------------------
Initial class:
T31 = 3*a
T11 = a
T21 >= 2*a
a >= 30
T12 <= 20
T32 <= 28
T22 <= 28
a <= 70
T22 >= 18
T32 >= 20
T12 >= 10
Firing T12
-----------------------
Class:
T13 = 0
T11 - T31 = -2*a
a >= 30
-T11 + T21 >= a
T32 <= 28
T22 <= 28
T22 >= 18
T32 >= 20
a <= 70
T11 >= a - 20
T11 <= a - 10
```

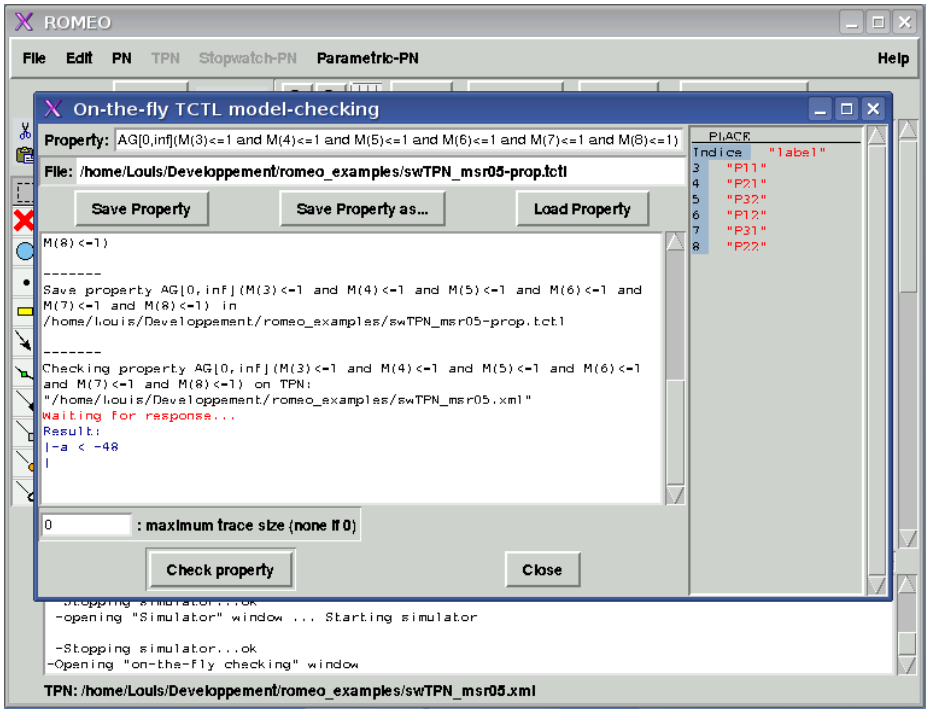
TRANSITION	
Indice	"label"
1	"T11"
3	"T12"
5	"T21"
6	"T31"
7	"T32"
8	"T22"
9	"T33"
10	"T13"
11	"T23"

Fig. 3. ROMEO simulator

This formula uses a new parameter b that is a maximum bound for the WCRT. The result of the parametric model-checking with ROMEO is $b \geq 96$ and so 96 is the WCRT of $task_3$, which is in accordance with [4].

General informations

– ROMEO is available at `http://romeo.rts-software.org/` for Linux, MacOSX and Windows platforms.
– ROMEO has several academic and industrial users.
– Industrial users such as *DGA*, *SODIUS*, *Dassault Aviation* and *EADS* have lead to several industrial contracts. These contracts cover an amount of almost 100k€ and have provided a grant for a Ph.D student.



Property: AG[0,inf](M(3)<=1 and M(4)<=1 and M(5)<=1 and M(6)<=1 and M(7)<=1 and M(8)<=1)

File: /home/Louis/Developpement/romeo_examples/swTPN_msr05-prop.tctl

```
M(8)<=1)

-------
Save property AG[0,inf](M(3)<=1 and M(4)<=1 and M(5)<=1 and M(6)<=1 and
M(7)<=1 and M(8)<=1) in
/home/Louis/Developpement/romeo_examples/swTPN_msr05-prop.tctl

-------
Checking property AG[0,inf](M(3)<=1 and M(4)<=1 and M(5)<=1 and M(6)<=1
and M(7)<=1 and M(8)<=1) on TPN:
"/home/Louis/Developpement/romeo_examples/swTPN_msr05.xml"
Waiting for response...
Result:
|-a < -48
|
```

0 : maximum trace size (none if 0)

PLACE	
Indice	"label"
3	"P11"
4	"P21"
5	"P32"
6	"P12"
7	"P31"
8	"P22"

```
 -opening "Simulator" window ... Starting simulator

 -Stopping simulator...ok
-Opening "on-the-fly checking" window
```

TPN: /home/Louis/Developpement/romeo_examples/swTPN_msr05.xml

Fig. 4. Parametric model-checking