Time(d) Petri Net

Serge Haddad

LSV
ENS Paris-Saclay & CNRS & Inria
haddad@lsv.fr

Petri Nets 2018, June 25th 2018

1. Time and Petri Nets
2. Time Petri Net: Syntax and Semantic
3. Analysis of Time Petri Nets
4. More on Difference Bound Matrices
5. Timed Petri Nets: Syntax and Semantics
6. Analysis of Timed Petri Nets

Outline

1. Time and Petri Nets

Time Petri Net: Syntax and Semantic

Analysis of Time Petri Nets

More on Difference Bound Matrices

Timed Petri Nets: Syntax and Semantics

Analysis of Timed Petri Nets

Time in Discrete Event Systems

Intuitively

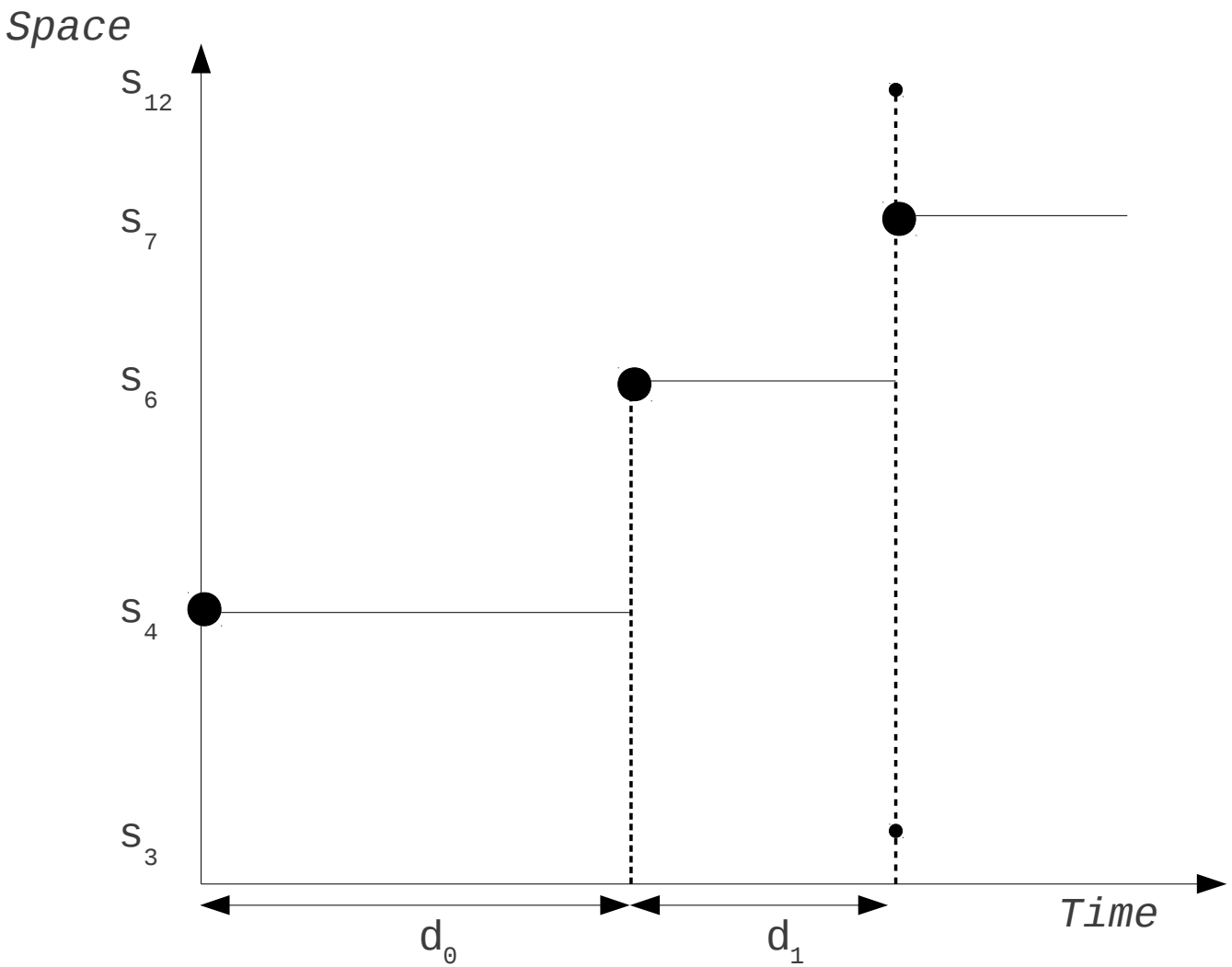
A timed execution of a discrete event system (DES) is a finite or infinite sequence of events: $e_1, e_2, \ldots$ interleaved with (possibly null) delays.
(generated by some operational model)

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:
  1. S_0 is the initial state,
  2. 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:
  1. T_0 is the time elapsed before the occurrence of e_0,
  2. 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

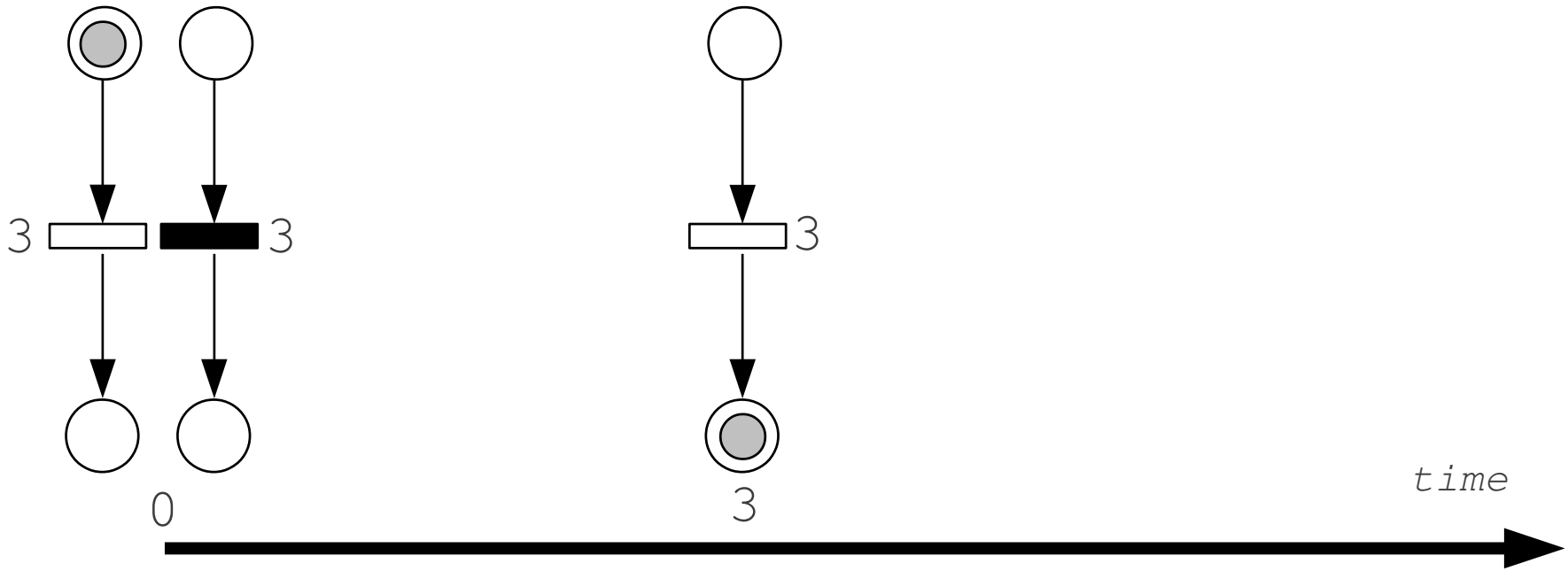
- Beginning and end of transition firings
- Transition firings

What are the delays?

Timing requirements for transition firing

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

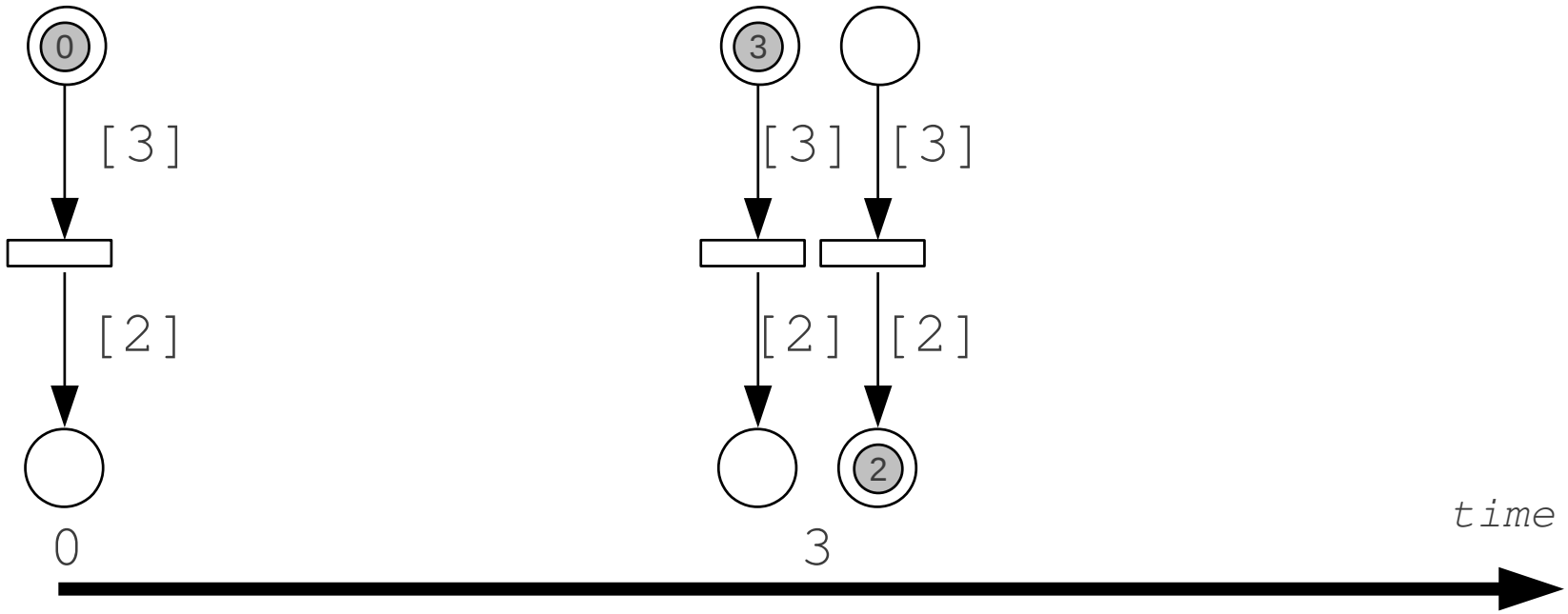
A Duration-Based Semantic



Requires to specify durations.

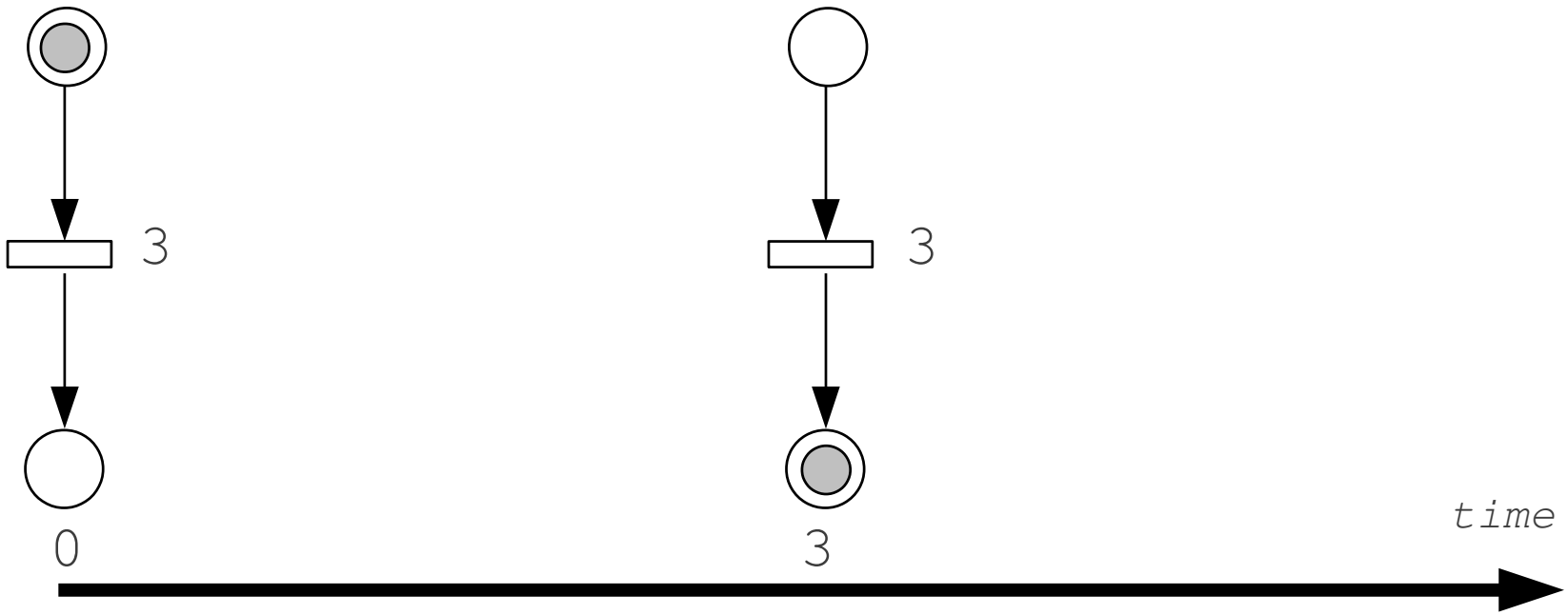
Problem: most of the time, states are not reachable markings of the net.

A Token-Based Semantic



Requires to specify age constraints.

A Delay-Based Semantic



Requires to specify transition delays.

Outline

Time and Petri Nets

2 Time Petri Net: Syntax and Semantic

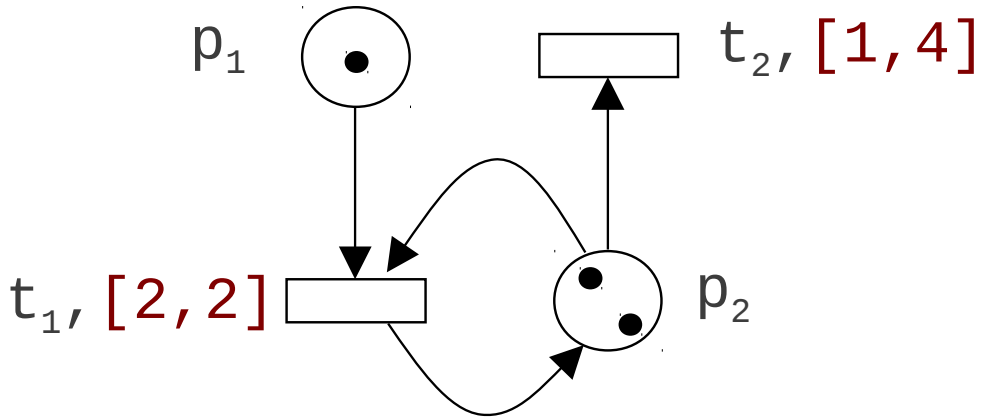
Analysis of Time Petri Nets

More on Difference Bound Matrices

Timed Petri Nets: Syntax and Semantics

Analysis of Timed Petri Nets

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.

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

Time intervals: temporal conditions of event occurrence

TPN: Transition Occurrence

Logical part *(as in Petri nets)*

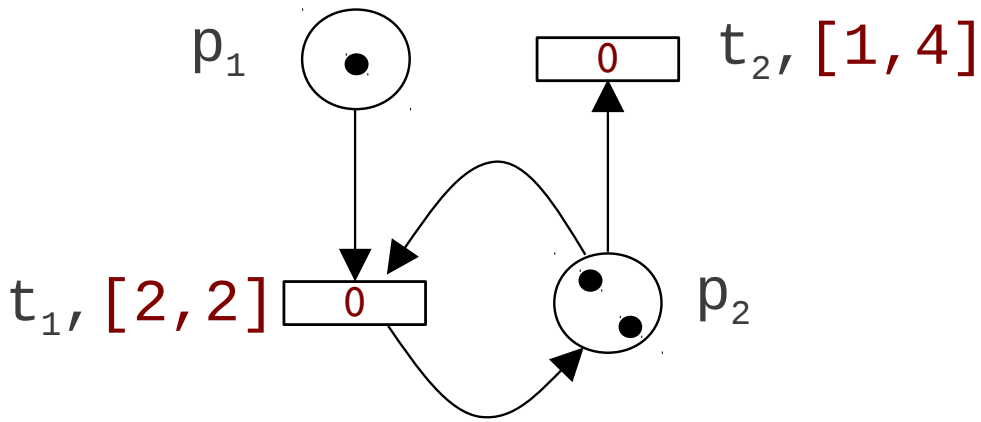
- The logical part of a state (or *configuration*) is a *marking* m,
  i.e. a number of tokens per place $m(p)$.
- A transition is *enabled*
  if the tokens required by the preconditions are present in the marking.

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

Illustration of Transition Occurrence



The initial configuration is $(p_1 + 2p_2, (0, 0))$

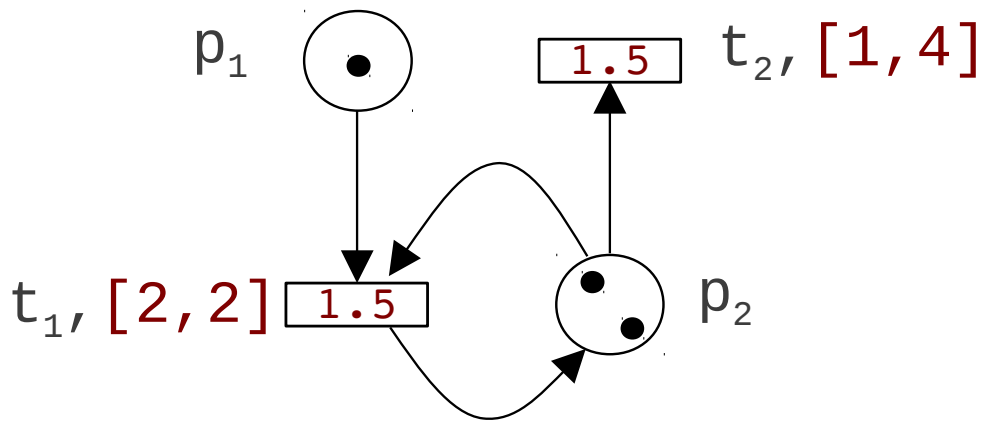
Both t_1 and t_2 are enabled.

None is firable.

Configuration Change by Time Elapsing

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 $(m, \nu) \xrightarrow{d} (m, \nu + d)$



$$(p_1 + 2p_2, (0, 0)) \xrightarrow{1.5} (p_1 + 2p_2, (1.5, 1.5))$$

Now t_2 is firable.

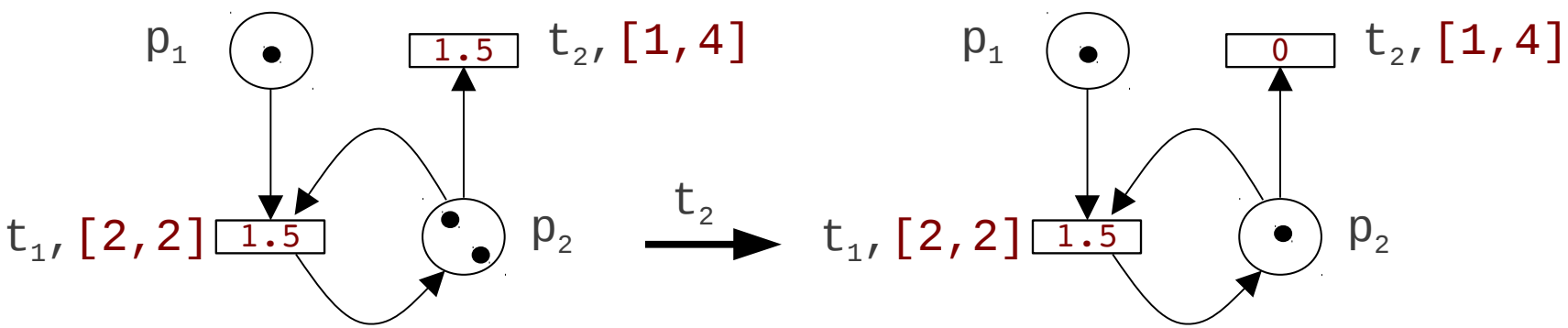
Configuration Change by Transition Firing

Transition firing t

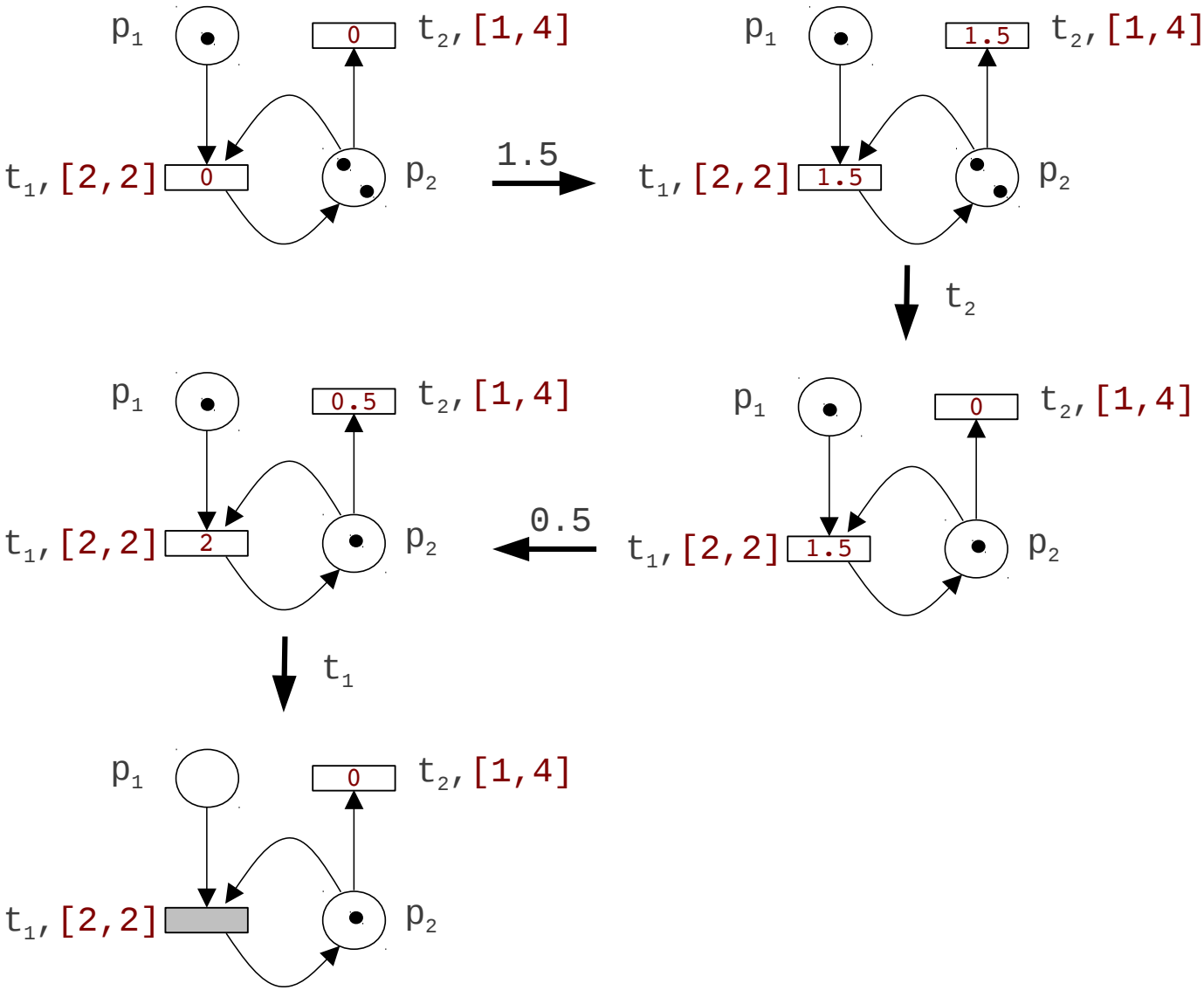
- Tokens required by the precondition are consumed and tokens specified by the postcondition are produced.
- Clocks values of *newly enabled* transitions are reset leading to valuation ν'.
- Thus $(m, \nu) \xrightarrow{t} (m - Pre(t) + Post(t), \nu')$

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 $t' = t$



An Execution



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.

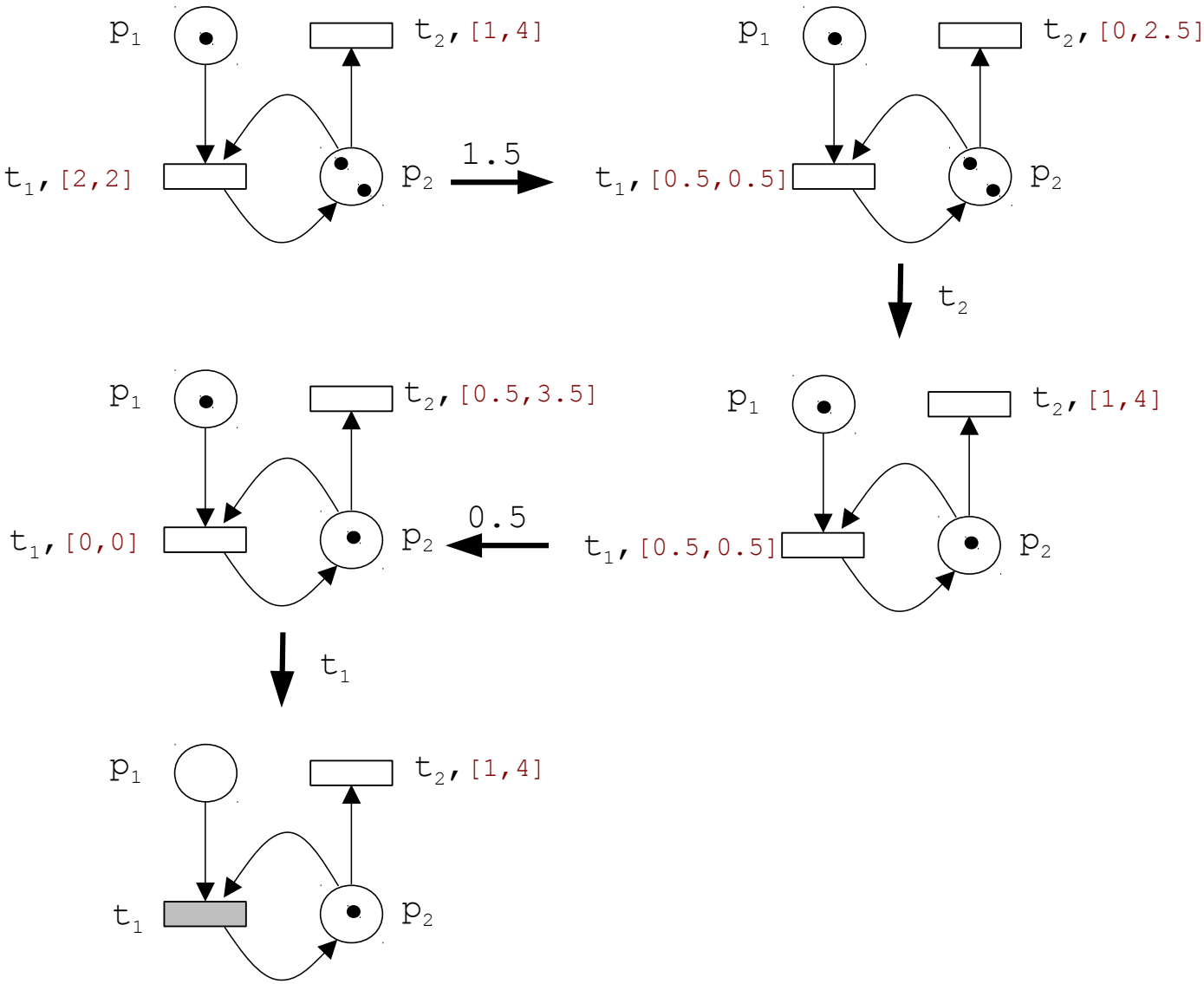
Firing of t

- 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 $[\max(0, \overline{e}(t) - d), \overline{l}(t) - d]$.

Revisiting the Execution



Outline

Time and Petri Nets

Time Petri Net: Syntax and Semantic

3 Analysis of Time Petri Nets

More on Difference Bound Matrices

Timed Petri Nets: Syntax and Semantics

Analysis of Timed Petri Nets

Properties

Generic properties

- Reachability Given some state m can the system reach 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.

Is e eventually followed by e'
in every maximal sequence?

- Timed Temporal Logic TCTL, MTL, MITL, etc.

Is e eventually followed by e' within at most 10 t.u.
in every maximal sequence?

Overview

In TPNs,

all relevant properties are undecidable.

In *bounded* TPNs,

- many generic properties are decidable,
- some temporal model checking is decidable.

Boundedness problem

- Undecidable for TPNs
- A decidable sufficient condition: boundedness of the underlying PN

Key algorithms: class graph constructions

What is a Class?

A class is a finite representation of an infinite set of reachable configurations.

A class is defined by:

- A marking m;
- Let T_m be the set of enabled transitions from m.
  A set of variables $\{x_0 = 0\} \cup \{x_t\}_{t \in T_m}$ with x_t the possible firing delay for t;
- A matrix C *(called a DBM)* representing a set of constraints:

$$C(x_1, \ldots, x_n) \equiv \bigwedge_{i,j} x_j - x_i \leq c_{ij}$$

The Initial Class

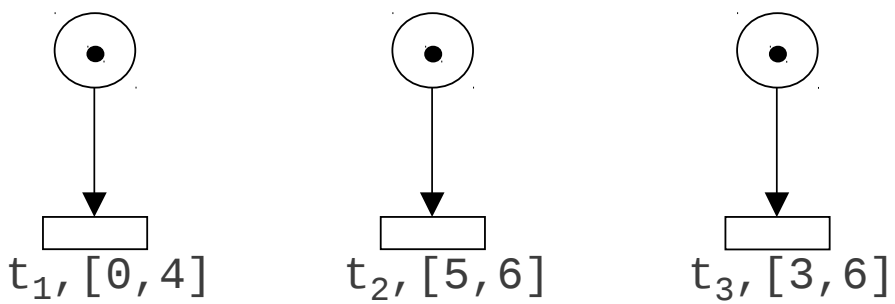
Definition

Let $T_0 = \{t_1, \ldots, t_k\}$ be the set of transitions enabled at m_0.

Then:

$$C(x_{t_1}, \ldots, x_{t_k}) \equiv \bigwedge_{t \in T_0} e(t) \leq x_t \leq l(t)$$

Example



The initial class: $C \equiv 0 \leq x_1 \leq 4 \wedge 5 \leq x_2 \leq 6 \wedge 3 \leq x_3 \leq 6$

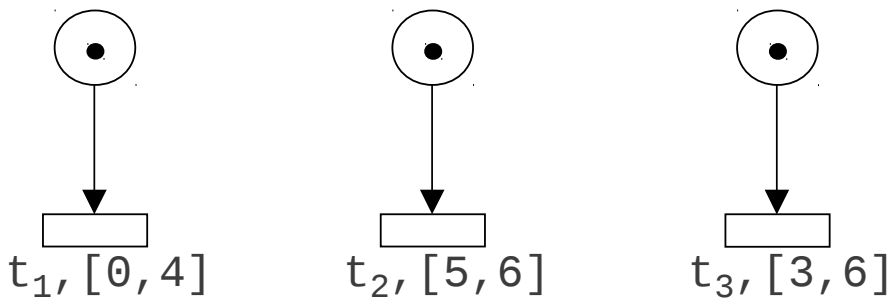
Firability of a Transition from a Class

Firability of t^* from class (m, C)

In order to fire some $t^* \in T_m$, the following system must have a solution:

$$C_{t^*} \equiv C \wedge \bigwedge_{t \in T_m \setminus \{t^*\}} x_{t^*} \leq x_t$$

Example



$$C_{t_1} \equiv 0 \leq x_1 \leq 4 \wedge 5 \leq x_2 \leq 6 \wedge 3 \leq x_3 \leq 6 \wedge x_1 \leq x_2 \wedge x_1 \leq x_3$$

Firing a Transition from a Class

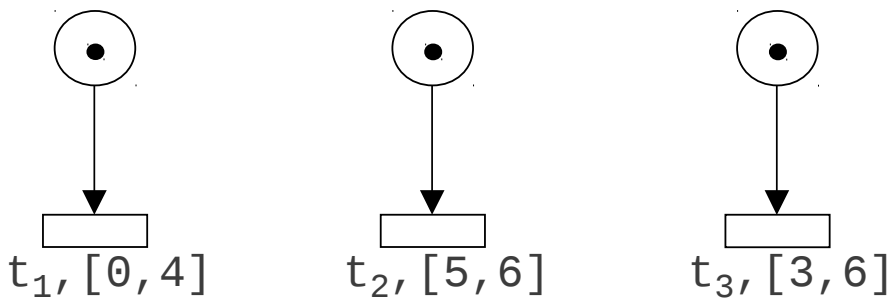
Let $m \xrightarrow{t^*} m'$ and $T_{m'}$ be the transitions enabled at m' with delays x'_t then:

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

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

$$C' \equiv \exists x_{t_1} \ldots \exists x_{t_k}\ C_{t^*} \wedge \bigwedge_{t \in T_{m'}} \tilde{C}_t \qquad \text{with } T_m = \{t_1, \ldots, t_k\}$$

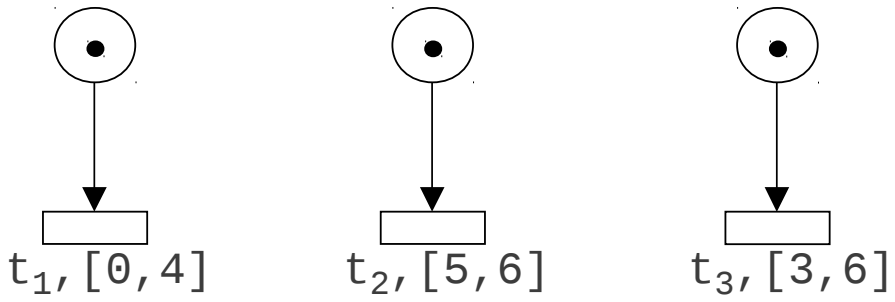
Example



$$C_{t_1} \equiv \exists x_1 \exists x_2 \exists x_3\ 0 \leq x_1 \leq 4 \wedge 5 \leq x_2 \leq 6 \wedge 3 \leq x_3 \leq 6$$
$$\wedge\ x_1 \leq x_2 \wedge x_1 \leq x_3$$
$$\wedge\ x'_2 = x_2 - x_1 \wedge x'_3 = x_3 - x_1$$

Problem: this is no more a DBM-based representation of a class!

Recovering the Class Representation



$$C_{t_1} \equiv \exists x_1 \exists x_2 \exists x_3 \; 0 \leq x_1 \leq 4 \wedge 5 \leq x_2 \leq 6 \wedge 3 \leq x_3 \leq 6$$
$$\wedge\, x_1 \leq x_2 \wedge x_1 \leq x_3$$
$$\wedge\, x_2' = x_2 - x_1 \wedge x_3' = x_3 - x_1$$

Recovery process

- Elimination of x_2 and x_3 by substitution

  $C_{t_1} \equiv \exists x_1 \; 0 \leq \; x_1 \leq 4 \wedge 5 \leq x_2' + x_1 \leq 6 \wedge 3 \leq x_3' + x_1 \leq 6$

- Elimination of x_1 by upper and lower bounds

  $C_{t_1} \equiv \exists x_1 \;\; \max(0, 5 - x_2', 3 - x_3') \leq x_1 \leq \min(4, 6 - x_2', 6 - x_3')$
  $\equiv 1 \leq x_2' \leq 6 \wedge 3 \leq x_3' \leq 6 \wedge -3 \leq x_3' - x_2' \leq 1$

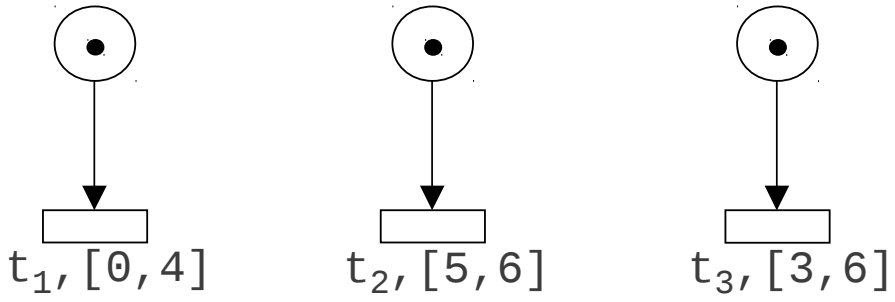
The Class Graph Algorithm

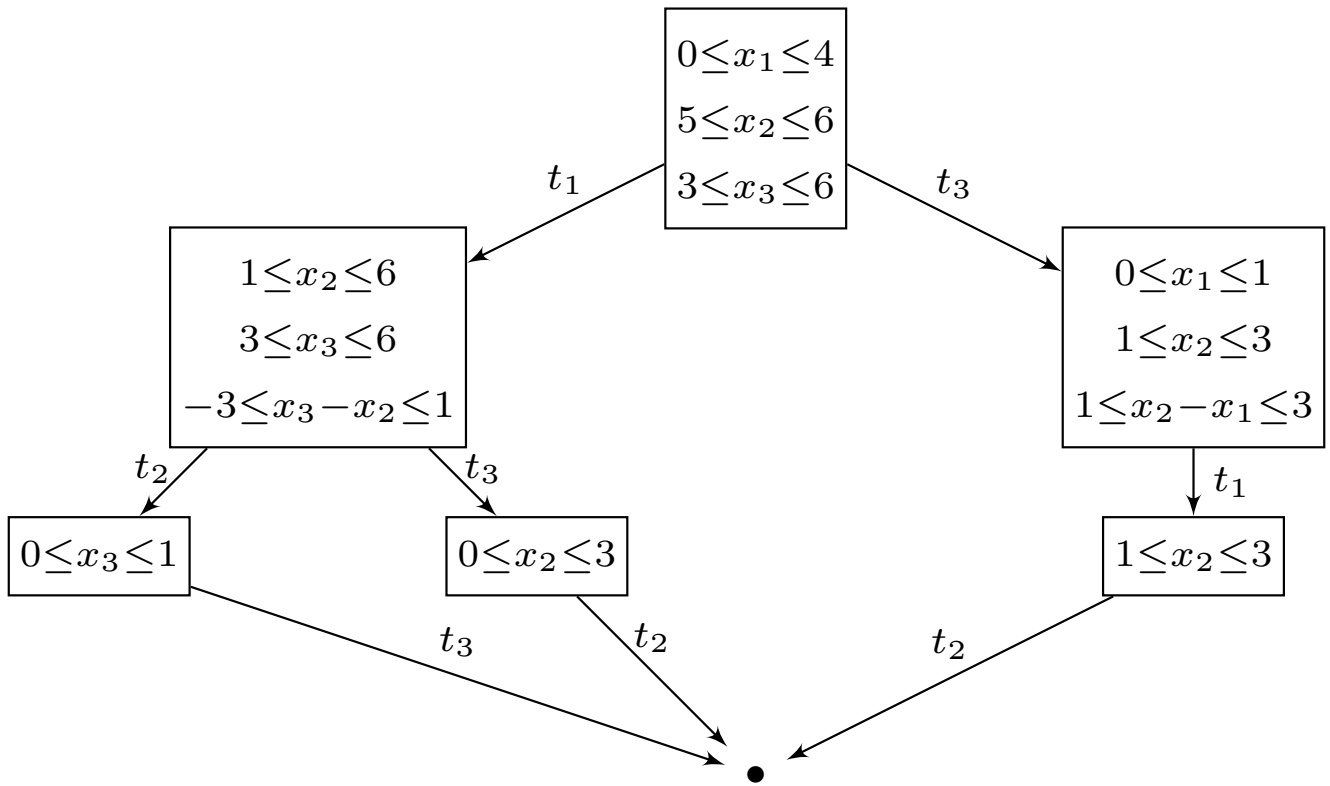
Add the initial class C to G *(the class graph)*
$Insert(Heap, C)$
While $Heap$ is not empty **do**
  $C \leftarrow Pick(Heap)$
  For all t firable from C **do**
    $C' \leftarrow Fire(C, t)$
    If C' does not belong the graph **then**
      Add C' to G
      $Insert(Heap, C')$
    Add $C \xrightarrow{t} C'$ to G

Open issues (solved below)

- How to check the emptiness of a DBM?
- How to check whether two DBMs admit the same set of solutions?

A Class Graph





Properties of the Class Graph

Finiteness for bounded nets

- The number of reachable markings is finite.
- The absolute value of integers occurring in the DBMs 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 graph.
- For every path of the graph there is a corresponding firing sequence.
- Thus the reachable markings are exactly those occurring in the graph.
- The reachable configurations are those of the classes occurring in the graph.

Outline

Time and Petri Nets

Time Petri Net: Syntax and Semantic

Analysis of Time Petri Nets

4 More on Difference Bound Matrices

Timed Petri Nets: Syntax and Semantics

Analysis of Timed Petri Nets

Properties of DBM

There exists a canonical representation for non empty DBM.

Canonization and emptyness checking can be done in polynomial time.

DBMs are effectively closed under:

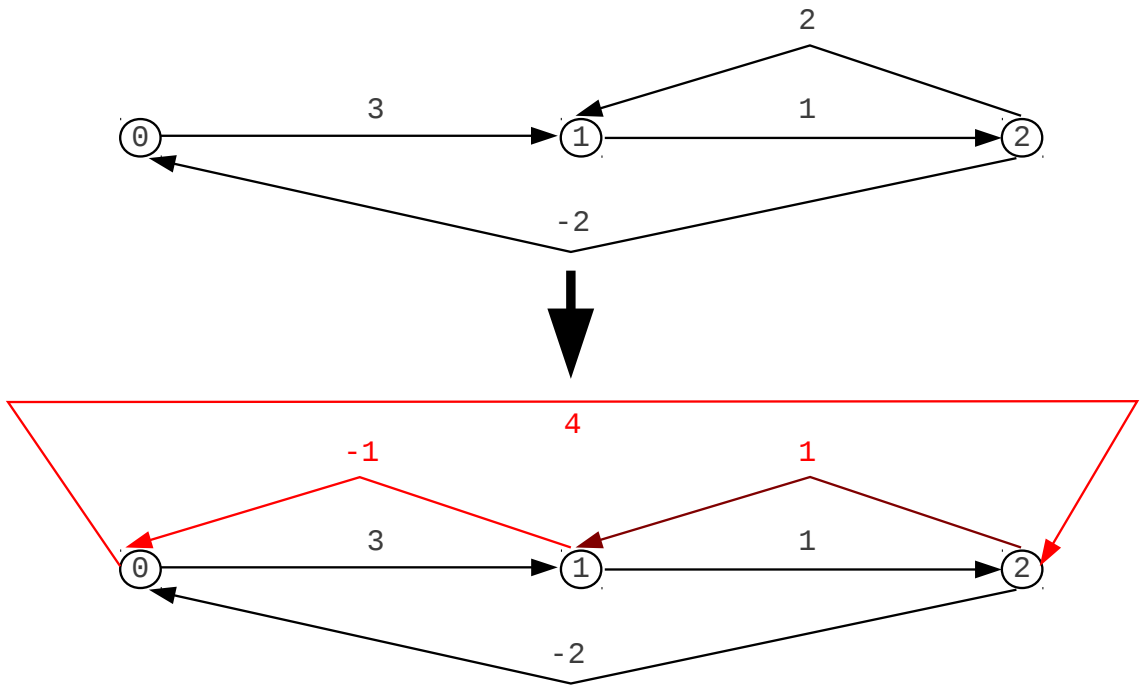
1. Projection $\exists x_1\ C(x_1, x_2, \ldots, x_n)$
2. Relativization $\exists x_1\ C(x_1, x_2 + x_1, \ldots, x_n + x_1)$
3. Past $\exists d\ C(x_1 + d, x_2 + d, \ldots, x_n + d)$
4. Future $\exists d\ C(x_1 - d, x_2 - d, \ldots, x_n - d)$
5. Reset $\exists x\ C(x, x_2, \ldots, x_n) \wedge x_1 = 0$

Canonization

Canonization is done by a shortest path computation.

Let the constraints 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:

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

The Canonization Algorithm

Floyd-Warshall Algorithm

For all k **do**

  For all j **do**

    For all i **do**

      $c_{ij} \leftarrow \min(c_{ij}, c_{ik} + c_{kj})$

For all i **do**

  If $c_{ii} < 0$ **Then** Return(Empty DBM)

Correctness of the Algorithm

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 $c_{ik} \leq c_{ij} + c_{jk}$ 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 $k \neq i$)*

Outline

Time and Petri Nets

Time Petri Net: Syntax and Semantic

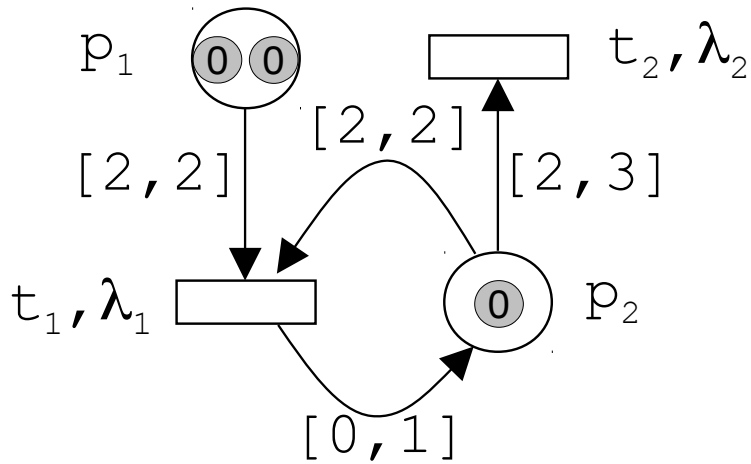
Analysis of Time Petri Nets

More on Difference Bound Matrices

5 Timed Petri Nets: Syntax and Semantics

Analysis of Timed Petri Nets

Timed Petri Net (TdPN): Syntax



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

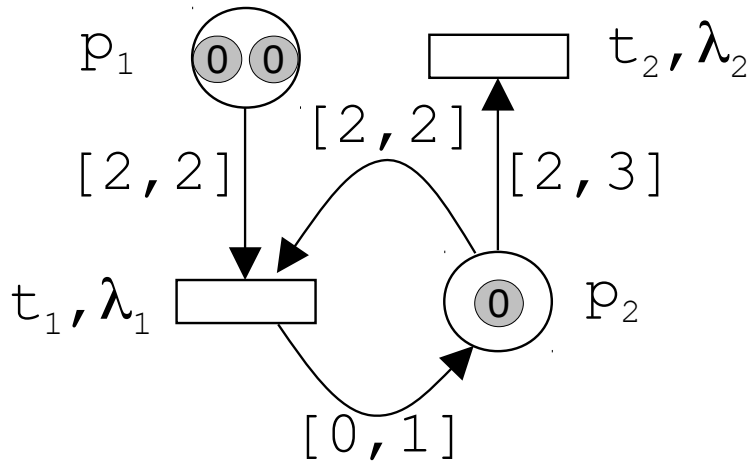
Transitions: events, actions, etc.
Labels: observable behaviour

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: Marking Formalization

A marking is a finite multiset of tokens with locations and ages:

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

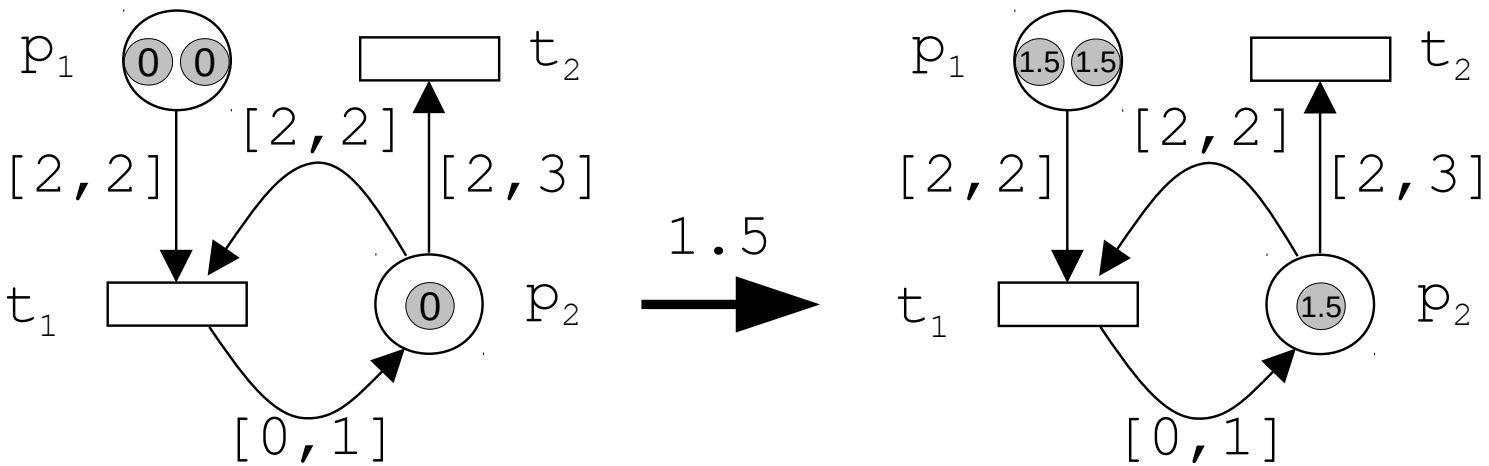


$$m_0 = 2.(p_1, 0) + (p_2, 0)$$

TdPN: Time Elapsing

Time elapsing d

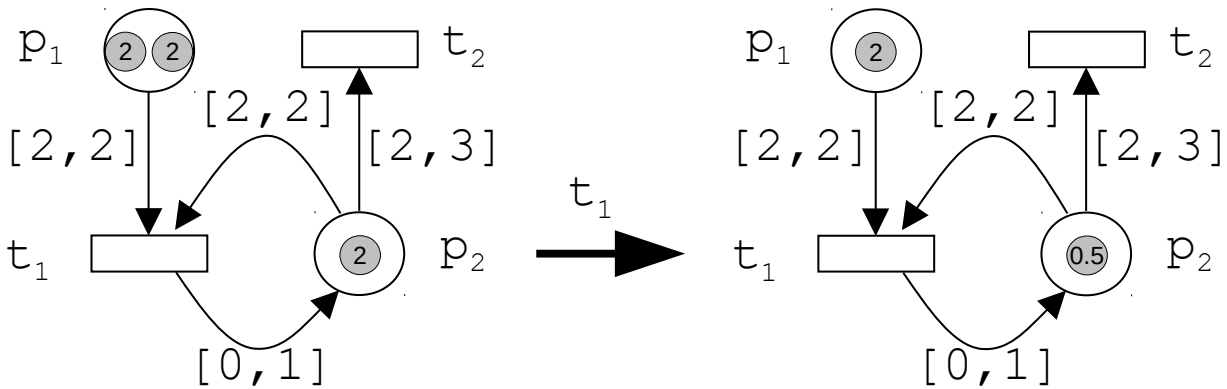
- Time may elapse without any restriction.
- The age of tokens is accordingly updated $m \xrightarrow{d} m'$ such that $m' = \sum_{1 \leq i \leq r} a_i.(p_i, \tau_i + d)$ when $m = \sum_{1 \leq i \leq r} a_i.(p_i, \tau_i)$



TdPN: Transition Occurrence

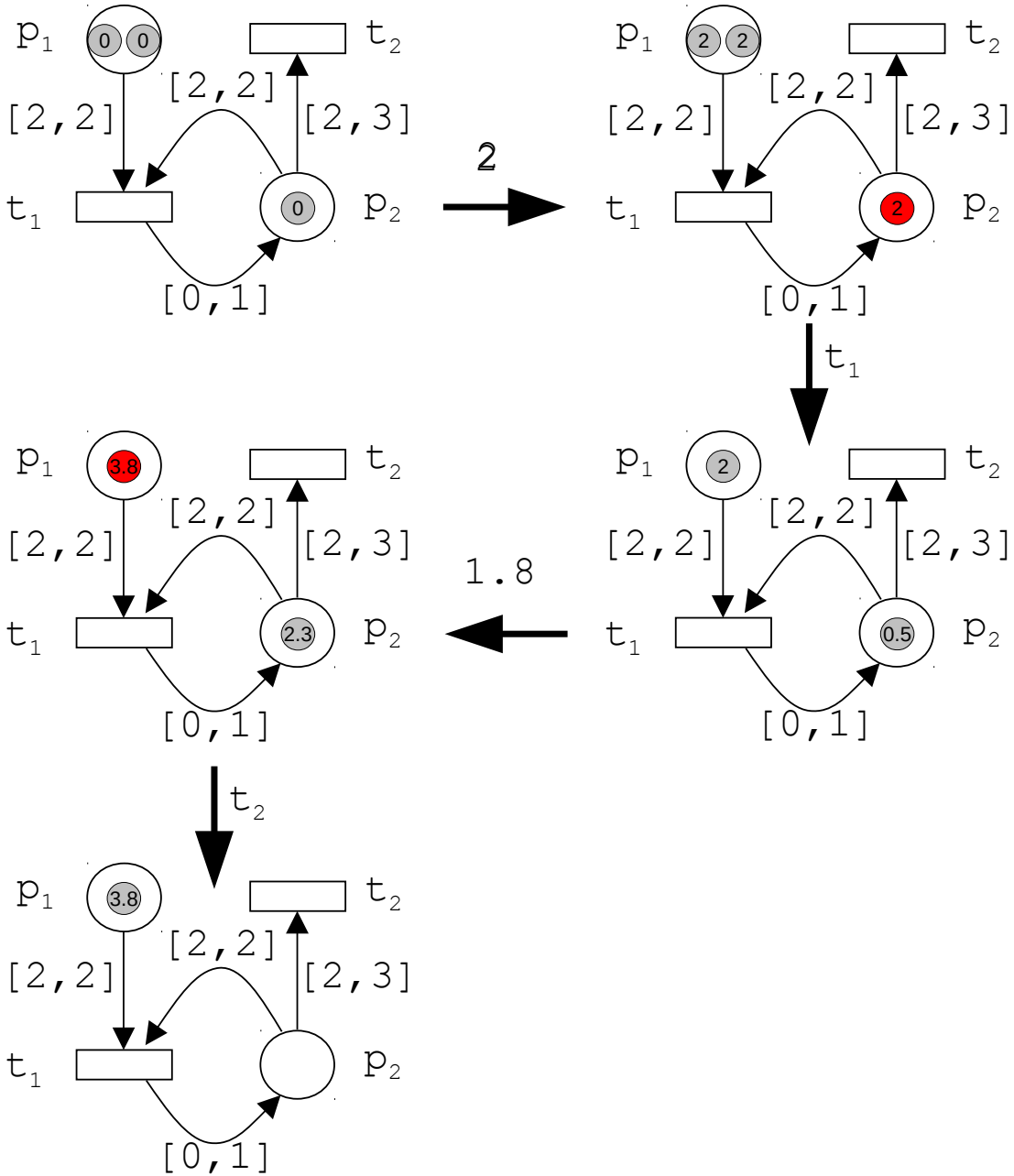
Firing a transition

- A transition t is firable if for every input place p of t there exists an appropriate token, i.e. some (p, τ_i) such that $\tau_i \in Pre(p, t)$.
- 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.



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

TdPN: an Execution



Outline

Time and Petri Nets

Time Petri Net: Syntax and Semantic

Analysis of Time Petri Nets

More on Difference Bound Matrices

Timed Petri Nets: Syntax and Semantics

6 Analysis of Timed Petri Nets

Reachability Analysis of Bounded TdPNs

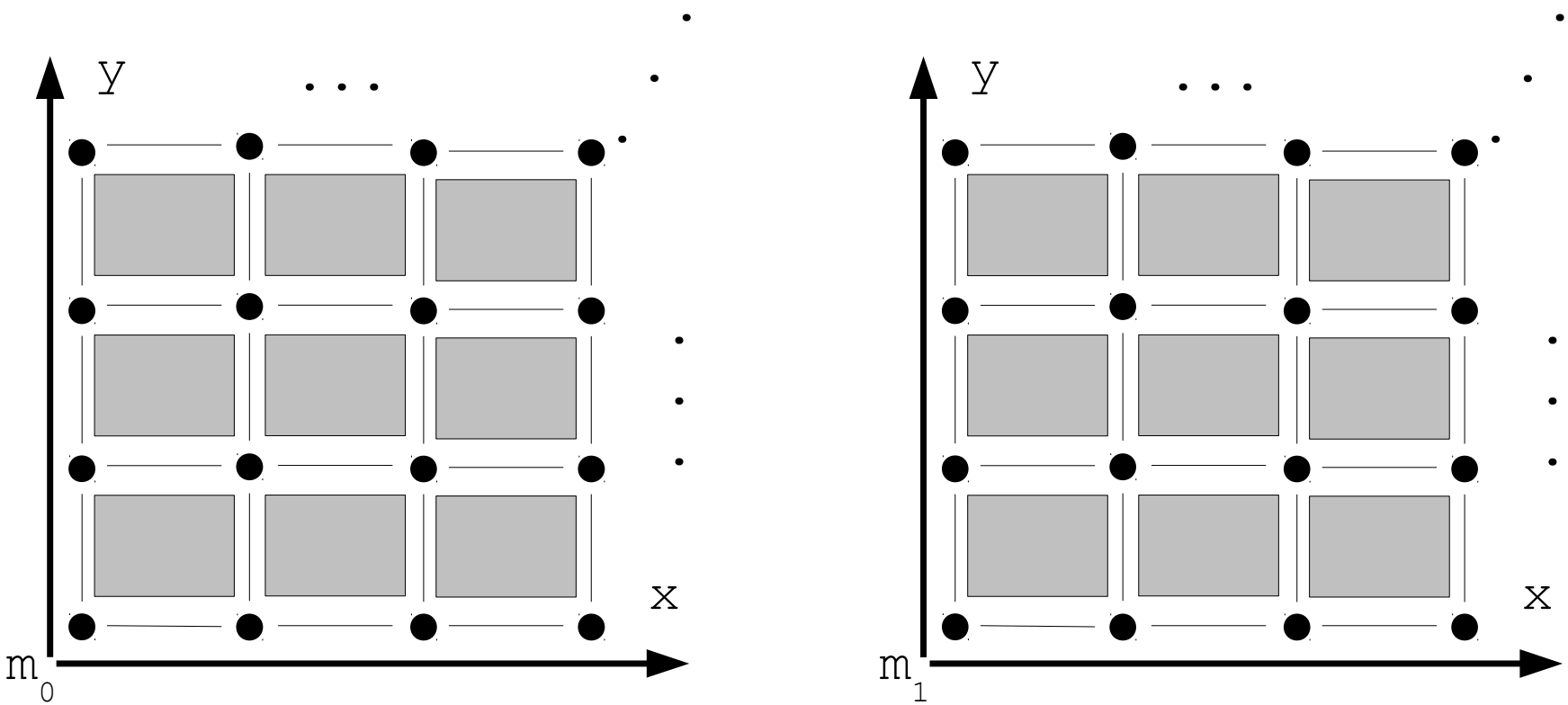
The number of (reachable) configurations is infinite (and even uncountable). So one wants to partition configurations into *regions* such that:

1. The number of tokens of any place for two configurations in a region is the same.
2. Two configurations in a region allow the same transition firings and the new configurations belong to the same region.
3. If a configuration in a region letting time elapse reaches a new region every other configuration may reach the same region by time elapsing.
4. There is a finite representation of a region such that the discrete and time successors of the region are computable.
5. The number of regions is finite.

A First Partition (two tokens and two markings)

Intervals check integer values.

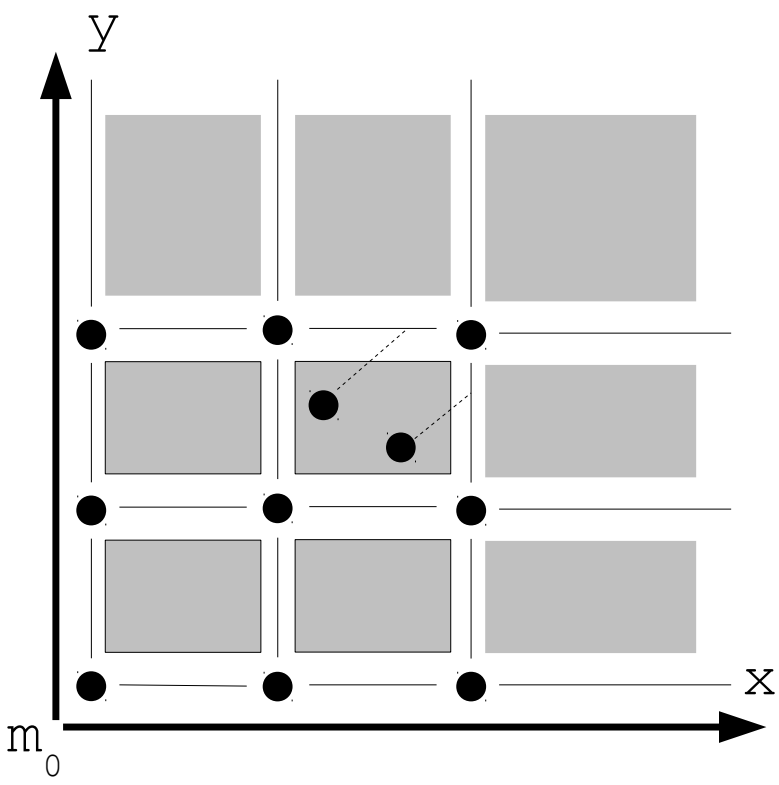
(x and y are the ages of the two tokens)

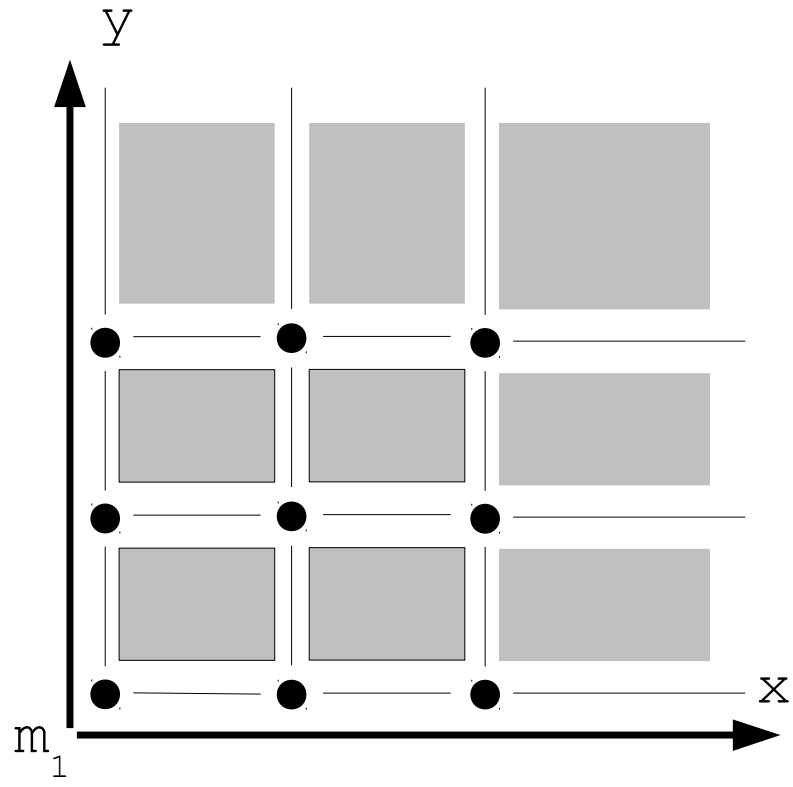


Why this partition is not appropriate?

A Second Partition (two tokens and two markings)

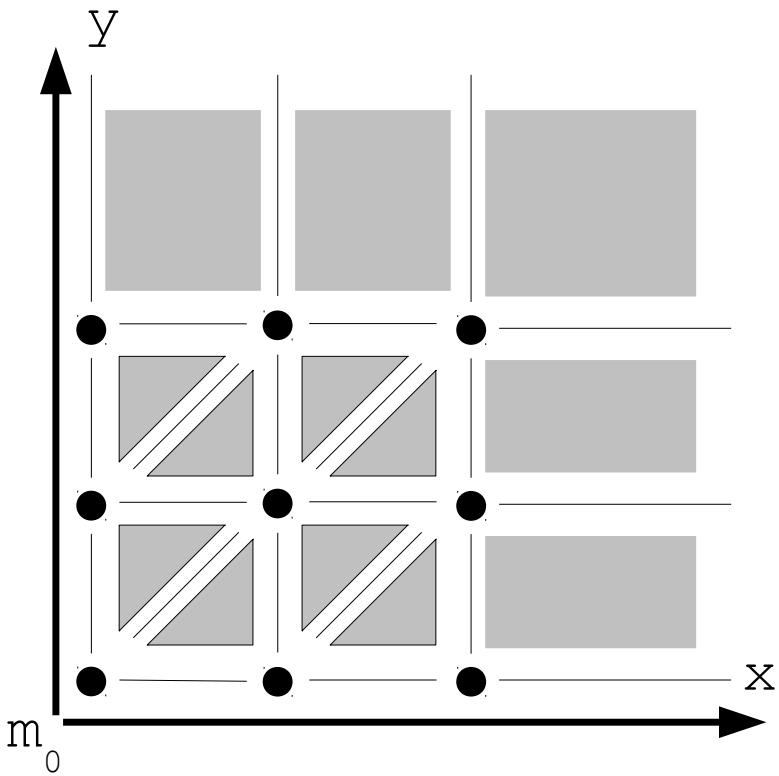
The exact value of a token age is irrelevant when it is beyond the maximal constant of the TdPN (here 2)

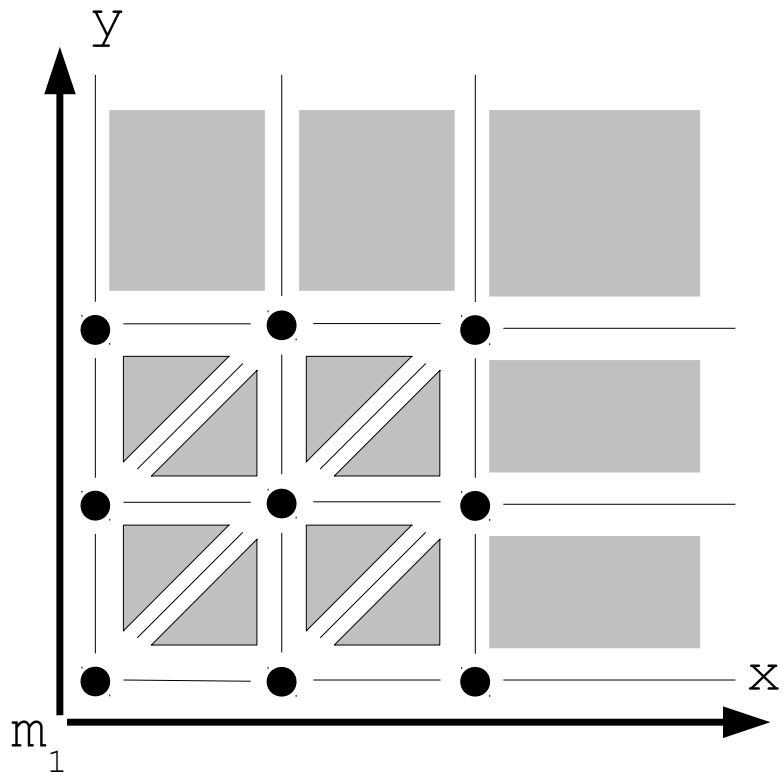




Why this partition is not appropriate?

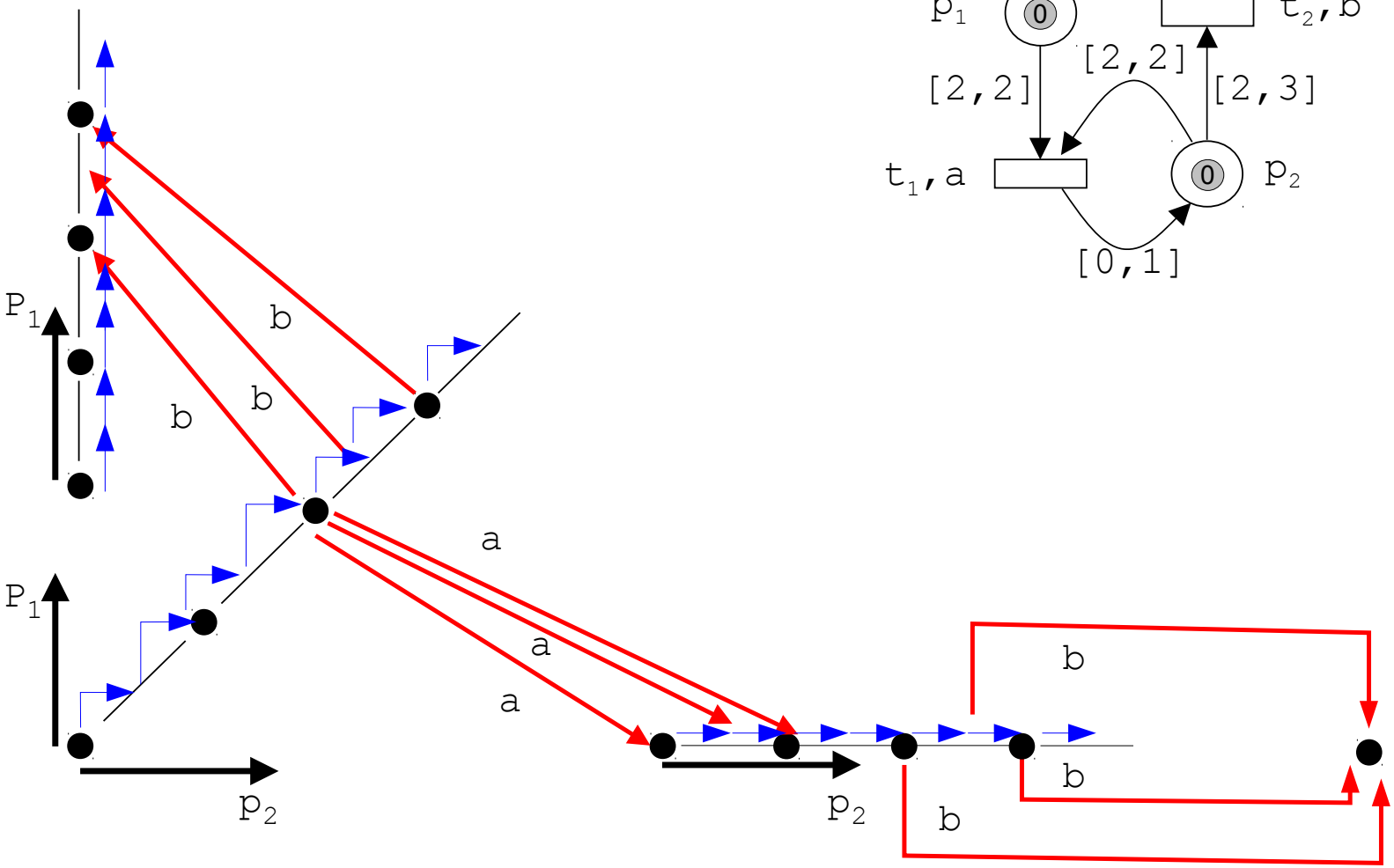
A Third Partition (two clocks and two locations)





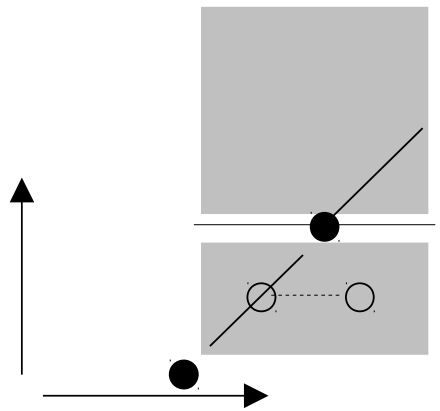
Check that this partition is appropriate

The Region Graph: Illustration



About Reachability in the Region Graph

Warning: when a region is "reachable", it does not mean that every configuration of the region is reachable.



However it means that there is another reachable configuration of the region which differs only on the values of irrelevant ages of tokens.

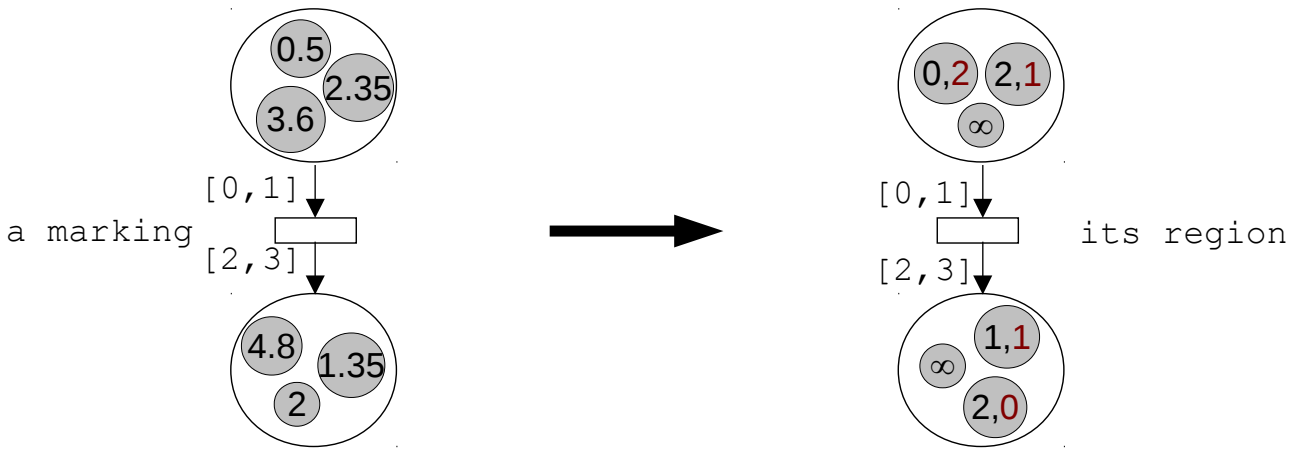
Hence, in order to check the reachability of a configuration, it is enough to increase the maximal constant.

Example: reachability of $((p_0, 1.7), (p_1, 2.3))$ requires to choose at least 3 as maximal constant.

Formalizing Regions

A *region of a TdPN* is:

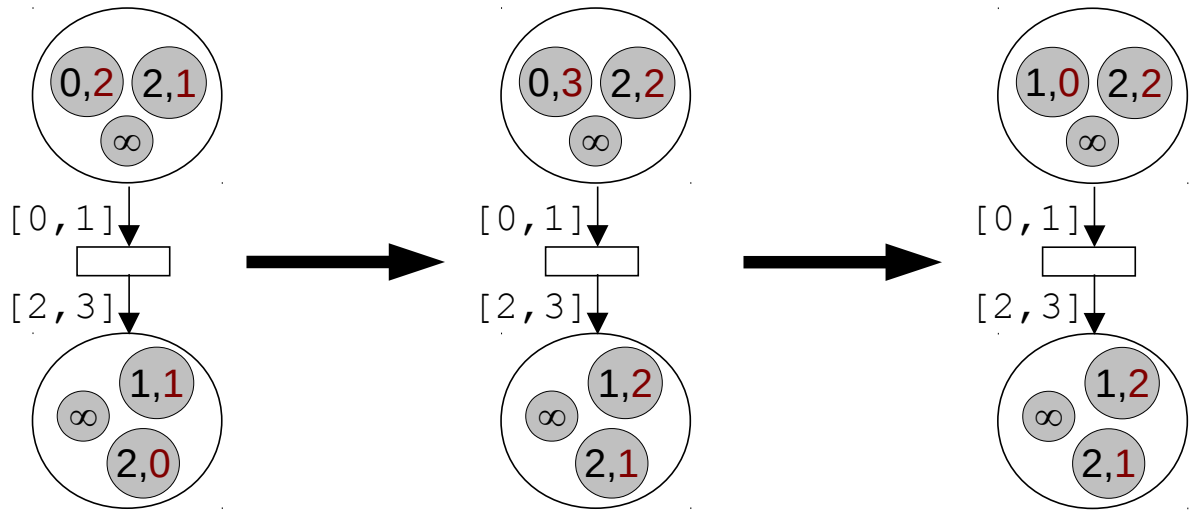
- $n+1$ different and ordered fractional parts of the token ages with the null fractional part.
- The distribution of tokens on places and their integer part (when less or equal than the maximal constant) for every fractional part.
- The distribution of tokens with age greater than the maximal constant on places.



In the unbounded case, the corresponding region graph is infinite but it is a *well-structured transition system* and thus (for instance) coverability can be decided by a symbolic backward exploration.

Time Elapsing for Regions

- When there is a token with integer age,
  the next region is obtained by letting elapse some amount of time
  such that there is no token with integer age.
- When there is no token with integer age and some tokens with finite age,
  the next region is obtained by letting elapse the (minimal) amount time
  to get a token with integer age.
- When there is no token with finite age, time elapses inside the region.



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.
- B. Bérard, F. Cassez, S. Haddad, D. Lime and O. H. Roux. The Expressive Power of Time Petri Nets. Theoretical Computer Science 474, pages 1-20, 2013.

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
- P. Bouyer, S. Haddad and P.-A. Reynier. Timed Petri Nets and Timed Automata: On the Discriminating Power of Zeno Sequences. Information and Computation 206(1), pages 73-107, 2008.