

A method for the efficient computation of an exact state space abstraction for Petri nets with stopwatches

Didier LIME

(w/ Morgan MAGNIN and Olivier H. ROUX, IRCCyN, Nantes, France)

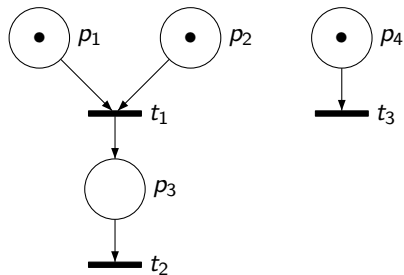
CISS – Aalborg University, DENMARK

27 August 2005

Introduction

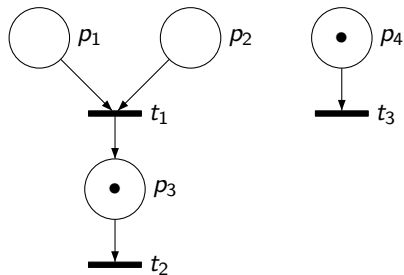
- ▶ Formal verification of real-time systems in the context of a **preemptive** scheduling policy
- ▶ Extensions of Time Petri Nets to model *stopped/resumed* actions
 - ▶ Scheduling Time Petri Nets [Roux, JESA'02]
 - ▶ Preemptive Time Petri Nets [Bucci, TSE'04]
 - ▶ **Inhibitor Hyperarcs Time Petri Nets** [Roux, ICATPN'04]
- ▶ “Everything” is undecidable (even for **bounded** nets)
- ▶ Computing abstractions requires **general convex polyhedra**
- ▶ Contributions
 - ▶ a necessary and sufficient condition for the need of polyhedra
 - ▶ an algorithm taking advantage of it
 - ▶ an implementation

The Model: IHTPNs



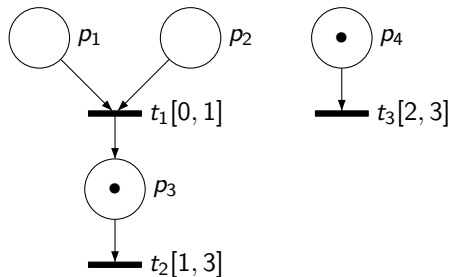
A Petri Net

The Model: IHTPNs



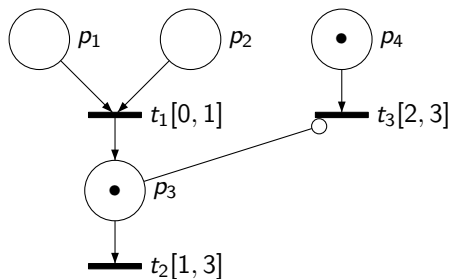
A Petri Net

The Model: IHTPNs



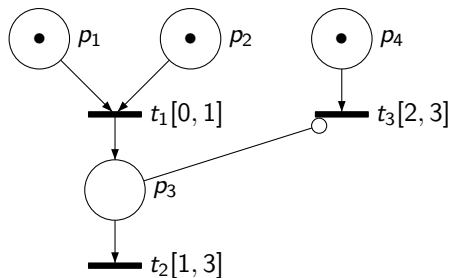
A **Time** Petri Net

The Model: IHTPNs



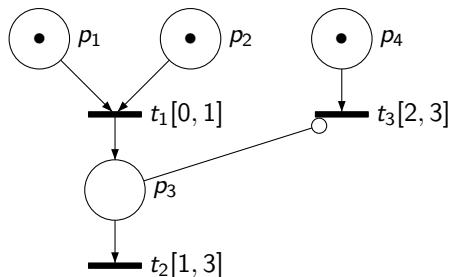
An **Inhibitor Hyperarc** Time Petri Net

IHTPNs: Semantics



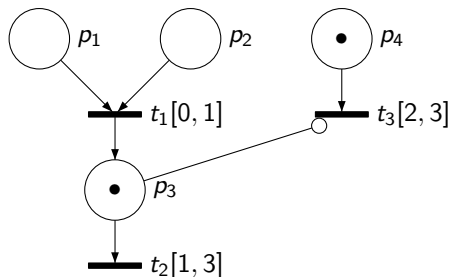
$$\begin{array}{l} \nu(t_1) = 0 \\ \nu(t_3) = 0 \end{array} \xrightarrow{0.7}$$

IHTPNs: Semantics



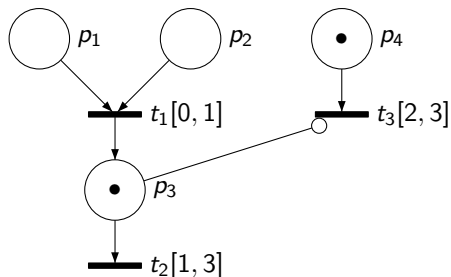
$$\begin{array}{l}
 \nu(t_1) = 0 \\
 \nu(t_3) = 0
 \end{array}
 \xrightarrow{0.7}
 \begin{array}{l}
 \nu(t_1) = 0.7 \\
 \nu(t_3) = 0.7
 \end{array}
 \xrightarrow{t_1}$$

IHTPNs: Semantics



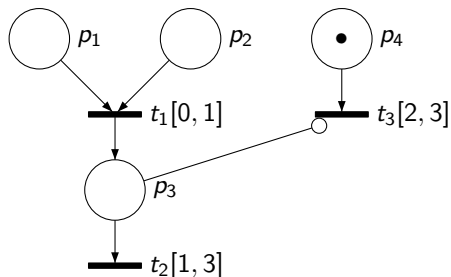
$$\begin{array}{l}
 \nu(t_1) = 0 \\
 \nu(t_3) = 0
 \end{array}
 \xrightarrow{0.7}
 \begin{array}{l}
 \nu(t_1) = 0.7 \\
 \nu(t_3) = 0.7
 \end{array}
 \xrightarrow{t_1}
 \begin{array}{l}
 \nu(t_2) = 0 \\
 \nu(t_3) = 0.7
 \end{array}
 \xrightarrow{1.4}$$

IHTPNs: Semantics



$$\begin{array}{l}
 \nu(t_1) = 0 \\
 \nu(t_3) = 0
 \end{array}
 \xrightarrow{0.7}
 \begin{array}{l}
 \nu(t_1) = 0.7 \\
 \nu(t_3) = 0.7
 \end{array}
 \xrightarrow{t_1}
 \begin{array}{l}
 \nu(t_2) = 0 \\
 \nu(t_3) = 0.7
 \end{array}
 \xrightarrow{1.4}
 \begin{array}{l}
 \nu(t_2) = 1.4 \\
 \nu(t_3) = \mathbf{0.7}
 \end{array}
 \xrightarrow{t_2}$$

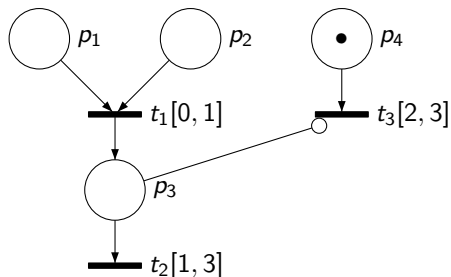
IHTPNs: Semantics



$$\begin{array}{l} \nu(t_1) = 0 \quad \xrightarrow{0.7} \quad \nu(t_1) = 0.7 \quad \xrightarrow{t_1} \quad \nu(t_2) = 0 \quad \xrightarrow{1.4} \quad \nu(t_2) = 1.4 \quad \xrightarrow{t_2} \\ \nu(t_3) = 0 \quad \xrightarrow{\quad} \quad \nu(t_3) = 0.7 \quad \xrightarrow{\quad} \quad \nu(t_3) = 0.7 \quad \xrightarrow{\quad} \quad \nu(t_3) = \mathbf{0.7} \end{array}$$

$$\nu(t_3) = 0.7 \quad \xrightarrow{0.5}$$

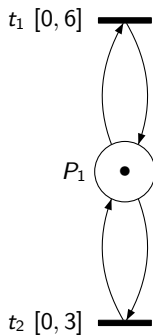
IHTPNs: Semantics



$$\begin{array}{l} \nu(t_1) = 0 \quad \xrightarrow{0.7} \quad \nu(t_1) = 0.7 \quad \xrightarrow{t_1} \quad \nu(t_2) = 0 \quad \xrightarrow{1.4} \quad \nu(t_2) = 1.4 \quad \xrightarrow{t_2} \\ \nu(t_3) = 0 \quad \xrightarrow{\quad} \quad \nu(t_3) = 0.7 \quad \xrightarrow{\quad} \quad \nu(t_3) = 0.7 \quad \xrightarrow{\quad} \quad \nu(t_3) = \mathbf{0.7} \end{array}$$

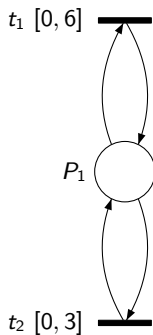
$$\nu(t_3) = 0.7 \quad \xrightarrow{0.5} \quad \nu(t_3) = 1.2 \quad \xrightarrow{t_3}$$

About newly enabled transitions



We fire t_1

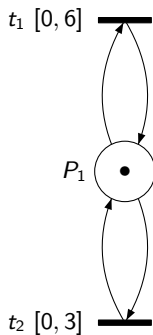
About newly enabled transitions



We fire t_1

t_1 and t_2 are not enabled by $M - \bullet t_1$

About newly enabled transitions

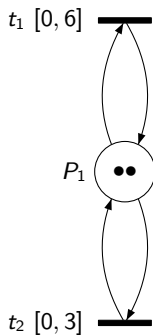


We fire t_1

t_1 and t_2 are not enabled by $M - \bullet t_1$

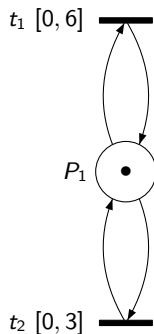
t_1 and t_2 are **newly** enabled

About newly enabled transitions



We fire t_1

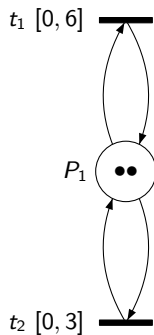
About newly enabled transitions



We fire t_1

t_1 and t_2 are enabled by $M - \bullet t_1$ but t_1 is the fired transition

About newly enabled transitions



We fire t_1

t_1 and t_2 are enabled by $M - \bullet t_1$ but t_1 is the fired transition
 t_2 **remains** enabled, t_1 is **newly** enabled

Abstractions

- ▶ **Infinite** state-space \Rightarrow Abstractions
- ▶ TPNs: Zone-based simulation graph [Gardey, TPLP'05]
- ▶ TPNs: **State class graph** [Berthomieux, TSE'91]
- ▶ TPNs w/ stopwatches (IHTPNs, . . .): **State class graph** [Lime, FET'03, Roux, ICATPN'04, Bucci, TSE'04]

Basic Algorithm

begin

$Passed = \emptyset$

$Waiting = \{C_0\}$

while $Waiting \neq \emptyset$

$C = \text{pop}(Waiting)$

$Passed = Passed \cup C$

for t firable from C

$C' = \text{AbstractSuccessor}(C, t)$

if $C' \notin Passed$

$Waiting = Waiting \cup C'$


end if

end for

end while

end

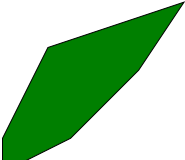
State Class

$$C = \left\{ \left(\begin{array}{c} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{array} \right), \text{  \right\}$$

TPNs: Zone (encoded by a Difference Bound Matrix (DBM) $[d_{ij}]_{i,j \in [0..n]}$):

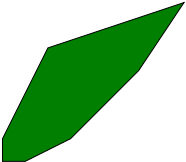
$$\begin{cases} -d_{0i} \leq \theta_i - \mathbf{0} \leq d_{i0}, \\ \theta_i - \theta_j \leq d_{ij} \end{cases}$$

State Class

$$C = \left\{ \left(\begin{array}{c} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{array} \right), \text{  \right\}$$


IHTPNs: General polyhedron: $A\bar{\Theta} \leq B$

Over-approximation

$$C = \left\{ \left(\begin{array}{c} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{array} \right), \text{  \right\}$$

Over-approximation using the **smallest englobing** zone

Over-approximation

$$C = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, \text{  \end{pmatrix}$$

Over-approximation using the **smallest englobing** zone

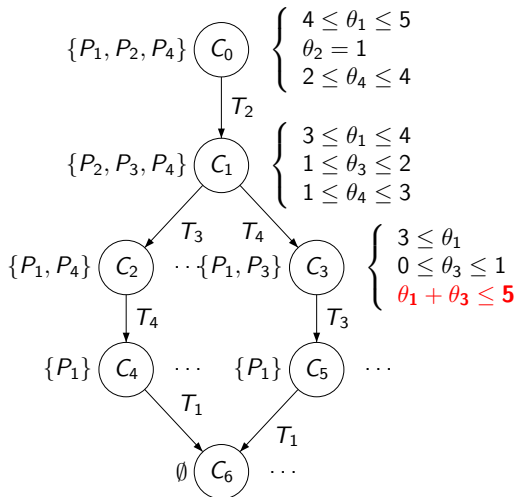
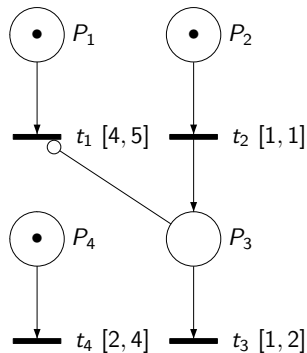
Computing the state class graphs

Let $C = (M, D)$ and $D = (A.\Theta \leq B)$. We fire t_f .

- ▶ $M' = M - \bullet t_f + t_f \bullet$
- ▶ D' is computed by:
 - ▶ for all **non-inhibited** enabled transitions t_i , constrain by $\theta_f \leq \theta_i$
 - ▶ for all **non-inhibited** enabled transitions t_i , $\theta'_i = \theta_i - \theta_f$
 - ▶ eliminate variables for **disabled** transitions (e.g. using Fourier-Motzkin method)
 - ▶ add new variables for **newly** enabled transitions t_i :

$$\alpha(t_i) \leq \theta_i \leq \beta(t_i)$$

Example



A Comparison

For **bounded** IHTPNs (n : number of clocks *i.e.* of enabled transitions):

	Memory	Time	properties
Exact comp.	$O(\exp(n))$	$O(\exp(n))$	LTL/CTL
Over-appr.	$O(n^2)$	$O(n^3)$	LTL/CTL (safety)

	#markings	#polyhedra
Exact comp.	bounded	unbounded
Over-appr.	unbounded	bounded

The Way of the Middle (1/2)

For IHTPNs:

- ▶ The polyhedron in the **initial** state class is **always a zone**.
- ▶ The successor of a non-zone polyhedron might be a zone
- ▶ The successor of a zone might not be a zone

The Way of the Middle (1/2)

For IHTPNs:

- ▶ The polyhedron in the **initial** state class is **always a zone**.
- ▶ The successor of a non-zone polyhedron might be a zone
- ▶ The successor of a zone might not be a zone

We want to start to compute with zones and fall back to general polyhedra when needed (and return to zones asap).

The Way of the Middle (1/2)

For IHTPNs:

- ▶ The polyhedron in the **initial** state class is **always a zone**.
- ▶ The successor of a non-zone polyhedron might be a zone (**easy to check**: $O(n^2)$)
- ▶ The successor of a zone might not be a zone (**not so easy to check**)

We want to start to compute with zones and fall back to general polyhedra when needed (and return to zones asap).

Abstractions: The Way of the Middle (2/2)

$D = [d_{ij}]_{i,j \in [0..n]}$ and $(M', D') = \text{AbstractSuccessor}((M, D), t_f)$.

D' is **not a zone** iff there are at least three enabled transitions t_i, t_j, t_k in D such that:

1. t_i, t_j, t_k are not disabled when firing t_f
2. t_i and t_k are not inhibited
3. t_j is inhibited
4. $d_{j0} + d_{ki} > d_{k0} + d_{ji}$ or $d_{0j} - d_{ik} < d_{0k} - d_{ij}$

Abstractions: The Way of the Middle (2/2)

$D = [d_{ij}]_{i,j \in [0..n]}$ and $(M', D') = \text{AbstractSuccessor}((M, D), t_f)$.

D' is **not a zone** iff there are at least three enabled transitions t_i, t_j, t_k in D such that:

1. t_i, t_j, t_k are not disabled when firing t_f
2. t_i and t_k are not inhibited
3. t_j is inhibited
4. $d_{j0} + d_{ki} > d_{k0} + d_{ji}$ or $d_{0j} - d_{ik} < d_{0k} - d_{ij}$

Complexity: $O(n^3)$

Experimental results

Scheduling-TPN	Overapproximation			Exact computation					
	Time	Nodes	Transitions	Polyhedral algo			Mixed algo		
				Time	Nodes	Transitions	Time	Nodes	Transitions
Example 1	6.63 s	2260	5700	33.51 s	2260	5700	6.69 s	2260	5700
Example 2	18.88 s	11167	25856	80.9 s	11167	25856	19.08 s	11167	25856
Example 3	6.45 s	2225	5371	23.51 s	2225	5371	6.61 s	2225	5371
Example 4	85.52 s	15178	49135	NA	NA	NA	85.59 s	15178	49135
Example 5	99.66 s	16323	54688	NA	NA	NA	99.73 s	16323	54688
Example 6	NA	NA	NA	0.19 s	19	24	0.19 s	19	24
Example 7	NA	NA	NA	26.92 s	4528	8699	13.24 s	4528	8699
Example 8	NA	NA	NA	NA	NA	NA	115.11 s	16650	32865
Example 9	1.72	478	1137	NA	NA	NA	NA	NA	NA

A Tool: Romeo

Features:

- ▶ **Graphical** editing of (Stopwatch) Time Petri Nets
- ▶ Symbolic **simulation**
- ▶ (A fragment of) **TCTL** Model-checking
- ▶ **State-space** computations
 - ▶ timed automata-like
 - ▶ State class-based (for stopwatches: Over-approximation with DBM, exact with polyhedra, exact with mixed method)
- ▶ **Translations** to Timed/Stopwatch Automata
 - ▶ State-space based
 - ▶ Structural

Conclusion and Further Work





Summary:

- ▶ An **exact** but **efficient** computation of the state class abstraction for TPN with **stopwatches**
- ▶ **Polynomial** checks for switching from DBMs to polyhedra (and the other way around)




Further work:

- ▶ Adaptation to timed automata-like forward computation
- ▶ Model checking based on this method
- ▶ Investigation of discrete time

References

-  Roux, O. and D. Lime, *Time Petri nets with inhibitor hyperarcs. Formal semantics and state space computation*, in: *The 25th International Conference on Application and Theory of Petri Nets, (ICATPN 2004)*, Lecture Notes in Computer Science **3099** (2004), pp. 371–390.
-  Roux, O. H. and A.-M. Déplanche, *A t-time Petri net extension for real time-task scheduling modeling*, *European Journal of Automation (JESA)* **36** (2002).
-  Bucci, G., A. Fedeli, L. Sassoli and E. Vicario, *Time state space analysis of real-time preemptive systems*, *IEEE transactions on software engineering* **30** (2004), pp. 97–111.
-  Berthomieu, B. and M. Diaz, *Modeling and verification of time dependent systems using time Petri nets*, *IEEE transactions on software engineering* **17** (1991), pp. 259–273.

References (Cont.)

-  Berthomieu, B., D. Lime, O. Roux and F. Vernadat, *Reachability problems and abstract state spaces for time Petri nets with stopwatches*, Technical Report 04483, Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS), Toulouse, France (2004).
-  Gardey, G., Roux, O., Roux, O., *State space computation and analysis of time Petri nets*. Theory and Practice of Logic Programming (TPLP). Special Issue on Specification Analysis and Verification of Reactive Systems (2005) to appear.
-  Lime, D. and O. Roux, *Expressiveness and analysis of scheduling extended time Petri nets*, in: *5th IFAC International Conference on Fieldbus Systems and their Applications, (FET 2003)* (2003).