Model-checking Real-time Systems with Roméo

École d'été Temps Réel 2017

Didier Lime

École Centrale de Nantes – LS2N

Paris, August 29th, 2017

Roméo

- Roméo is a tool for the verification of Time Petri Nets;
- Developed since 2001 by Olivier H. Roux and Didier Lime;
- Written in C++ (engine, ∼24K loc) and Tcl/Tk (GUI, ∼18K loc);
- Distributed under the terms of the CeCILL **open source** license;
- Available at:

  `http://romeo.rts-software.org`

What can we model with Roméo?

- Complex interactions → **Petri nets**;
- Complex discrete behaviors → **discrete variables**;
- Timing uncertainty → **time intervals**;
- Preemptive scheduling → **stopwatches**;
- Design uncertainty → **parameters**;
- Soft real-time constraints, or energy constraints → **cost optimisation**.

Roméo: Some Success Stories

- Analysis of resilience properties in oscillatory **biological** systems [AMI16];
- Environment requirements for an aerial **video tracking** system (with Thales Research) [PRH+16];
- **Operational scenarios** modelling in the DGA OMOTESC project (with Sodius Nantes, Charlotte Seidner's Ph. D.) [Sei09].

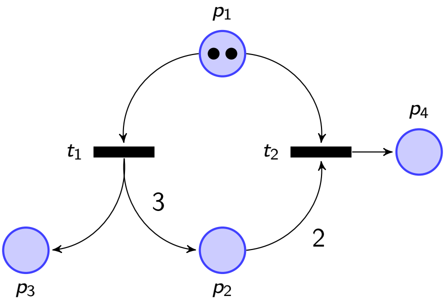
Outline

Introduction

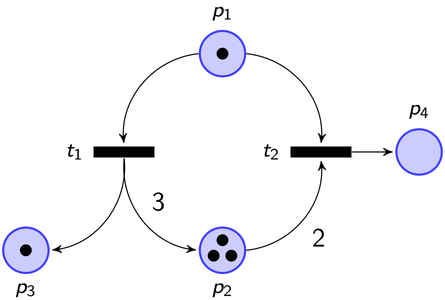
Time Petri Nets

Conclusion

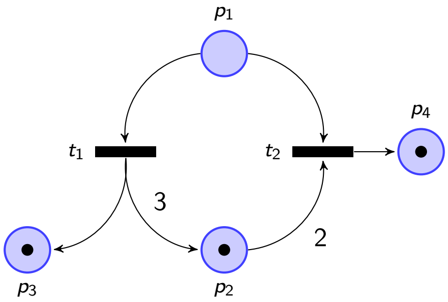
Petri Nets



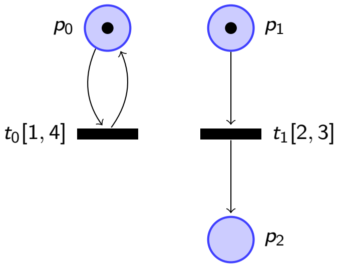
Petri Nets



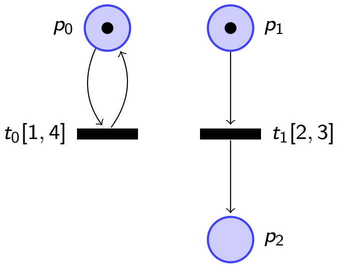
Petri Nets



Time Petri Nets

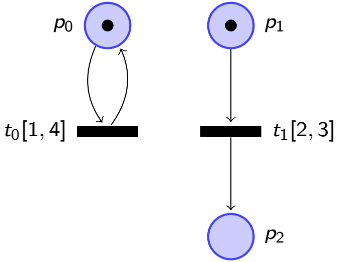


Time Petri Nets



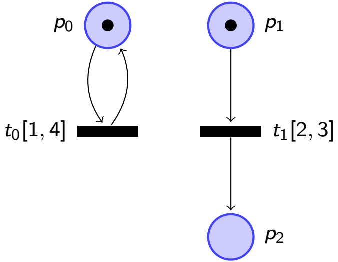
$t_0 \in [1,4]$
$t_1 \in [2,3]$

Time Petri Nets



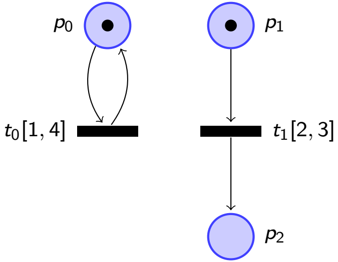
$$\begin{matrix} t_0 \in [1,4] \\ t_1 \in [2,3] \end{matrix} \xrightarrow{1.1} \begin{matrix} [0, 2.9] \\ [0.9,\ 1.9] \end{matrix}$$

Time Petri Nets



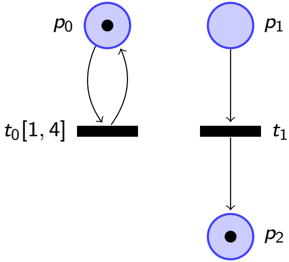
$$\begin{array}{l} t_0 \in [1,4] \\ t_1 \in [2,3] \end{array} \xrightarrow{1.1} \begin{array}{l} [0, 2.9] \\ [0.9,\ 1.9] \end{array} \xrightarrow{t_0} \begin{array}{l} [1,4] \\ [0.9,\ 1.9] \end{array}$$

Time Petri Nets



$$\begin{matrix} t_0 \in [1, 4] \\ t_1 \in [2, 3] \end{matrix} \xrightarrow{1.1} \begin{matrix} [0, 2.9] \\ [0.9,\ 1.9] \end{matrix} \xrightarrow{t_0} \begin{matrix} [1, 4] \\ [0.9,\ 1.9] \end{matrix} \xrightarrow{1.9} \begin{matrix} [0, 2.1] \\ [0,\ 0] \end{matrix}$$

Time Petri Nets

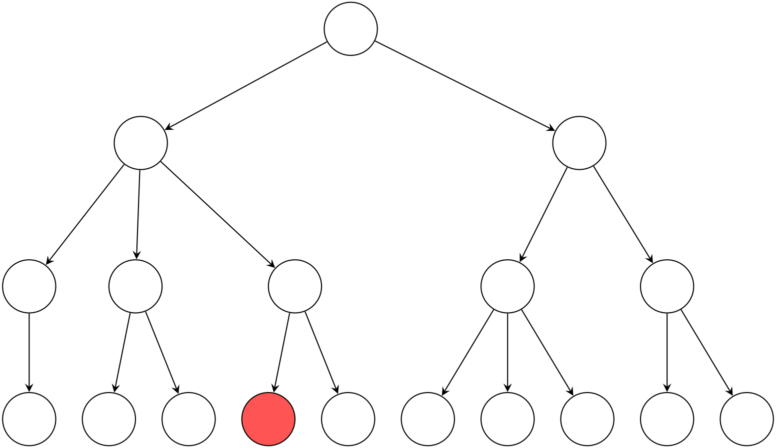


$$\begin{array}{l} t_0 \in [1,4] \\ t_1 \in [2,3] \end{array} \xrightarrow{1.1} \begin{array}{l} [0,2.9] \\ [0.9,\ 1.9] \end{array} \xrightarrow{t_0} \begin{array}{l} [1,4] \\ [0.9,\ 1.9] \end{array} \xrightarrow{1.9} \begin{array}{l} [0,2.1] \\ [0,\ 0] \end{array} \xrightarrow{t_1} \begin{array}{l} [0,\ 2.1] \\ \end{array}$$

Basic Properties

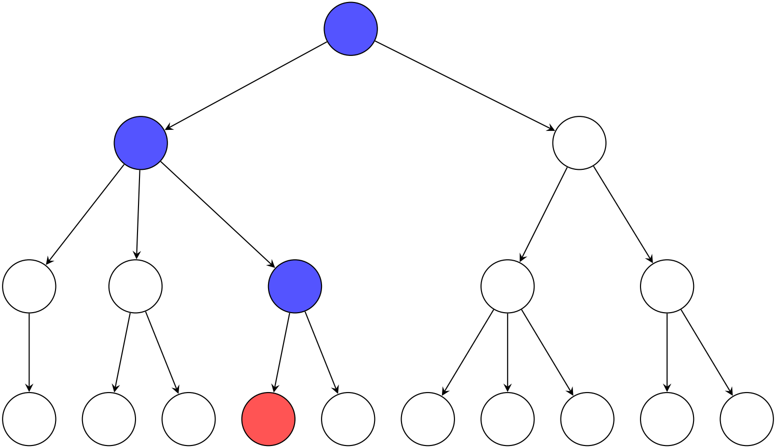
- The **non-nested** fragment of TCTL + (bounded) **response**;
- **Marking** properties are either:
  - linear constraints on the marking: $p_1 + 2 * p_2 > 4$
  - a boundedness property: `bounded(1)`
  - a deadlock property: `deadlock`
- **Temporal** properties (ϕ, ψ are marking properties):
  - $\mathtt{E}\,\phi\,\mathtt{U}\,[3,4]\,\psi$: there is a path on which ψ eventually holds in 3 to 4 t.u. and ϕ holds in the meantime;
  - $\mathtt{A}\,\phi\,\mathtt{U}\,[3,4]\,\psi$: on all paths ψ eventually holds in 3 to 4 t.u. and ϕ holds in the meantime;
  - $\phi -- > [0,5]\psi$: whenever ϕ holds, on all subsequent paths ψ holds within 5 t.u.
- Classic **shorthands**:
  - $\mathtt{EF}\,[3,4]\,\psi = \mathtt{E}\,\mathtt{true}\,\mathtt{U}\,[3,4]\,\psi$: reachability;
  - $\mathtt{AF}\,[3,4]\,\psi = \mathtt{A}\,\mathtt{true}\,\mathtt{U}\,[3,4]\,\psi$: inevitability;
  - $\mathtt{EG}\,\psi = \neg\mathtt{AF}\,(\neg\psi)$: preservability;
  - $\mathtt{AG}\,\psi = \neg\mathtt{EF}\,(\neg\psi)$: safety.

Basic properties



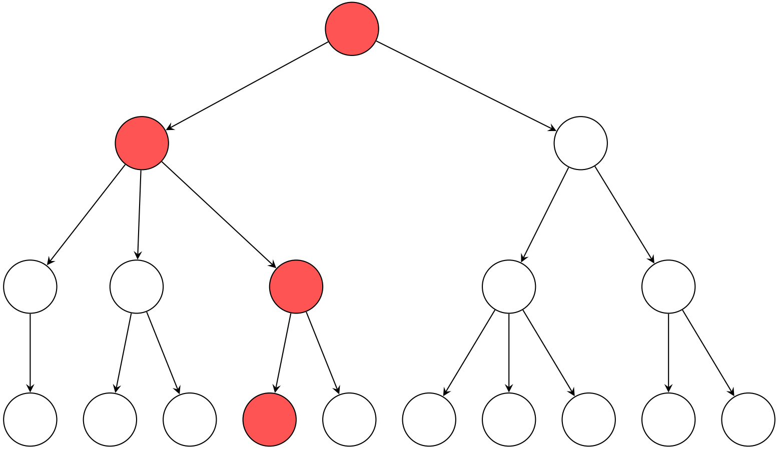
EFφ

Basic properties



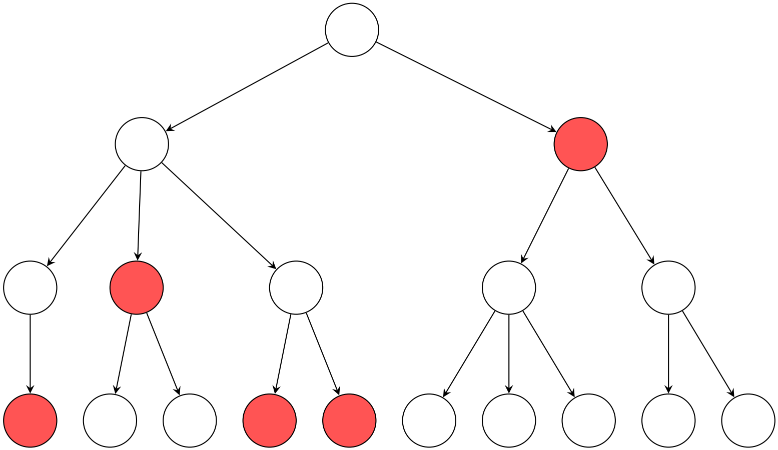
$\mathbf{E}\varphi\mathbf{U}\psi$

Basic properties



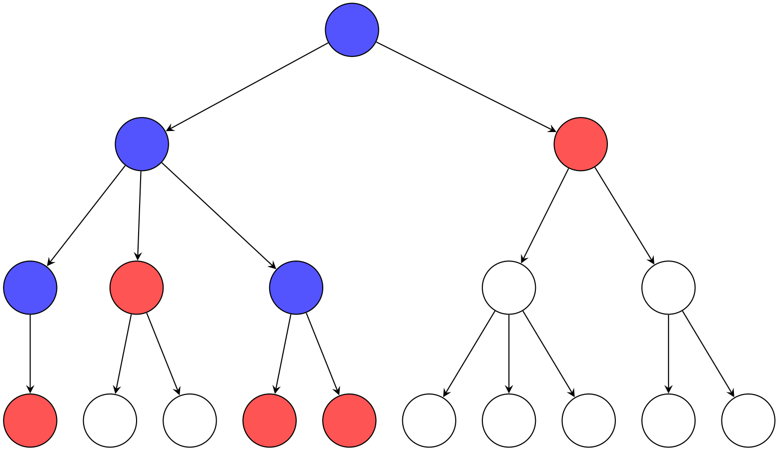
EGφ

Basic properties



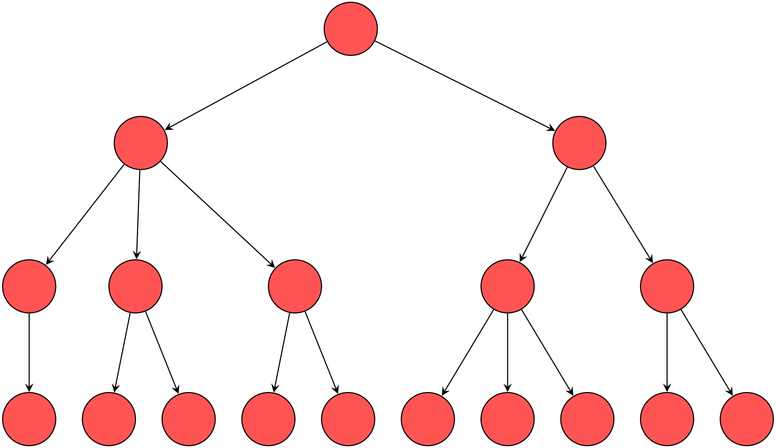
AFφ

Basic properties



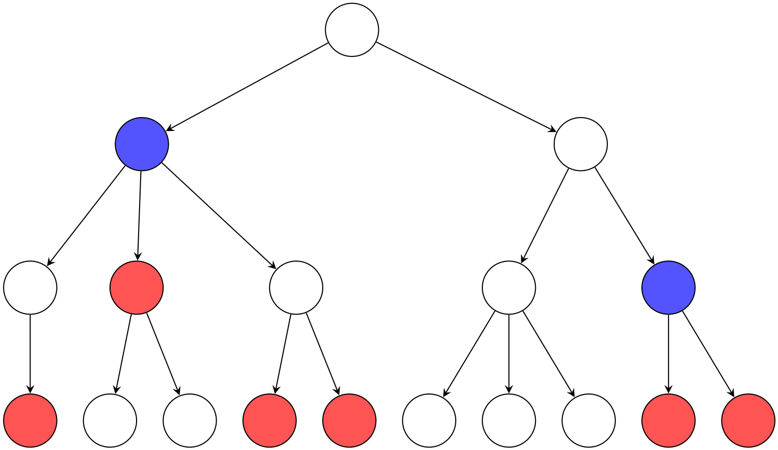
$$\mathbf{A}\varphi\mathbf{U}\psi$$

Basic properties



AGφ

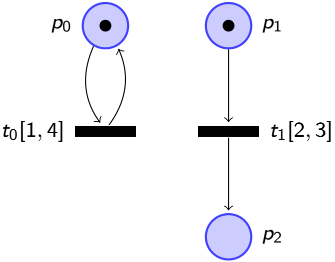
Basic properties



$\varphi \longrightarrow \psi$

State Classes [BD91]

- There is an uncountable number of states even in **bounded** TPNs;
- $\Rightarrow$ group all states obtained by the same sequence of transition firing;



Initially:

$$\begin{cases} 1 \leq t_0 \leq 4 \\ 2 \leq t_1 \leq 3 \end{cases}$$

Fire t_0:

$$\begin{cases} 1 \leq t_0 \leq 4 \\ 2 \leq t_1 \leq 3 \\ \mathbf{t_0 \leq t_1} \end{cases}$$

New times to fire:

$$\begin{cases} 1 \leq t_0 \leq 4 \\ 2 \leq \mathbf{t'_1} + \mathbf{t_0} \leq 3 \\ t_0 \leq \mathbf{t'_1} + \mathbf{t_0} \end{cases}$$

Disabled (incl. t_0):

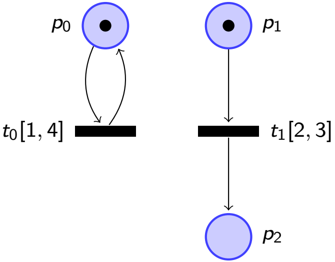
$$\{ \ \mathbf{0} \leq t'_1 \leq \mathbf{2}$$

Newly enabled:

$$\begin{cases} \mathbf{1 \leq t_0 \leq 4} \\ 0 \leq t_1 \leq 2 \end{cases}$$

State Classes [BD91]

- There is an uncountable number of states even in **bounded** TPNs;
- $\Rightarrow$ group all states obtained by the same sequence of transition firing;



Initially:

$$\begin{cases} 1 \leq t_0 \leq 4 \\ 2 \leq t_1 \leq 3 \end{cases}$$

Fire t_0:

$$\begin{cases} 1 \leq t_0 \leq 4 \\ 2 \leq t_1 \leq 3 \\ \mathbf{t_0 \leq t_1} \end{cases}$$

New times to fire:

$$\begin{cases} 1 \leq t_0 \leq 4 \\ 2 \leq \mathbf{t_1' + t_0} \leq 3 \\ t_0 \leq \mathbf{t_1' + t_0} \end{cases}$$

Disabled (incl. t_0):

$$\{ \ \mathbf{0} \leq t_1' \leq \mathbf{2}$$

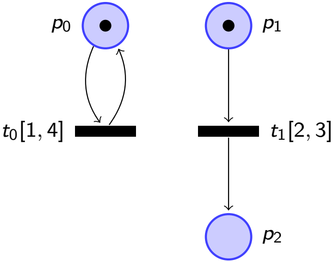
Newly enabled:

$$\begin{cases} \mathbf{1 \leq t_0 \leq 4} \\ 0 \leq t_1 \leq 2 \end{cases}$$

Class firing domains are zones (DBMs).

State Classes [BD91]

- There is an uncountable number of states even in **bounded** TPNs;
- $\Rightarrow$ group all states obtained by the same sequence of transition firing;



Initially:

$$\begin{cases} 1 \leq t_0 \leq 4 \\ 2 \leq t_1 \leq 3 \end{cases}$$

Fire t_0:

$$\begin{cases} 1 \leq t_0 \leq 4 \\ 2 \leq t_1 \leq 3 \\ \mathbf{t_0 \leq t_1} \end{cases}$$

New times to fire:

$$\begin{cases} 1 \leq t_0 \leq 4 \\ 2 \leq \mathbf{t'_1 + t_0} \leq 3 \\ t_0 \leq \mathbf{t'_1 + t_0} \end{cases}$$

Disabled (incl. t_0):

$$\{ \mathbf{0} \leq t'_1 \leq \mathbf{2}$$

Newly enabled:

$$\begin{cases} \mathbf{1 \leq t_0 \leq 4} \\ 0 \leq t_1 \leq 2 \end{cases}$$

Class firing domains are zones (DBMs).
Roméo can also do symbolic simulation using zones *à la* Timed Automata.

Conclusion

- Buy Roméo **now**!
  - Roméo allows for a wide range of analyses on Time Petri Nets (extended with variables);
  - The additional combined availability of costs, parameters, and stopwatches make it unique;
  - It is constantly evolving as a prototype but has good performance and not too many bugs.
- **Next** evolutions and uses:
  - Add timed control, à la Uppaal-Tiga, but with state classes;
  - Add lazy abstraction based algorithms [JL16];
  - Model the multicore version of Trampoline RTOS [TBFR17]

References I

Alexander Andreychenko, Morgan Magnin, and Katsumi Inoue.
Analyzing resilience properties in oscillatory biological systems using parametric model checking.
Biosystems, 149:50 – 58, 2016.
Selected papers from the Computational Methods in Systems Biology 2015 conference.

B. Berthomieu and M. Diaz.
Modeling and verification of time dependent systems using time Petri nets.
IEEE Trans. on Soft. Eng., 17(3):259–273, 1991.

Loïg Jezequel and Didier Lime.
Lazy reachability analysis in distributed systems.
In Josée Desharnais and Rhada Jagadeesan, editors, *The 27th International Conference on Concurrency Theory (CONCUR 2016)*, LIPIcs, Québec City, Québec, Canada, August 2016. Dagstuhl Publishing.

References II

Baptiste Parquier, Laurent Rioux, Rafik Henia, Romain Soulat, Olivier H. Roux, Didier Lime, and Étienne André.
Applying parametric model-checking techniques for reusing real-time critical systems.
In Cyrille Artho and Peter Csaba Ölveczky, editors, *5th International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS 2016)*, Communications in Computer and Information Science, Tokyo, Japan, November 2016. Springer.

Charlotte Seidner.
Vérification des EFFBDs : Model checking en Ingénierie Système. (EFFBDs Verification: Model checking in Systems Engineering).
PhD thesis, University of Nantes, France, 2009.

Toussaint Tigori, Jean-Luc Bechennec, Sebastien Faucou, and Olivier H. Roux.
Formal model-based synthesis of application specific static RTOS.
ACM Transactions on Embedded Computing Systems, 16(4), 2017.