Time and Timed Petri Nets

Serge Haddad

LSV
ENS Cachan & CNRS & INRIA
haddad@lsv.ens-cachan.fr

DISC'11, June 9th 2011

Outline

Time in Discrete Event Systems

Intuitively

A timed execution of a discrete event system (DES) is a finite or infinite sequence of events: $e_1, e_2, \ldots$ interleaved with (possibly null) delays.
(generated by some operational model)

More formally

A timed execution of a DES is defined by two finite or infinite sequences:

- The sequence of states $S_0, S_1, S_2, \ldots$ such that S_0 is the initial state and S_i is the state of the system after the occurrence of e_i.
- The sequence of delays $T_0, T_1, T_2, \ldots$ such that T_0 is the time elapsed before the occurrence of e_0 and T_i is the time elapsed between the occurrences of e_i and e_{i+1}.

Time in Discrete Event Systems

Intuitively

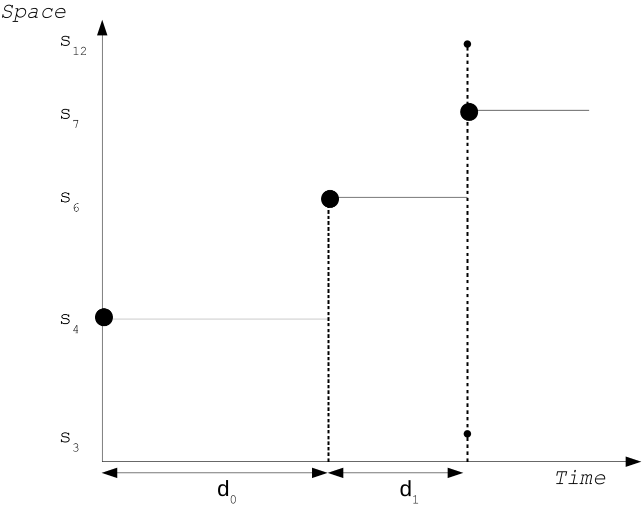
A timed execution of a discrete event system (DES) is a finite or infinite sequence of events: $e_1, e_2, \ldots$ interleaved with (possibly null) delays.
(generated by some operational model)

More formally

A timed execution of a DES is defined by two finite or infinite sequences:

- The sequence of states $S_0, S_1, S_2, \ldots$ such that S_0 is the initial state and S_i is the state of the system after the occurrence of e_i.
- The sequence of delays $T_0, T_1, T_2, \ldots$ such that T_0 is the time elapsed before the occurrence of e_0 and T_i is the time elapsed between the occurrences of e_i and e_{i+1}.

A Timed Execution



$T_0 = d_0$

$T_1 = d_1$

$T_2 = 0$

$T_3 = 0$

$S_0 = s_4$

$S_1 = s_6$

$S_2 = s_3$

$S_3 = s_{12}$

$S_4 = s_7$

Time in Petri Nets

What are the events?

Atomicity versus non atomicity

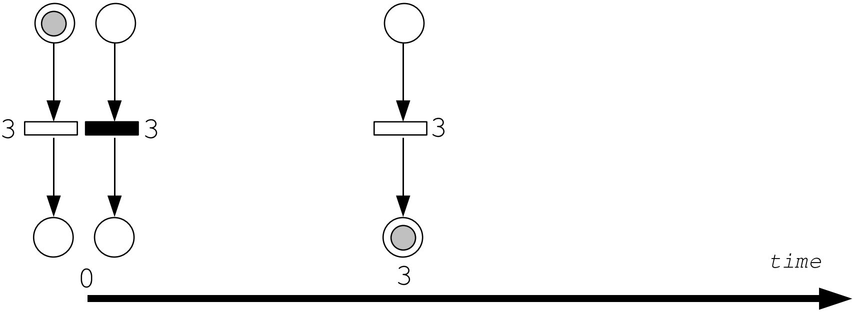
- Beginning and end of transition firings
- Transition firings

What are the delays?

Timing requirements for transition firing

- Duration of transition firing
  (asap requirement)
- Delay before firing
  (requirement between enabling and firing)
- Appropriate age of tokens
  (requirement on tokens)

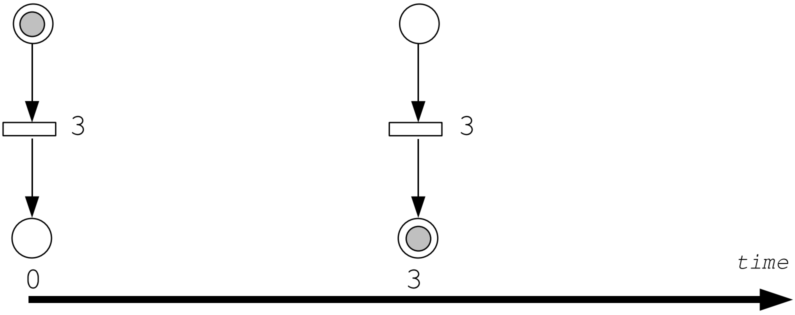
A Duration-Based Semantic



Requires to specify durations

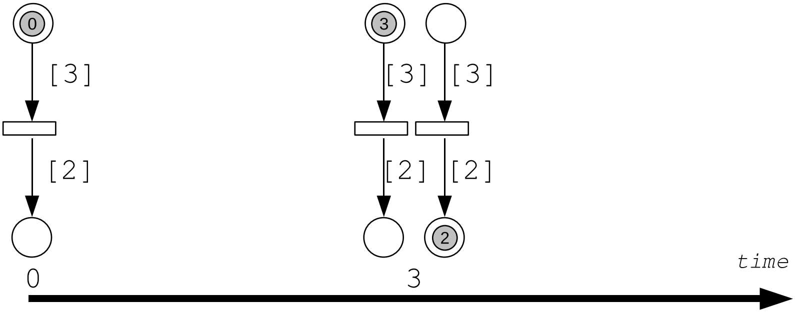
Problem: most of the time, states are not reachable markings of the net

A Delay-Based Semantic



Requires to specify transition delays

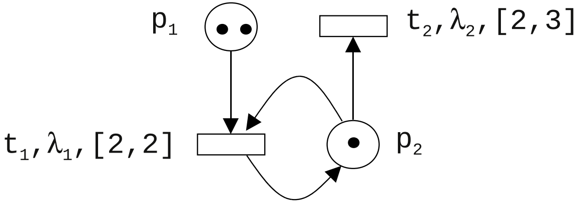
A Token-Based Semantic



Requires to specify age requirements

Outline

Time Petri Net (TPN): Syntax



Places: logical part of the state
Tokens: current value of the logical part of the state

Transitions: events, actions, etc.
Labels: observable behaviour

Arcs: Pre and Post (logical) conditions of event occurrence
Time intervals: temporal conditions of event occurrence

TPN: Transition Occurrence

Logical part

- The logical part of a state (or *configuration*) is a *marking* m, i.e. a number of tokens per place $m(p)$.
- A transition is *enabled* if the tokens required by the preconditions are present in the marking.

Timed part

- There is an implicit clock per enabled transition t and its value $\nu(t)$ defines the timed part of the state. The *clock valuation* ν is the timed part of the configuration.
- An enabled transition t is *firable* if its clock value lies in its interval $[e(t), l(t)]$.

Notation: $(m, \nu) \xrightarrow{t}$

TPN: Change of Configuration

Time elapsing d

- Time may elapse with updates of clocks if every clock value does not go *beyond the corresponding interval*.
- The marking is unchanged $(m, \nu) \xrightarrow{d} (m, \nu + d)$

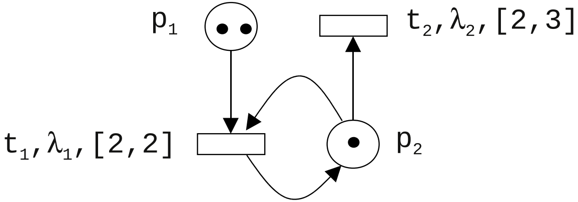
Transition firing t

- Tokens required by the precondition are consumed and tokens specified by the postcondition are produced.
- Clocks values of *newly enabled* transitions are reset leading to valuation ν'.
- Thus $(m, \nu) \xrightarrow{t} (m - Pre(t) + Post(t), \nu')$

A transition t' is newly enabled if

1. t' is enabled in $m - Pre(t) + Post(t)$
2. and t' is disabled in $m - Pre(t)$ or $t' = t$

TPN: an Execution



A maximal time elapsing $(2p_1 + p_2, (0,0)) \xrightarrow{2} (2p_1 + p_2, (2,2))$
before a transition firing $(2p_1 + p_2, (2,2)) \xrightarrow{t_1} (p_1 + p_2, (0,0))$

followed by a (maximal) time elapsing $(p_1 + p_2, (0,0)) \xrightarrow{2} (p_1 + p_2, (2,2))$
before a transition firing $(p_1 + p_2, (2,2)) \xrightarrow{t_1} (p_2, (-,0))$

followed by a non maximal time elapsing $(p_2, (-,0)) \xrightarrow{2.5} (p_2, (-,2.5))$
before a transition firing $(p_2, (-,2.5)) \xrightarrow{t_2} (0, (-,-))$

TPN: an Equivalent Semantic

The timed part is defined by a *dynamic* firing interval $[\overline{e}(t), \overline{l}(t)]$ associated with every enabled transition t.

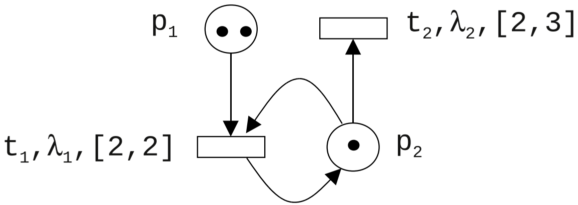
Firing of t

- A transition may fire if it is enabled and $\overline{e}(t) = 0$.
- Intervals of newly enabled transition are reinitialized: $[\overline{e}(t), \overline{l}(t)] := [e(t), l(t)]$.

Time elapsing d

- Time d may elapse if for every enabled transition t, $d \leq \overline{l}(t)$.
- Time intervals are accordingly updated $[\max(0, \overline{e}(t) - d), \overline{l}(t) - d]$.

TPN: Execution Revisited

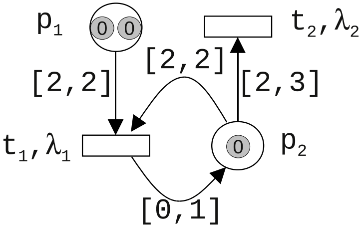


A maximal time elapsing $(2p_1 + p_2, ([2,2],[2,3])) \xrightarrow{2} (2p_1 + p_2, ([0,0],[0,1]))$
before a transition firing $(2p_1 + p_2, ([0,0],[0,1])) \xrightarrow{t_1} (p_1 + p_2, ([2,2],[2,3]))$

followed by a time elapsing $(p_1 + p_2, ([2,2],[2,3])) \xrightarrow{2} (p_1 + p_2, ([0,0],[0,1]))$
before a transition firing $(p_1 + p_2, [0,0],[0,1]) \xrightarrow{t_1} (p_2, (-, [2,3]))$

followed by a non maximal time elapsing $(p_2, (-, [2,3])) \xrightarrow{2.5} (p_2, (-, [0, 0.5]))$
before a transition firing $(p_2, (-, [0, 0.5])) \xrightarrow{t_2} (0, (-, -))$

Timed Petri Net (TdPN): Syntax



Places: both logical and timed part of the state
Tokens: have an age

Transitions: events, actions, etc.
Labels: observable behaviour

Arcs: Pre (resp. Post) conditions of event occurrence
are multisets of timed intervals corresponding to
required (resp. possible) age of consumed (resp. produced) tokens

TdPN: Transition Occurrence

Marking and (simplified) precondition

- The marking of place p, $m(p)$ is a finite multiset of ages

$$m(p) = \sum_{1 \leq i \leq r} a_i.\tau_i \text{ with } r \geq 0 \text{ and } a_i > 0$$

.

- The precondition of a transition t with input place p, $Pre(p,t)$ is an interval.

A transition t is firable if for every input place p of t

There exists an appropriate token, i.e. some i such that $\tau_i \in Pre(p,t)$

Observation: the generalization to bags of intervals is intuitive
but requires technical machinery.

TdPN: Change of Configuration

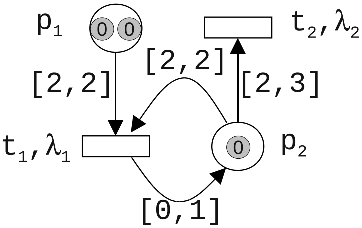
Time elapsing d

- Time may elapse without any restriction.
- The age of tokens is accordingly updated $m \xrightarrow{d} m'$
  such that $m'(p) = \sum_{1 \leq i \leq r} a_i.(\tau_i + d)$ when $m(p) = \sum_{1 \leq i \leq r} a_i.\tau_i$

Transition firing t

- Tokens selected by the precondition are consumed.
- Tokens specified by the postcondition are produced
  with an initial age non deterministically chosen in the corresponding interval.

TdPN: an Execution



A time elapsing $2.(p_1,0)+(p_2,0)\xrightarrow{2}2.(p_1,2)+(p_2,2)$
before a transition firing $2.(p_1,2)+(p_2,2)\xrightarrow{t_1}(p_1,2)+(p_2,0.5)$
(observe that the age of token in p_2 could be different)

followed by a time elapsing $(p_1,2)+(p_2,0.5)\xrightarrow{1.8}(p_1,3.8)+(p_2,2.3)$
(observe that the token in p_1 is dead)

before a transition firing $(p_1,3.8)+(p_2,2.3)\xrightarrow{t_2}(p_1,3.8)$

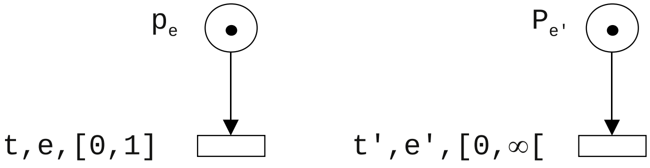
Outline

Limit of TPN: a Concurrent System

How to model this system?

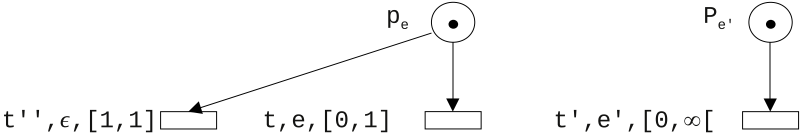
- There are two concurrent events e and e' which may (but have not to) occur.
- e may only occur at instants in $[0, 1]$.
- e' may occur at every instant.

Limit of TPN: a First Modelling



Wrong : e' may only occur after time 1 if e occurs.

Limit of TPN: a Second Modelling



Wrong : $(p_e + p_{e'}, (0,0,0)) \xrightarrow{1} (p_e + p_{e'}, (1,1,1)) \xrightarrow{t''} (p_{e'}, (-,1,-))$
and now at time 1 e can no longer occur.

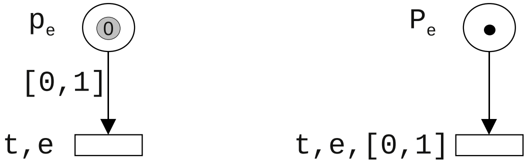
Limit of TPN: Modelling with TdPN

Limit of TdPN: an Urgent Requirement

How to model this system?

- There is a single event e
- which must occur in time interval $[0, 1]$.

Limit of TdPN: Two Modellings



The TPN modelling is correct.

The TdPN modelling is wrong.
More generally there is no way to enforce the firing of a transition.

Outline

Properties

Generic properties

- Reachability Given some state m can the system reach m?
- Coverability Given some state m can the system reach some state "greater or equal than" m?
- Non Termination Does there exist an infinite firing sequence?
- Deadlock Does there exist a state from which no transition will fire?

Specific properties

- Temporal Logic CTL, LTL, CTL*, etc.
  Exemple: Is e eventually followed by e' in every maximal sequence?
- Bisimulation Given two systems, are their discrete behaviours distinguishable by an active observer?
- Timed Temporal Logic TCTL, MTL, MITL, etc.
  Exemple: Is e eventually followed by e' within at most 10 t.u. in every maximal sequence?
- Timed Bisimulation Given two systems, are their timed behaviours distinguishable by an active observer?

Overview

TPN

- In TPNs, all relevant properties are undecidable.
- In *bounded* TPNs, many generic properties are decidable and temporal model checking is decidable.
  (by class graph constructions see later)

TdPN

- In TdPNs, some generic properties like coverability are decidable.
  (see my second talk)
- In TdPNs, some other generic properties like reachability are undecidable.

Principle of Class Graph

Let $T_0 = \{t_1, \ldots, t_k\}$ the set of transitions enabled at m_0
Let x_t be the possible firing delay for $t \in T_0$, the constraint for firing delays is:

$$D_0 \equiv \bigwedge_{t \in T_0} e(t) \leq x_t \leq l(t)$$

In order to fire some t^*, the following system must have a solution.

$$D_{0,t^*} \equiv D_0 \wedge \bigwedge_{t \in T_0 \setminus \{t^*\}} x_{t^*} \leq x_t$$

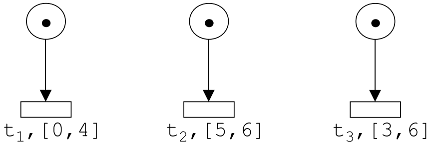
Let $m_0 \xrightarrow{t^*} m_1$ and T_1 be the transitions enabled at m_1 with delays x'_t then:

- If t is newly enabled, the constraint is $C_t \equiv e(t) \leq x'_t \leq l(t)$
- Otherwise the constraints are inherited by $C_t \equiv x'_t = x_t - x_{t^*}$

Consequently, the constraints for firing delays after firing of t^* is

$$D_1 \equiv \exists x_{t_1} \ldots \exists x_{t_k} \; D_{0,t^*} \wedge \bigwedge_{t \in T_1} C_t$$

Class Graph: an Illustration



$$D_0 \equiv 0 \leq x_1 \leq 4 \wedge 5 \leq x_2 \leq 6 \wedge 3 \leq x_3 \leq 6$$

$$D_{0,t_1} \equiv D_0 \wedge x_1 \leq x_2 \wedge x_1 \leq x_3$$

$$D_1 \equiv \exists x_1 \exists x_2 \exists x_3 \; D_{0,t_1} \wedge x_2' = x_2 - x_1 \wedge x_3' = x_3 - x_1$$

Elimination of x_2 and x_3 by substitution

$$D_1 \equiv \exists x_1 \; 0 \leq \; x_1 \leq 4 \wedge 5 \leq x_2' + x_1 \leq 6 \wedge 3 \leq x_3' + x_1 \leq 6$$

Elimination of x_1 by upper and lower bounds

$$D_1 \equiv \exists x_1 \; \max(0, 5 - x_2', 3 - x_3') \leq x_1 \leq \min(4, 6 - x_2', 6 - x_3')$$

$$D_1 \equiv 1 \leq x_2' \leq 6 \wedge 3 \leq x_3' \leq 6 \wedge -3 \leq x_3' - x_2' \leq 1$$

Representation of a Class

A class is defined by:

- A marking m (with T_m the set of enabled transitions);
- A set of variables $\{x_0\} \cup \{x_t\}_{t \in T_m}$ with x_0 denoting the current time;
- A matrix C (called a DBM) representing a set of constraints $C(x_1, \ldots, x_n) \equiv \bigwedge_{i,j} x_j - x_i \leq c_{ij}$

Properties of DBM

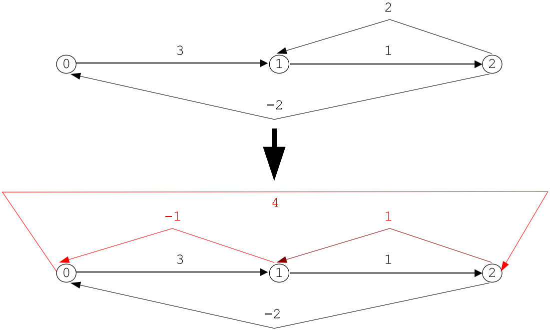
- There exists a canonical representation for non empty DBM;
- Canonization and emptyness checking can be done in polynomial time;
- DBM are effectively closed under:
  1. Projection $\exists x_1\ C(x_1, x_2, \ldots, x_n)$
  2. Relativization $\exists x_1\ C(x_1, x_2 + x_1, \ldots, x_n + x_1)$
  3. Past $\exists d\ C(x_1 + d, x_2 + d, \ldots, x_n + d)$
  4. Future $\exists d\ C(x_1 - d, x_2 - d, \ldots, x_n - d)$
  5. Reset $\exists d\ C(x, x_2, \ldots, x_n) \wedge x_1 = 0$

Canonization: Graph Illustration

Canonization is done by a shortest path computation

Let the constraint be:

$$x_1 - x_0 \leq 3 \wedge -2 \leq x_2 - x_1 \leq 1 \wedge x_0 - x_2 \leq -2$$



Then the canonized constraint is

$$1 \leq x_1 - x_0 \leq 3 \wedge -1 \leq x_2 - x_1 \leq 1 \wedge -4 \leq x_0 - x_2 \leq -2$$

Canonization: the Algorithm

Canonization

For i, j, k such that $i \notin \{j, k\}$ **do**

1. $temp \leftarrow \min(c_{jk}, c_{ji} + c_{ik})$
2. **If** $j \neq k$ **Then** $c_{jk} \leftarrow temp$
   Else If $temp < 0$ **Then** Return(Empty DBM)

Correctness of the shortest path algorithm ...

- The algorithm returns Empty DBM iff there is a negative cycle in the graph.
- Otherwise c_{ij} is the length of a shortest path from x_i to x_j
  and consequently $c_{ij} \leq c_{ij} + c_{jk}$ for all k.

implies correctness of the canonization.

- If there is a negative cycle in the graph there is no solution of the DBM.
  (by transitivity one gets $x_i - x_i < 0$)
- Otherwise for all i, j there is no solution with $x_j - x_i > c_{ij}$
  and a solution with $x_j - x_i = c_{ij}$ *(define $x_i = 0$ and $x_k = c_{ik}$ for all $k \neq i$)*

Properties of the Class Graph

Finiteness for bounded nets

- The number of reachable markings is finite.
- The absolute value of integers occurring in the DBM are bounded by: $\max(\max_{t \in T}(l(t) \mid l(t) \text{ finite}), \max_{t \in T}(e(t) \mid l(t) \text{ infinite}))$

Trace and marking representation

- The untiming of every firing sequence of the TPN is a path of the class graph.
- For every path of the class graph there is at least one corresponding firing sequence.
- Thus the reachable markings are exactly those occurring on the class graph.

Main References

On definition and analysis of TPNs

- B. Berthomieu, M. Menasche, A State enumeration approach for analyzing time Petri nets, 3rd European Workshop on Petri Nets, Varenna, Italy, 1982.
- B. Berthomieu, M. Diaz, Modeling and verification of time dependent systems using time Petri nets. IEEE Transactions on Software Engineering, 17(3):259-273, 1991.

On definition and analysis of TdPNs

- V. Valero, D. Frutos-Escrig, F. R. F. Cuartero. On non-decidability of reachability for timed-arc Petri nets. In Proc. 8th Int. Work. Petri Nets and Performance Models (PNPM'99), pages 188-196. IEEE Computer Society Press, 1999
- P. A. Abdulla, A. Nylén. Timed Petri nets and bqos. In Proc. 22nd International Conference on Application and Theory of Petri Nets (ICATPN'01), volume 2075 of Lecture Notes in Computer Science, pages 53-70. Springer, 2001