

Structural Translation From Time Petri Nets to Timed Automata

Franck Cassez and Olivier H. Roux

IRCCyN/CNRS
BP 92101
1 rue de la Noë
44321 Nantes cedex 3
France

Automated Verification of Critical Systems (AVoCS'04)
4 September 2004, London (UK)

<http://www.irccyn.ec-nantes.fr>

Contents

1. **Context & Related Work**
2. Time Petri Nets & Timed Automata
3. Translation: TPN to TA
4. Conclusion

Context

■ Petri Nets with time

- **Timed** Petri Nets ([Ramchandani, 1974]) – sharp timing constraints
P-Timed PN = T-Timed PN
- **Time** Petri Nets (TPN) ([Merlin, 1974]) – interval timing constraints
T-TPN \neq P-TPN
Timed PN \subseteq T-TPN and in P-TPN
TPN \subseteq Time Stream Petri Nets ([Diaz & Senac, 1994])

Context

■ Petri Nets with time

- **Timed Petri Nets** ([Ramchandani, 1974]) – sharp timing constraints

P-Timed PN = T-Timed PN

- **T-Time Petri Nets (TPN)** ([Merlin, 1974]) – interval timing constraints

T-TPN \neq P-TPN

Timed PN \subseteq T-TPN and in P-TPN

TPN \subseteq Time Stream Petri Nets ([Diaz & Senac, 1994])

Context

- **Main Results & Tools for T-TPNs [Berthomieu & Diaz, 1991]**
 - **Boundedness for TPNs undecidable**
 - **Reachability for bounded TPNs decidable**
 - **Tools: computation of the state class graph (SCG)**
 - Tina [Berthomieu, 2003]
Computes the SCG, **untimed CTL*** model-checking
 - Roméo [Gardey et al., 2003]
Computes the SCG, Region Graph, Reachability

Context

- **Main Results & Tools for T-TPNs [Berthomieu & Diaz, 1991]**
 - **Boundedness for TPNs undecidable**
 - **Reachability for bounded TPNs decidable**
 - **Tools: computation of the state class graph (SCG)**
 - Tina [Berthomieu, 2003]
Computes the SCG, **untimed CTL*** model-checking
 - Roméo [Gardey et al., 2003]
Computes the SCG, Region Graph, Reachability
- **Timed Automata [Alur & Dill, 1994]**
Finite Automata extended with real-valued clocks

Context

- **Main Results & Tools for T-TPNs [Berthomieu & Diaz, 1991]**
 - **Boundedness** for TPNs **undecidable**
 - **Reachability** for **bounded** TPNs **decidable**
 - **Tools: computation of the state class graph (SCG)**
 - Tina [Berthomieu, 2003]
Computes the SCG, **untimed CTL*** model-checking
 - Roméo [Gardey et al., 2003]
Computes the SCG, Region Graph, Reachability

- **Main Results & Tools for Timed Automata ([Alur & Dill, 1994]):**
 - **Reachability + Timed CTL** model-checking **decidable**
 - **Tools:**
 - Uppaal [Pettersson & Larsen, 2000]
 - Kronos [Yovine, 1997]
 - Cmc [Laroussinie et al, 1998]

Related Work

- From **1-safe TPN** to TA [Sifakis & Yovine, 1996]
- From bounded TPN to TA [Sava, 2001]
No correctness proof (equivalence of the semantics ?)
- From TPN to TA [Lime & Roux, 2003]
correctness proof (timed bisimilarity)
Enriched SCG = TA \implies heavy computation
Needs a dedicated tool ([Gardey et al., 2003])

Related Work

- Previous approaches:
 - Either restricted to 1-safe TPN
 - No formal correctness proof of the translation
 - Or need to compute the state space of the TPN

Related Work

■ Previous approaches:

- Either restricted to 1-safe TPN
- No formal correctness proof of the translation
- Or need to compute the state space of the TPN

■ Our aim:

- **Structural** translation (no heavy computation)
- **Correctness proof** of the translation (behavioural equivalence)

Related Work

■ Previous approaches:

- Either restricted to 1-safe TPN
- No formal correctness proof of the translation
- Or need to compute the state space of the TPN

■ Our aim:

- **Structural** translation (no heavy computation)
- **Correctness proof** of the translation (behavioural equivalence)

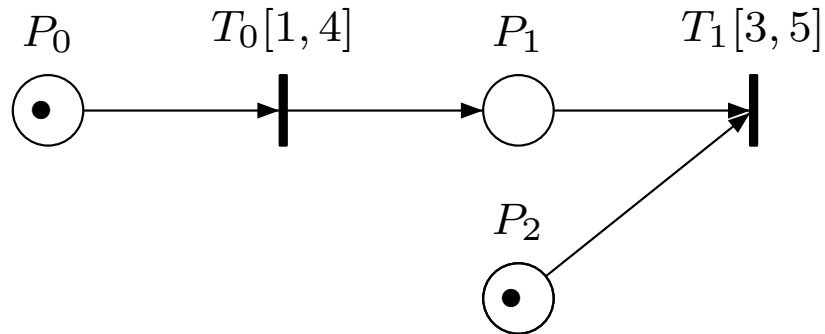
■ Results:

- **Structural** translation
- Applies to **non safe TPNs**
- **Correctness proof** of the translation (behavioural equivalence)
- **Model-checking** of TCTL for **bounded T-TPN**
- Allows to **use efficient tools** for analysis of TA

Contents

1. Context & Related Work
2. Time Petri Nets & Timed Automata
3. Translation: TPN to TA
4. Conclusion

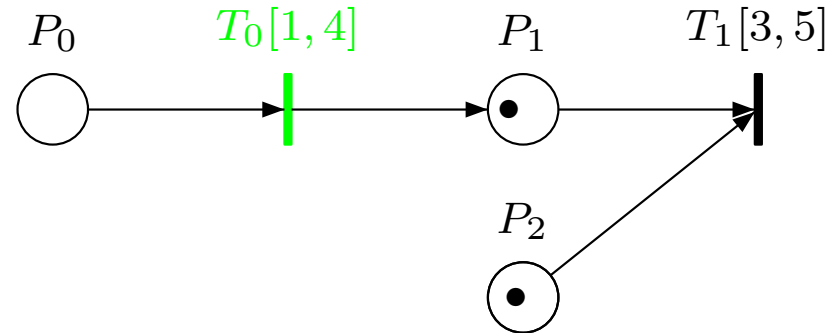
Time Petri Nets – Semantics



■ Initially: $P_0 = P_2 = 1$ at $\delta = 0$

$(P_0 P_2, 0)$

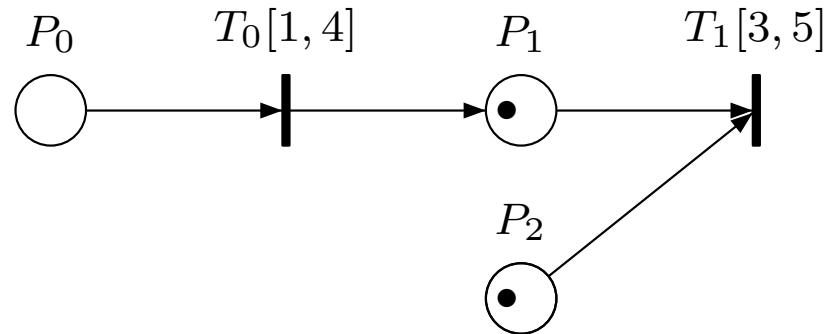
Time Petri Nets – Semantics



- **Initially:** $P_0 = P_2 = 1$ at $\delta = 0$
- $\delta \in [1, 4]$: T_0 enabled; fire T_0 at $\delta = 3.7$

$$(P_0P_2, 0) \xrightarrow{3.7} (P_0P_2, 3.7) \xrightarrow{T_0} (P_1P_2, 3.7)$$

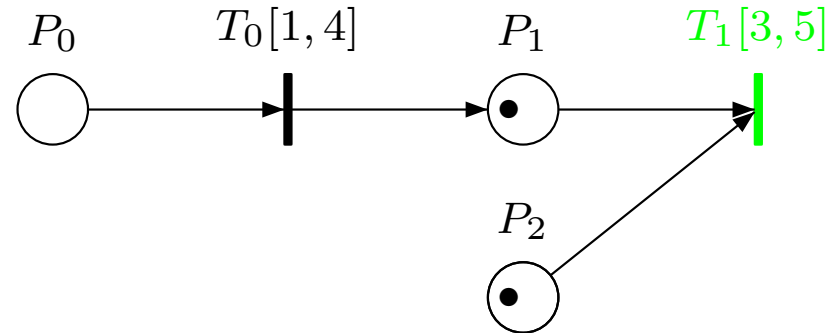
Time Petri Nets – Semantics



- **Initially:** $P_0 = P_2 = 1$ at $\delta = 0$
- $\delta \in [1, 4]$: T_0 enabled; fire T_0 at $\delta = 3.7$
- “untimed” T_1 is enabled \implies clock for T_1 starts

$$(P_0P_2, 0) \xrightarrow{3.7} (P_0P_2, 3.7) \xrightarrow{T_0} (P_1P_2, 3.7)$$

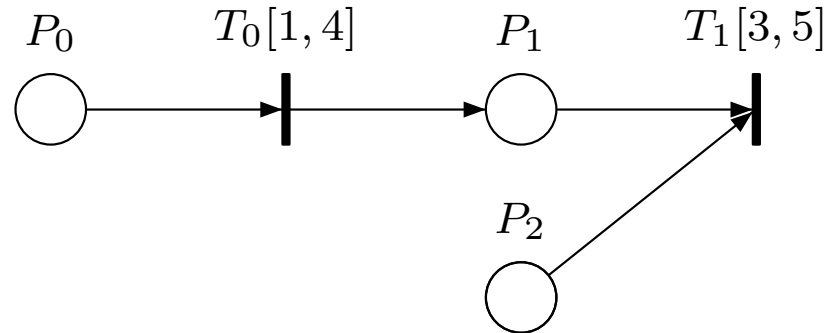
Time Petri Nets – Semantics



- Initially: $P_0 = P_2 = 1$ at $\delta = 0$
- $\delta \in [1, 4]$: T_0 enabled; fire T_0 at $\delta = 3.7$
- “untimed” T_1 is enabled \implies clock for T_1 starts
- after 3 t.u. “timed” T_1 enabled and **must** fire before 5 t.u.

$$(P_0P_2, 0) \xrightarrow{3.7} (P_0P_2, 3.7) \xrightarrow{T_0} (P_1P_2, 3.7) \xrightarrow{3 \leq t \leq 5} (P_1P_2, 3.7 + t)$$

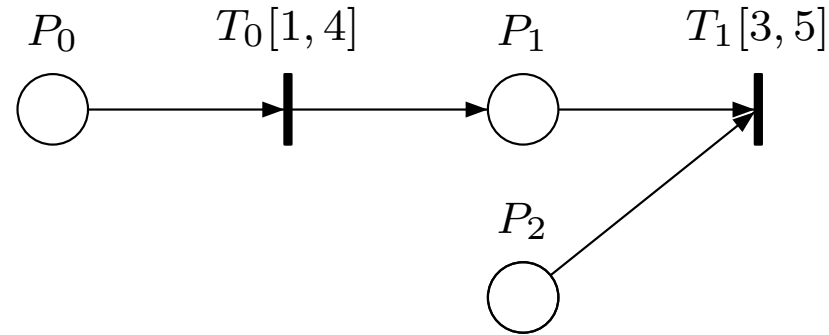
Time Petri Nets – Semantics



- Initially: $P_0 = P_2 = 1$ at $\delta = 0$
- $\delta \in [1, 4]$: T_0 enabled; fire T_0 at $\delta = 3.7$
- “untimed” T_1 is enabled \implies clock for T_1 starts
- after 3 t.u. “timed” T_1 enabled and **must** fire before 5 t.u.
- fire T_1 and time-elapsing

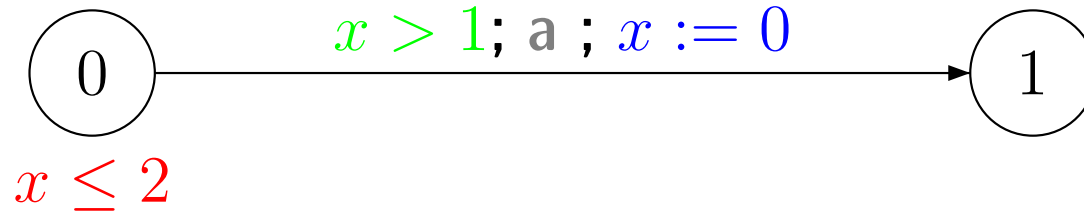
$$\begin{aligned}
 (P_0P_2, 0) &\xrightarrow{3.7} (P_0P_2, 3.7) \xrightarrow{T_0} (P_1P_2, 3.7) \xrightarrow{3 \leq t \leq 5} (P_1P_2, 3.7 + t) \\
 &\xrightarrow{T_1} (\emptyset, 3.7 + t) \xrightarrow{t' \geq 0} (\emptyset, 3.7 + t + t')
 \end{aligned}$$

Time Petri Nets – Semantics



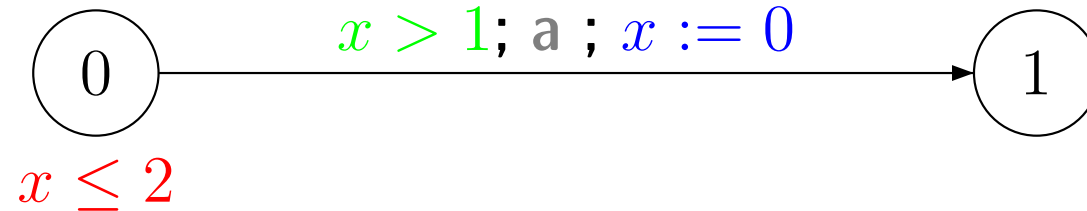
- \mathcal{T} a TPN
- **Semantics** of $\mathcal{T} = \llbracket \mathcal{T} \rrbracket =$ sequence of alternating
 - **Discrete** step
 - **Time** step
- $\llbracket \mathcal{T} \rrbracket =$ **Timed Transition System (TTS)**

Timed Automata [Alur & Dill, 1994]



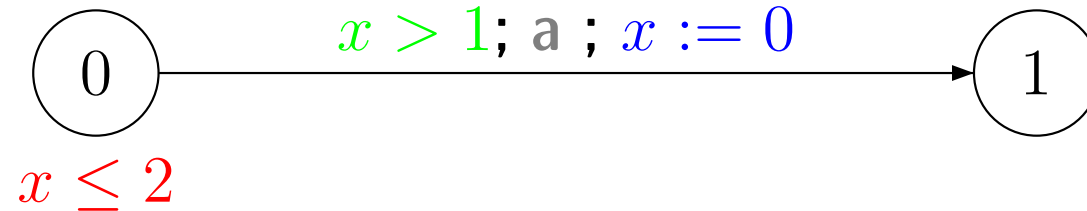
- Finite structure + real-valued clocks

Timed Automata [Alur & Dill, 1994]



- Finite structure + real-valued clocks
- Invariant - Label - Guard - Reset

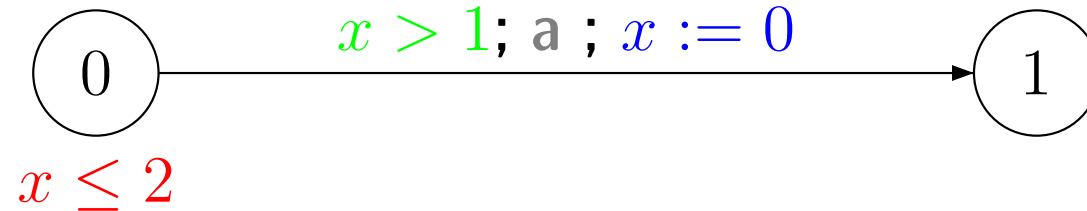
Timed Automata [Alur & Dill, 1994]



- Finite structure + real-valued clocks
- Invariant - Label - Guard - Reset

$(0, x = 0)$

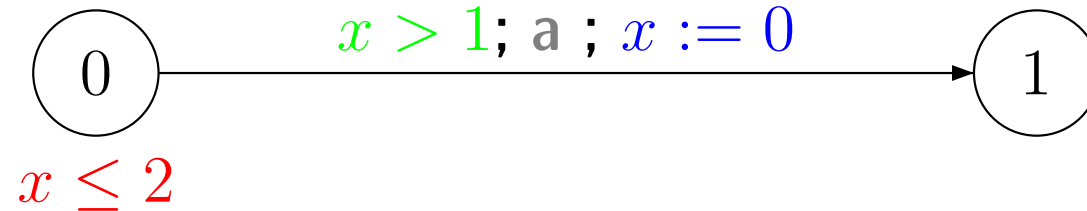
Timed Automata [Alur & Dill, 1994]



- Finite structure + real-valued clocks
- Invariant - Label - Guard - Reset

$$(0, x = 0) \xrightarrow{1.65} (0, x = 1.65)$$

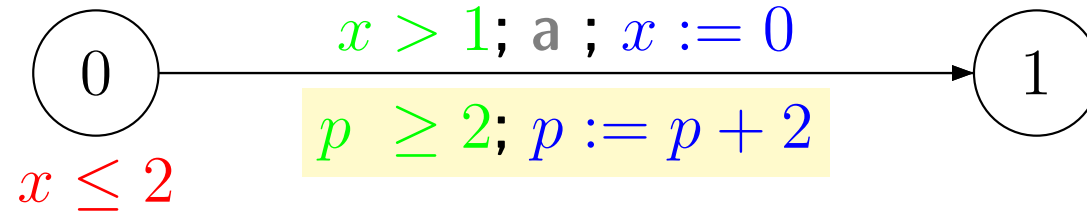
Timed Automata [Alur & Dill, 1994]



- Finite structure + real-valued clocks
- Invariant - Label - Guard - Reset

$$(0, x = 0) \xrightarrow{1.65} (0, x = 1.65) \xrightarrow{a} (1, x = 0) \xrightarrow{t \geq 0} (1, x = t)$$

Timed Automata [Alur & Dill, 1994]

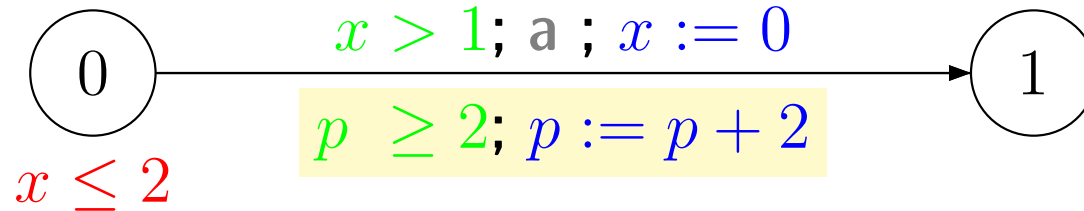


- Finite structure + real-valued clocks
- Invariant - Label - Guard - Reset

$$(0, x = 0) \xrightarrow{1.65} (0, x = 1.65) \xrightarrow{a} (1, x = 0) \xrightarrow{t \geq 0} (1, x = t)$$

- + (arrays of) integer variables

Timed Automata [Alur & Dill, 1994]



- Timed Automata (TA) + **bounded** integer variables
- Semantics of a TA = $\llbracket \mathcal{A} \rrbracket$ = sequence of alternating
 - **Discrete** step
 - **Time** step
- Semantics: $\llbracket \mathcal{A} \rrbracket$ = **Timed Transition System (TTS)**

Contents

1. Context & Related Work
2. Time Petri Nets & Timed Automata
3. Translation: TPN to TA
4. Conclusion

From TPNs to TA

- Given: \mathcal{T} a TPN with n transitions and m places
- \mathbf{p} : array of integers; $\mathbf{p}[i]$ = tokens in place i , $i \in [1..m]$
- \mathcal{A}_i a TA with one clock associated to transition T_i
[\implies TA for a transition]
- SU a cyclic supervisor (4 states) – computes new marking
[\implies TA of the SU]
- Synchronization: $\mathcal{A} = (SU \mid \mathcal{A}_1 \mid \dots \mid \mathcal{A}_n)$
 \mathcal{A} has n clocks
- Discrete step in $\llbracket \mathcal{T} \rrbracket = 4$ discrete steps in $\llbracket \mathcal{A} \rrbracket$

From TPNs to TA

- Given: \mathcal{T} a TPN with n transitions and m places
 - \mathbf{p} : array of integers; $\mathbf{p}[i] =$ tokens in place i , $i \in [1..m]$
 - Synchronization: $\mathcal{A} = (SU \mid \mathcal{A}_1 \mid \cdots \mid \mathcal{A}_n)$
 \mathcal{A} has n clocks
 - Discrete step in $\llbracket \mathcal{T} \rrbracket = 4$ discrete steps in $\llbracket \mathcal{A} \rrbracket$
-
- Results
 1. Theorem 3.2: $\llbracket \mathcal{A} \rrbracket$ and $\llbracket \mathcal{T} \rrbracket$ are timed bisimilar
 2. \mathcal{T} is bounded iff \mathbf{p} is bounded
 3. \mathcal{T} has k reachable markings $\implies \mathcal{A}$ has $\leq 4 \cdot k \cdot n$ discrete states

From TPNs to TA

■ Results

1. **Theorem 3.2:** $\llbracket \mathcal{A} \rrbracket$ and $\llbracket \mathcal{T} \rrbracket$ are **timed bisimilar**
2. \mathcal{T} is bounded iff p is bounded
3. \mathcal{T} has k reachable markings $\implies \mathcal{A}$ has $\leq 4 \cdot k \cdot n$ discrete states

■ Consequences

- Quantitative logic TCTL decidable on **bounded** TPNs
- Efficient translation to Uppaal: Roméo [[Gardey et al., 2003](#)]

From TPNs to TA

■ Results

1. **Theorem 3.2:** $\llbracket \mathcal{A} \rrbracket$ and $\llbracket \mathcal{T} \rrbracket$ are **timed bisimilar**
2. \mathcal{T} is bounded iff \mathcal{p} is bounded
3. \mathcal{T} has k reachable markings $\implies \mathcal{A}$ has $\leq 4 \cdot k \cdot n$ discrete states

■ Consequences

- Quantitative logic TCTL decidable on **bounded** TPNs
- Efficient translation to Uppaal: Roméo [[Gardey et al., 2003](#)]

■ Number of clocks ?

- Useful clocks: only for **enabled** transitions
- **Active clock** reduction
- Use of **active clocks** feature in Uppaal

Contents

1. Context & Related Work
2. Time Petri Nets & Timed Automata
3. Translation: TPN to TA
4. **Conclusion**

Conclusion & Future Work

Summary

- **Formal semantics** for TPNs
- **Structural** translation to TA
 - **Correctness** proof
 - **Model-checking** TCTL

Future Work

- Use on real **case studies** (with Uppaal)
- **Expressiveness**: from TA to TPNs ?

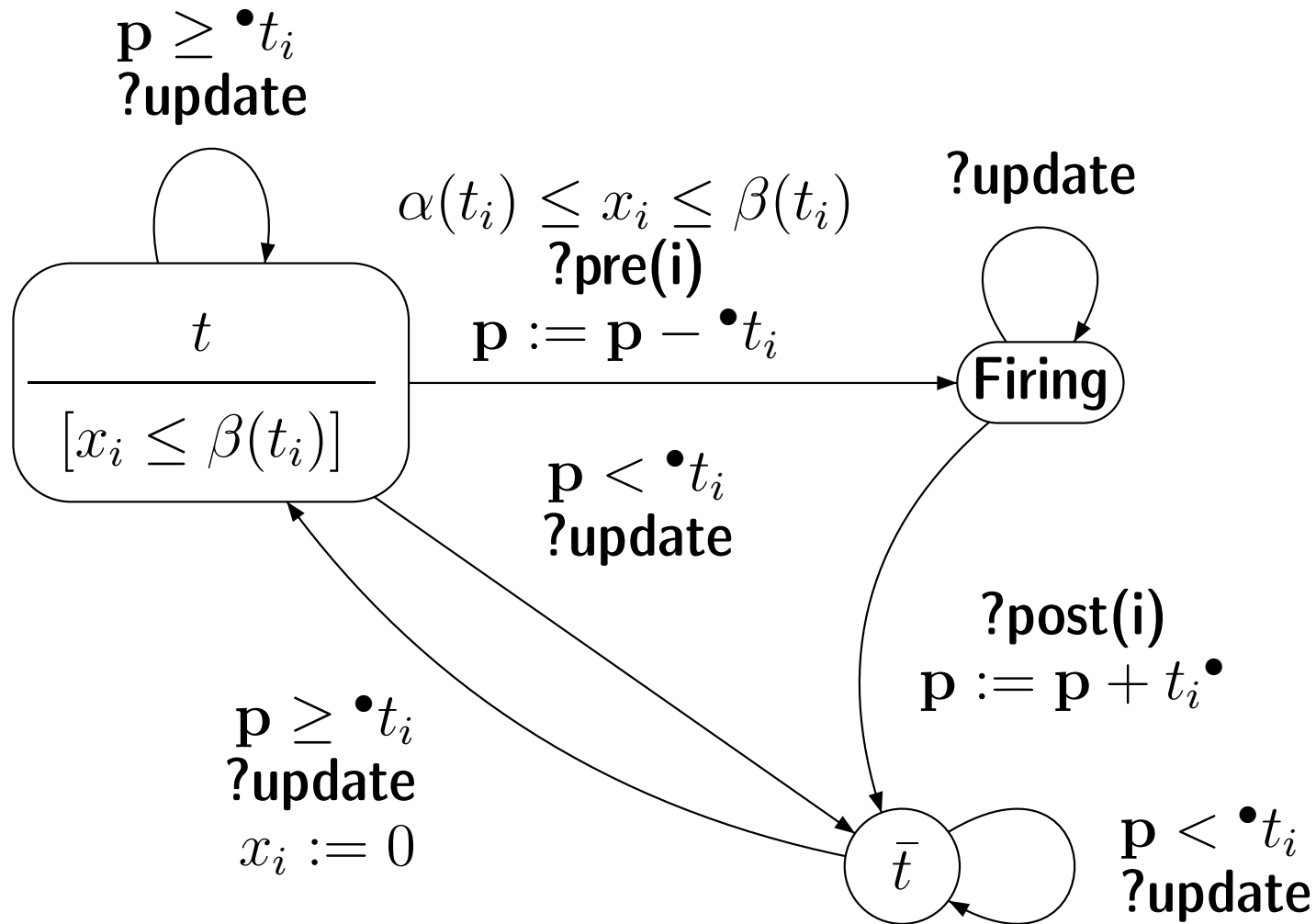
References (1)

- [Alur & Dill, 1994] R. Alur and D. Dill. A theory of timed automata. Theoretical Computer Science B, 126:183–235, 1994. 3, 7
- [Berthomieu & Diaz, 1991] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. IEEE Transactions on Software Engineering, 17(3):259–273, March 1991. 3
- [Diaz & Senac, 1994] M. Diaz and P. Senac. Time stream Petri nets: a model for timed multimedia information. In ATPN'94, volume 815 of LNCS, pages 219–238, 1994. 3
- [Gardey et al., 2003] G. Gardey, D. Lime, and O. (H.) Roux. Roméo: A tool for Time Petri Nets Analysis, 2003. The tool can be freely downloaded from www.irccyn.ec-nantes.fr/irccyn/d/fr/equipes/TempsReel/logs. 3, 4, 9
- [Laroussinie et al, 1998] F. Laroussinie and K. G. Larsen. CMC: A tool for compositional model-checking of real-time systems. In FORTE-PSTV'98, pages 439–456. Kluwer Academic Publishers, 1998. 3
- [Berthomieu, 2003] B. Berthomieu. Tina: Time petri Net Analyzer, 2003. The tool can be freely downloaded from <http://www.laas.fr/tina/>. 3

References (2)

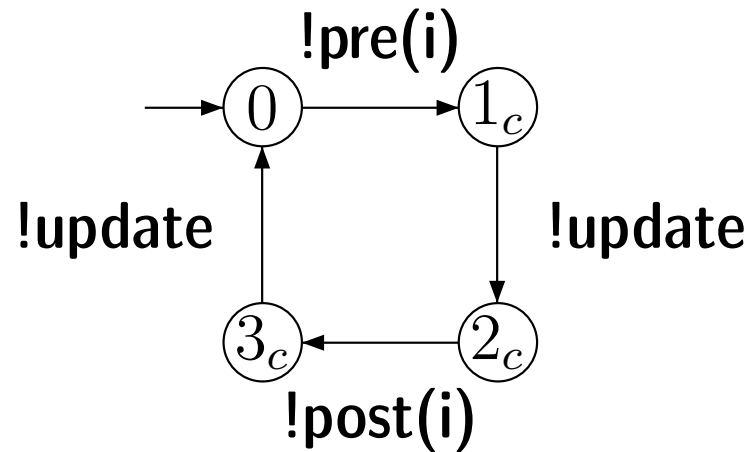
- [Lime & Roux, 2003] D. Lime and O. H. Roux. State class timed automaton of a time Petri net. In PNPM'03. IEEE Computer Society, September 2003. 4
- [Merlin, 1974] P. M. Merlin. A study of the recoverability of computing systems. PhD thesis, University of California, Irvine, CA, 1974. 3
- [Pettersson & Larsen, 2000] P. Pettersson and K. G. Larsen. UPPAAL2k. Bulletin of the European Association for Theoretical Computer Science, 70:40–44, February 2000. 3
- [Ramchandani, 1974] C. Ramchandani. Analysis of asynchronous concurrent systems by timed Petri nets. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974. 3
- [Sava, 2001] A. T. Sava. Sur la synthèse de la commande des systèmes à évènements discrets temporisés. PhD thesis, IINPG, Grenoble, France, november 2001. 4
- [Sifakis & Yovine, 1996] J. Sifakis and S. Yovine. Compositional specification of timed systems. In STACS'96, volume 1046 of LNCS, pages 347–359, 1996. 4
- [Yovine, 1997] S. Yovine. Kronos: A Verification Tool for real-Time Systems. Journal of Software Tools for Technology Transfer, 1(1/2):123–133, October 1997. 3

Automaton for one Transition



(a) The automaton \mathcal{A}_i for transition t_i

Automaton for the Supervisor



(b) Supervisor SU

About Active Clocks

