Petri Nets with time

Olivier (H.) Roux

Ecole Centrale de Nantes / Laboratoire des Sciences du Numérique de Nantes

2018-2019

Plan

- **Petri nets with time**
- **T-time Petri Nets**
- **Other Semantics**
  - P-time Petri Nets
  - A-time Petri Nets
  - Strong vs Weak Semantics
- **Expressiveness and properties of TPN**
- **State Space of Time Petri Nets**
- **Model-checking of Time Petri Nets**
  - Temporal logics
  - Checking TPN with observers
- **Stopwatch Petri Nets**

Introduction

Timed constraints are added to Petri nets

in different ways

- date (point) [Ram74] : Timed Petri nets
- interval [Mer74] : Time Petri nets

Introduction

Timed constraints are added to Petri nets

in different ways

- date (point) [Ram74] : Timed Petri nets
- interval [Mer74] : Time Petri nets

These constraints are associated with

- places : P-timed or P-time
- arcs (edge)
- transitions
- tokens...

Time Petri Nets

Several semantics

- strong semantics
- weak semantics

Time Petri Nets

Several semantics

- strong semantics
- weak semantics

The main models:

- T-time Petri Nets with strong semantics[Mer74, BD91]
- P-time Petri Nets with strong semantics [KDC96]
- A-time Petri Nets with weak semantics[Han93, AN01, dFRA00]

The most widely used model is : T-time Petri Nets with strong semantics called Time Petri Nets (TPN).

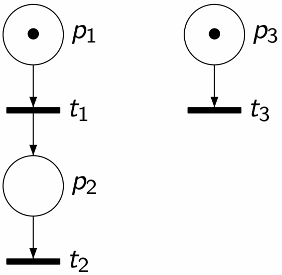
Plan

- Petri nets with time
- **T-time Petri Nets**
- Other Semantics
  - P-time Petri Nets
  - A-time Petri Nets
  - Strong vs Weak Semantics
- Expressiveness and properties of TPN
- State Space of Time Petri Nets
- Model-checking of Time Petri Nets
  - Temporal logics
  - Checking TPN with observers
- Stopwatch Petri Nets

T-time Petri Nets (*T-TPN*)

T-TPN: Time constraints are associated with transitions

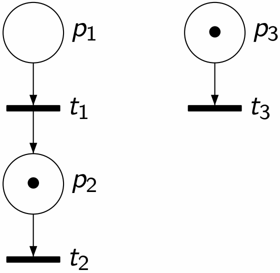
Example (Net of L. Gallon)



T-time Petri Nets (*T-TPN*)

T-TPN: Time constraints are associated with transitions

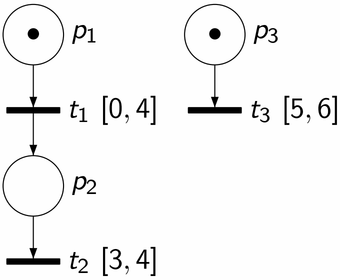
Example (Net of L. Gallon)



T-time Petri Nets (*T-TPN*)

T-TPN: Time constraints are associated with transitions

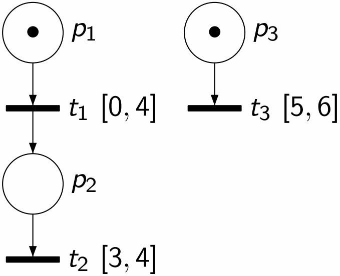
Example (Net of L. Gallon)



T-time Petri Nets (*T-TPN*)

T-TPN: Time constraints are associated with transitions

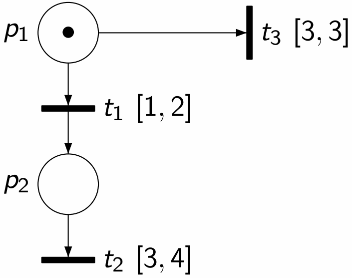
Example (Net of L. Gallon)



$$\begin{array}{l}\{p_1,p_3\}\\ \nu(t_1)=0\\ \nu(t_3)=0\end{array} \xrightarrow{\epsilon(4)} \begin{array}{l}\{p_1,p_3\}\\ \nu(t_1)=4\\ \nu(t_3)=4\end{array} \xrightarrow{t_1} \begin{array}{l}\{p_2,p_3\}\\ \nu(t_2)=0\\ \nu(t_3)=4\end{array} \xrightarrow{\epsilon(1)} \begin{array}{l}\{p_2,p_3\}\\ \nu(t_2)=1\\ \nu(t_3)=5\end{array} \xrightarrow{t_3} \cdots$$

Some examples

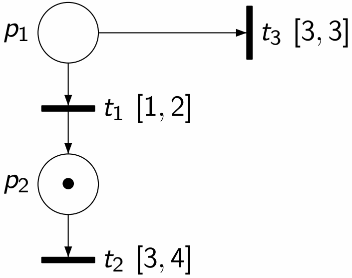
Example (Priority)



Is it possible to fire t3 ?

Some examples

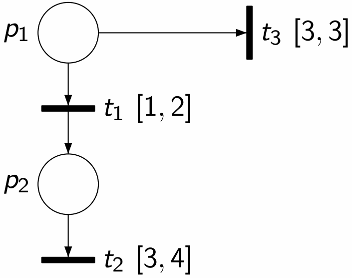
Example (Priority)



Is it possible to fire t3 ?

Some examples

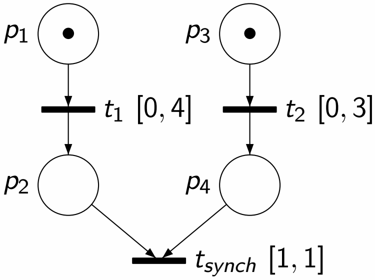
Example (Priority)



Is it possible to fire t3 ?

Some examples

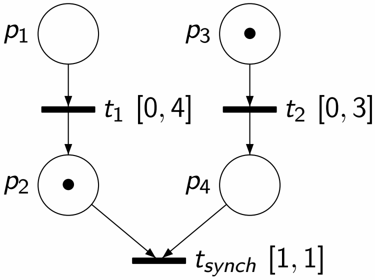
Example (Synchronization)



Is it possible to fire t_{synch} before t_1 or t_2 ?

Some examples

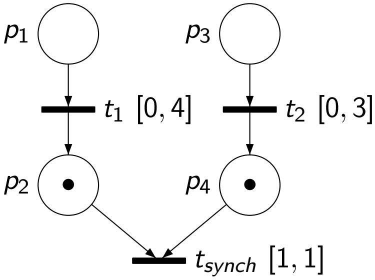
Example (Synchronization)



Is it possible to fire t_{synch} before t_1 or t_2 ?

Some examples

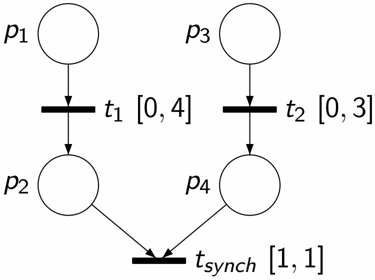
Example (Synchronization)



Is it possible to fire t_{synch} before t_1 or t_2 ?

Some examples

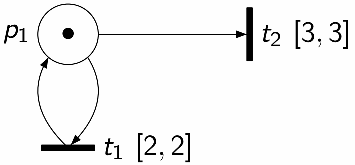
Example (Synchronization)



Is it possible to fire t_{synch} before t_1 or t_2 ?

Some examples

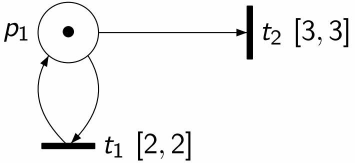
Example (Continuously enabled)



Is it possible to fire t2 ?

Some examples

Example (Continuously enabled)

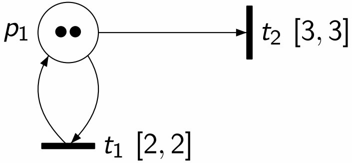


Is it possible to fire t2 ?

$$\begin{array}{l}\{p_1\}\\ \nu(t_1)=0\\ \nu(t_2)=0\end{array} \xrightarrow{\epsilon(2)} \begin{array}{l}\{p_1\}\\ \nu(t_1)=2\\ \nu(t_2)=2\end{array} \xrightarrow{t_1} \begin{array}{l}\{p_1\}\\ \nu(t_1)=0\\ \nu(t_2)=0\end{array} \xrightarrow{\epsilon(2)} \begin{array}{l}\{p_1\}\\ \nu(t_1)=2\\ \nu(t_2)=2\end{array} \xrightarrow{t_1} \cdots$$

Some examples

Example (Continuously enabled)



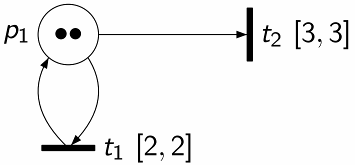
Is it possible to fire t2 ?

$$\begin{matrix}\{p_1\} \\ \nu(t_1)=0 \\ \nu(t_2)=0\end{matrix} \xrightarrow{\epsilon(2)} \begin{matrix}\{p_1\} \\ \nu(t_1)=2 \\ \nu(t_2)=2\end{matrix} \xrightarrow{t_1} \begin{matrix}\{p_1\} \\ \nu(t_1)=0 \\ \nu(t_2)=0\end{matrix} \xrightarrow{\epsilon(2)} \begin{matrix}\{p_1\} \\ \nu(t_1)=2 \\ \nu(t_2)=2\end{matrix} \xrightarrow{t_1} \cdots$$

And with 2 tokens in P_1 ?

Some examples

Example (Continuously enabled)



Is it possible to fire t2 ?

$$\begin{matrix}\{p_1\}\\ \nu(t_1)=0\\ \nu(t_2)=0\end{matrix} \xrightarrow{\epsilon(2)} \begin{matrix}\{p_1\}\\ \nu(t_1)=2\\ \nu(t_2)=2\end{matrix} \xrightarrow{t_1} \begin{matrix}\{p_1\}\\ \nu(t_1)=0\\ \nu(t_2)=0\end{matrix} \xrightarrow{\epsilon(2)} \begin{matrix}\{p_1\}\\ \nu(t_1)=2\\ \nu(t_2)=2\end{matrix} \xrightarrow{t_1} \cdots$$

And with 2 tokens in P_1 ?

$$\begin{matrix}\{p_1,p_1\}\\ \nu(t_1)=0\\ \nu(t_2)=0\end{matrix} \xrightarrow{\epsilon(2)} \begin{matrix}\{p_1,p_1\}\\ \nu(t_1)=2\\ \nu(t_2)=2\end{matrix} \xrightarrow{t_1} \begin{matrix}\{p_1,p_1\}\\ \nu(t_1)=0\\ \nu(t_2)=2\end{matrix} \xrightarrow{\epsilon(1)} \begin{matrix}\{p_1,p_1\}\\ \nu(t_1)=1\\ \nu(t_2)=3\end{matrix} \xrightarrow{t_2} \begin{matrix}\{p_1\}\\ \nu(t_1)=1\\ \nu(t_2)=0\end{matrix} \xrightarrow{\epsilon(1)}\xrightarrow{t_1} \cdots$$

A TPN : $\mathcal{N} = (\mathcal{P}, \mathcal{T}, {}^{\bullet}(.), (.)^{\bullet}, \mathcal{M}_{\prime}, \mathcal{I})$

Newly Enabled Transition

A TPN : $\mathcal{N} = (\mathcal{P}, \mathcal{T}, {}^{\bullet}(.), (.)^{\bullet}, \mathcal{M}_{\prime}, \mathcal{I})$

Newly Enabled Transition

t' is Newly enabled by the firing of t from M :

$$\begin{aligned} \uparrow enabled(t', M, t) = \big(M - {}^{\bullet}t + t^{\bullet} \geq {}^{\bullet}t'\big) \wedge \\ \big((M - {}^{\bullet}t < {}^{\bullet}t') \vee (t' = t)\big) \end{aligned} \tag{1}$$

Definition and Semantics of *T-TPN*

Definition

A *Time Petri Net* $\mathcal{N}$ is a tuple $(P, T, ^\bullet(.), (.)^\bullet, M_0, I)$
Remark : Often $I = [\alpha, \beta]$

Definition and Semantics of *T-TPN*

Definition

A *Time Petri Net* $\mathcal{N}$ is a tuple $(P, T, ^\bullet(.), (.)^\bullet, M_0, I)$
Remark : Often $I = [\alpha, \beta]$

Definition

A State of a *T-TPN* is $S = (M, \nu)$ with

- M : a marking and
- ν: a valuation *enabled* $(M) \mapsto \mathbb{R}_{\geq 0}$

Definition and Semantics of *T-TPN*

Definition

A *Time Petri Net* $\mathcal{N}$ is a tuple $(P, T, {}^\bullet(.), (.)^\bullet, M_0, I)$
Remark : Often $I = [\alpha, \beta]$

Definition

A State of a *T-TPN* is $S = (M, \nu)$ with

- M : a marking and
- ν: a valuation *enabled* $(M) \mapsto \mathbb{R}_{\geq 0}$

Definition

- The semantics of a TPN $\mathcal{N}$ is a Timed Transition System $S_{\mathcal{N}} = (Q, \{q_0\}, T, \rightarrow)$

Definition (Semantics of Time Petri Nets)

- States: (M, ν) with M : a marking and $\nu : enabled\,(M) \mapsto \mathbb{R}_{\geq 0}$

Definition (Semantics of Time Petri Nets)

- States: (M, ν) with M : a marking and ν : *enabled* $(M) \mapsto \mathbb{R}_{\geq 0}$
- Discrete Transition: $(M, \nu) \xrightarrow{t} (M', \nu')$ ssi

$$\begin{cases} t \text{ is enabled by } M \text{ and } M' = M - {}^{\bullet}t + t^{\bullet} \\ \nu(t) \text{ is in } I \text{ (the interval associated with } t\text{)} \\ \nu'(t') = 0 \text{ if } t' \text{ is enabled by the firing of } t,\ \nu'(t') = \nu(t') \text{ otherwise} \end{cases}$$

Definition (Semantics of Time Petri Nets)

- States: (M, ν) with M : a marking and $\nu : enabled(M) \mapsto \mathbb{R}_{\geq 0}$
- Discrete Transition: $(M, \nu) \xrightarrow{t} (M', \nu')$ ssi
$$\begin{cases} t \textit{ is enabled by } M \textit{ and } M' = M - {}^{\bullet}t + t^{\bullet} \\ \nu(t) \textit{ is in } I \textit{ (the interval associated with } t) \\ \nu'(t') = 0 \textit{ if } t' \textit{ is enabled by the firing of } t,\ \nu'(t') = \nu(t') \textit{ otherwise} \end{cases}$$
- Timed Transition: $(M, \nu) \xrightarrow{d} (M', \nu')$ iff
$$\begin{cases} M = M' \textit{ and } \nu' = \nu + d \textit{ (updates of enabled transitions)} \\ \textit{For all enabled transitions } t,\ \forall d' \leq d,\ \nu(t) + d' \in I(t)^{\downarrow} \end{cases}$$

Definition (Semantics of Time Petri Nets)

- States: (M, ν) with M : a marking and $\nu : enabled(M) \mapsto \mathbb{R}_{\geq 0}$
- Discrete Transition: $(M, \nu) \xrightarrow{t} (M', \nu')$ ssi
$$\begin{cases} t \textit{ is enabled by } M \textit{ and } M' = M - {}^{\bullet}t + t^{\bullet} \\ \nu(t) \textit{ is in } I \textit{ (the interval associated with } t\textit{)} \\ \nu'(t') = 0 \textit{ if } t' \textit{ is enabled by the firing of } t\textit{, } \nu'(t') = \nu(t') \textit{ otherwise} \end{cases}$$
- Timed Transition: $(M, \nu) \xrightarrow{d} (M', \nu')$ iff
$$\begin{cases} M = M' \textit{ and } \nu' = \nu + d \textit{ (updates of enabled transitions)} \\ \textit{For all enabled transitions } t\textit{, } \forall d' \leq d\textit{, } \nu(t) + d' \in I(t)^{\downarrow} \end{cases}$$
- Un TPN generate a set of runs = alternation of discrete and continuous steps

Definition (Semantics of Time Petri Nets)

- States: (M, ν) with M : a marking and $\nu : enabled(M) \mapsto \mathbb{R}_{\geq 0}$
- Discrete Transition: $(M, \nu) \xrightarrow{t} (M', \nu')$ ssi
$$\begin{cases} t \text{ is enabled by } M \text{ and } M' = M - {}^\bullet t + t^\bullet \\ \nu(t) \text{ is in } I \text{ (the interval associated with } t) \\ \nu'(t') = 0 \text{ if } t' \text{ is enabled by the firing of } t,\ \nu'(t') = \nu(t') \text{ otherwise} \end{cases}$$
- Timed Transition: $(M, \nu) \xrightarrow{d} (M', \nu')$ iff
$$\begin{cases} M = M' \text{ and } \nu' = \nu + d \text{ (updates of enabled transitions)} \\ \text{For all enabled transitions } t,\ \forall d' \leq d,\ \nu(t) + d' \in I(t)^{\downarrow} \end{cases}$$
- Un TPN generate a set of runs = alternation of discrete and continuous steps
- The semantics of a TPN N = Timed Transition System S_N

Formally

A *Time Petri Net* $\mathcal{N}$ is a tuple $(P, T, {}^\bullet(.), (.)^\bullet, M_0, I)$ where:

- $P = \{p_1, p_2, \cdots, p_m\}$ is a finite set of places
- $T = \{t_1, t_2, \cdots, t_n\}$ is a finite set of transitions and $P \cap T = \emptyset$;
- ${}^\bullet(.) \in (\mathbb{N}^P)^T$ is the backward incidence mapping; $(.)^\bullet \in (\mathbb{N}^P)^T$ is the forward incidence mapping;
- $M_0 \in \mathbb{N}^P$ is the initial marking;
- $I : T \to \mathcal{I}(\mathbb{Q}_{\geq 0})$ associates with each transition a firing interval;

Semantics of Time Petri Nets $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, I)$

A state $Q = (M, \nu)$ of $\mathcal{N}$ is a pair with $M \in \mathbb{N}^P$ and $\nu \in \mathbb{R}_{\geq 0}^{enabled(M)}$.

The semantics of $\mathcal{N}$ is a Timed Transition System $S_{\mathcal{N}} = (Q, \{q_0\}, T, \rightarrow)$:

- $q_0 = (M_0, \mathbf{0})$,
- $\longrightarrow \in Q \times (T \cup \mathbb{R}_{\geq 0}) \times Q$ is the transition relations:

Semantics of Time Petri Nets $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, M_0, I)$

A state $Q = (M, \nu)$ of $\mathcal{N}$ is a pair with $M \in \mathbb{N}^P$ and $\nu \in \mathbb{R}_{\geq 0}^{enabled(M)}$.

The semantics of $\mathcal{N}$ is a Timed Transition System $S_{\mathcal{N}} = (Q, \{q_0\}, T, \rightarrow)$:

- $q_0 = (M_0, \mathbf{0})$,
- $\longrightarrow \in Q \times (T \cup \mathbb{R}_{\geq 0}) \times Q$ is the transition relations:
  - the discrete transition relation is defined $\forall t \in T$ by:

$$(M,\nu) \xrightarrow{t} (M',\nu') \textit{ iff } \begin{cases} t \in enabled\,(M) \wedge M' = M - {}^\bullet t + t^\bullet \\ \nu(t) \in I(t), \\ \forall t' \in enabled\,(M')\,, \nu'(t') = \begin{cases} 0 \textit{ if } \uparrow\!enabled(t', M, t), \\ \nu(t') \textit{ otherwise.} \end{cases} \end{cases}$$

Semantics of Time Petri Nets $\mathcal{N} = (P, T, {}^{\bullet}(.), (.)^{\bullet}, M_0, I)$

A state $Q = (M, \nu)$ of $\mathcal{N}$ is a pair with $M \in \mathbb{N}^P$ and $\nu \in \mathbb{R}_{\geq 0}^{enabled(M)}$.

The semantics of $\mathcal{N}$ is a Timed Transition System $S_{\mathcal{N}} = (Q, \{q_0\}, T, \rightarrow)$:

- $q_0 = (M_0, \mathbf{0})$,
- $\longrightarrow \in Q \times (T \cup \mathbb{R}_{\geq 0}) \times Q$ is the transition relations:
  - the discrete transition relation is defined $\forall t \in T$ by:

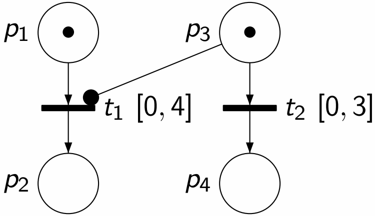
$$(M, \nu) \xrightarrow{t} (M', \nu') \ iff \begin{cases} t \in enabled\,(M) \wedge M' = M - {}^{\bullet}t + t^{\bullet} \\ \nu(t) \in I(t), \\ \forall t' \in enabled\,(M')\,, \nu'(t') = \begin{cases} 0 \ if \ \uparrow\!enabled(t', M, t), \\ \nu(t') \ otherwise. \end{cases} \end{cases}$$

  - continuous transition relation is defined $\forall d \in \mathbb{R}_{\geq 0}$ by:

$$(M, \nu) \xrightarrow{d} (M, \nu') \ iff \begin{cases} \nu' = \nu + d \\ \forall t \in enabled\,(M)\,, \nu'(t) \in I(t)^{\downarrow} \end{cases}$$

Some particular arcs

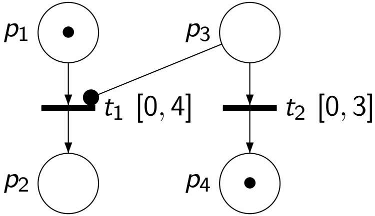
Example (Logical inhibitor arc)



Is it possible to fire t_1 before t_2 ?

Some particular arcs

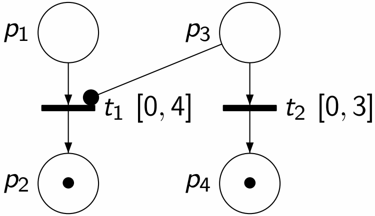
Example (Logical inhibitor arc)



Is it possible to fire t_1 before t_2 ?

Some particular arcs

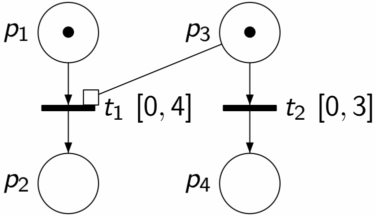
Example (Logical inhibitor arc)



Is it possible to fire t_1 before t_2 ?

Some particular arcs

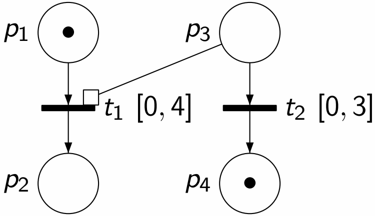
Example (Read arc)



Is it possible to fire t_2 before t_1 ?

Some particular arcs

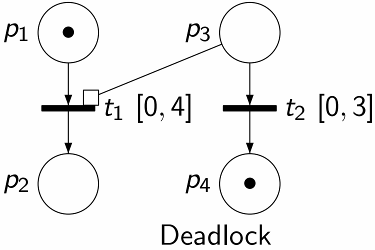
Example (Read arc)



Is it possible to fire t_2 before t_1 ?

Some particular arcs

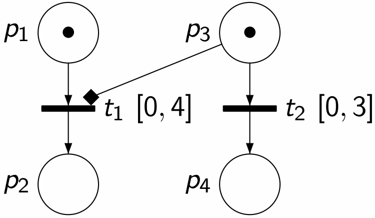
Example (Read arc)



Is it possible to fire t_2 before t_1 ?

Some particular arcs

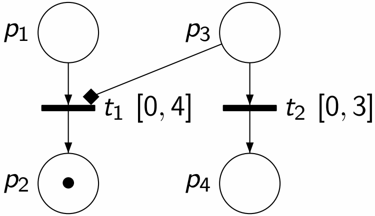
Example (Reset arc)



Is it possible to fire t_1 before t_2 ?

Some particular arcs

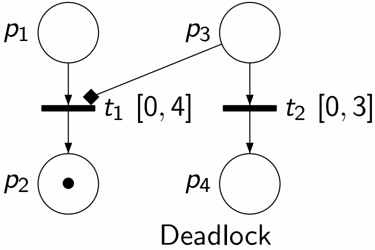
Example (Reset arc)



Is it possible to fire t_1 before t_2 ?

Some particular arcs

Example (Reset arc)



Is it possible to fire t_1 before t_2 ?

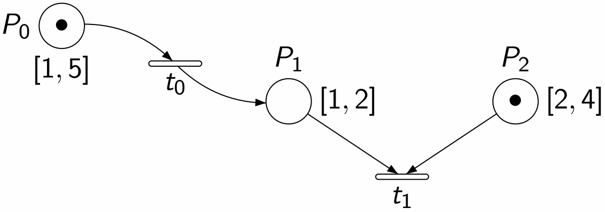
Plan

- Petri nets with time
- T-time Petri Nets
- **Other Semantics**
  - P-time Petri Nets
  - A-time Petri Nets
  - Strong vs Weak Semantics
- Expressiveness and properties of TPN
- State Space of Time Petri Nets
- Model-checking of Time Petri Nets
  - Temporal logics
  - Checking TPN with observers
- Stopwatch Petri Nets

P-time Petri Nets (*P-TPN*)

P-TPN: Time constraints are associated with places

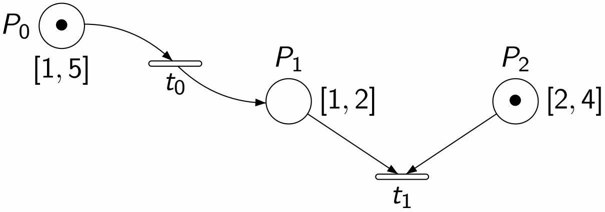
Example (A TPN $\in$ *P-TPN*)



P-time Petri Nets (*P-TPN*)

P-TPN: Time constraints are associated with places

Example (A TPN $\in$ *P-TPN*)

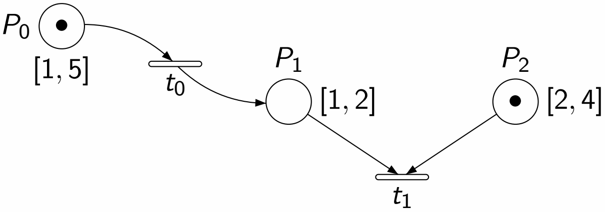


$$\begin{array}{c} \{p_0, p_2\} \\ \nu(p_0) = 0 \\ \nu(p_2) = 0 \end{array} \xrightarrow{\epsilon(3)} \begin{array}{c} \{p_0, p_2\} \\ \nu(p_0) = 3 \\ \nu(p_2) = 3 \end{array} \xrightarrow{t_0} \begin{array}{c} \{p_1, p_2\} \\ \nu(p_1) = 0 \\ \nu(p_2) = 3 \end{array} \xrightarrow{\epsilon(1)} \begin{array}{c} \{p_1, p_2\} \\ \nu(p_1) = 1 \\ \nu(p_2) = 4 \end{array} \xrightarrow{t_1}$$

P-time Petri Nets (*P-TPN*)

P-TPN: Time constraints are associated with places

Example (A TPN $\in$ *P-TPN*)



$$\begin{array}{c} \{p_0,p_2\} \\ \nu(p_0)=0 \\ \nu(p_2)=0 \end{array} \xrightarrow{\epsilon(3)} \begin{array}{c} \{p_0,p_2\} \\ \nu(p_0)=3 \\ \nu(p_2)=3 \end{array} \xrightarrow{t_0} \begin{array}{c} \{p_1,p_2\} \\ \nu(p_1)=0 \\ \nu(p_2)=3 \end{array} \xrightarrow{\epsilon(1)} \begin{array}{c} \{p_1,p_2\} \\ \nu(p_1)=1 \\ \nu(p_2)=4 \end{array} \xrightarrow{t_1}$$

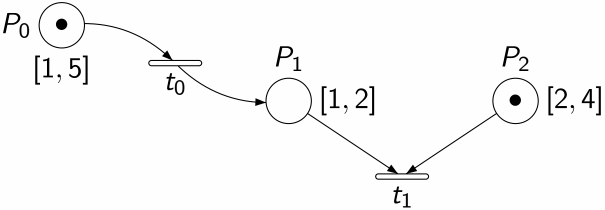
Weak semantics :

$$\begin{array}{c} \{p_0,p_2\} \\ \nu(p_0)=0 \\ \nu(p_2)=0 \end{array} \xrightarrow{\epsilon(3)} \begin{array}{c} \{p_0,p_2\} \\ \nu(p_0)=3 \\ \nu(p_2)=3 \end{array} \xrightarrow{t_0} \begin{array}{c} \{p_1,p_2\} \\ \nu(p_1)=0 \\ \nu(p_2)=3 \end{array} \xrightarrow{\epsilon(2)} \begin{array}{c} \{p_1,\hat{p}_2\} \\ \nu(p_1)=2 \\ \nu(p_2)=5 \end{array} \xrightarrow{\epsilon(\cdots)}$$

P-time Petri Nets (*P-TPN*)

P-TPN: Time constraints are associated with places

Example (A TPN $\in$ *P-TPN*)



$$\begin{array}{l}\{p_0,p_2\}\\ \nu(p_0)=0\\ \nu(p_2)=0\end{array} \xrightarrow{\epsilon(3)} \begin{array}{l}\{p_0,p_2\}\\ \nu(p_0)=3\\ \nu(p_2)=3\end{array} \xrightarrow{t_0} \begin{array}{l}\{p_1,p_2\}\\ \nu(p_1)=0\\ \nu(p_2)=3\end{array} \xrightarrow{\epsilon(1)} \begin{array}{l}\{p_1,p_2\}\\ \nu(p_1)=1\\ \nu(p_2)=4\end{array} \xrightarrow{t_1}$$

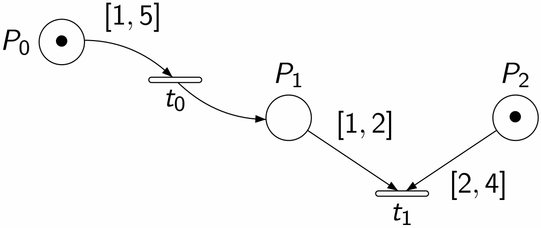
Strong semantics :

$$\begin{array}{l}\{p_0,p_2\}\\ \nu(p_0)=0\\ \nu(p_2)=0\end{array} \xrightarrow{\epsilon(5)} \begin{array}{l}\{p_0,\hat{p}_2\}\\ \nu(p_0)=5\\ \nu(p_2)=5\end{array} \xrightarrow{t_0} \begin{array}{l}\{p_1,\hat{p}_2\}\\ \nu(p_1)=0\\ \nu(p_2)=5\end{array} \xrightarrow{\epsilon(\cdots)}$$

A-time Petri Nets (*A-TPN*)

Time constraints are associated with arcs (Place,transition)

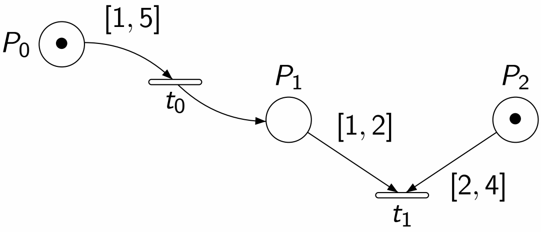
Example (Un TPN $\in$ *A-TPN*)



A-time Petri Nets (*A-TPN*)

Time constraints are associated with arcs (Place,transition)

Example (Un TPN $\in$ *A-TPN*)

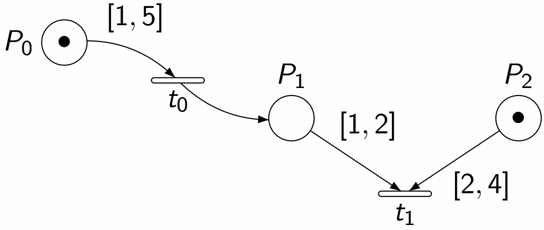


$$\begin{array}{l}\{p_0,p_2\}\\ \nu(p_0)=0\\ \nu(p_2)=0\end{array} \xrightarrow{\epsilon(3)} \begin{array}{l}\{p_0,p_2\}\\ \nu(p_0)=3\\ \nu(p_2)=3\end{array} \xrightarrow{t_0} \begin{array}{l}\{p_1,p_2\}\\ \nu(p_1)=0\\ \nu(p_2)=3\end{array} \xrightarrow{\epsilon(1)} \begin{array}{l}\{p_1,p_2\}\\ \nu(p_1)=1\\ \nu(p_2)=4\end{array} \xrightarrow{t_1}$$

A-time Petri Nets (*A-TPN*)

Time constraints are associated with arcs (Place,transition)

Example (Un TPN $\in$ *A-TPN*)



$$\begin{array}{c} \{p_0, p_2\} \\ \nu(p_0) = 0 \\ \nu(p_2) = 0 \end{array} \xrightarrow{\epsilon(3)} \begin{array}{c} \{p_0, p_2\} \\ \nu(p_0) = 3 \\ \nu(p_2) = 3 \end{array} \xrightarrow{t_0} \begin{array}{c} \{p_1, p_2\} \\ \nu(p_1) = 0 \\ \nu(p_2) = 3 \end{array} \xrightarrow{\epsilon(1)} \begin{array}{c} \{p_1, p_2\} \\ \nu(p_1) = 1 \\ \nu(p_2) = 4 \end{array} \xrightarrow{t_1}$$

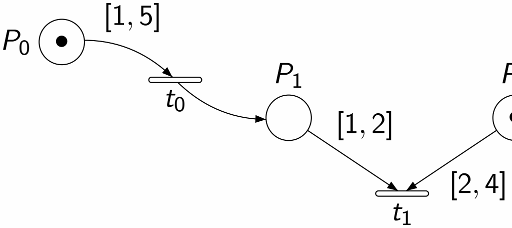
Weak semantics :

$$\begin{array}{c} \{p_0, p_2\} \\ \nu(p_0) = 0 \\ \nu(p_2) = 0 \end{array} \xrightarrow{\epsilon(3)} \begin{array}{c} \{p_0, p_2\} \\ \nu(p_0) = 3 \\ \nu(p_2) = 3 \end{array} \xrightarrow{t_0} \begin{array}{c} \{p_1, p_2\} \\ \nu(p_1) = 0 \\ \nu(p_2) = 3 \end{array} \xrightarrow{\epsilon(2)} \begin{array}{c} \{p_1, \hat{p}_2\} \\ \nu(p_1) = 2 \\ \nu(p_2) = 5 \end{array} \xrightarrow{\epsilon(\cdots)}$$

A-time Petri Nets (*A-TPN*)

Time constraints are associated with arcs (Place,transition)

Example (Un TPN $\in$ *A-TPN*)



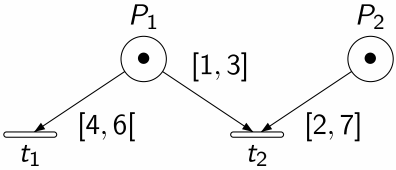
$$\begin{matrix}\{p_0,p_2\}\\ \nu(p_0)=0\\ \nu(p_2)=0\end{matrix} \xrightarrow{\epsilon(3)} \begin{matrix}\{p_0,p_2\}\\ \nu(p_0)=3\\ \nu(p_2)=3\end{matrix} \xrightarrow{t_0} \begin{matrix}\{p_1,p_2\}\\ \nu(p_1)=0\\ \nu(p_2)=3\end{matrix} \xrightarrow{\epsilon(1)} \begin{matrix}\{p_1,p_2\}\\ \nu(p_1)=1\\ \nu(p_2)=4\end{matrix} \xrightarrow{t_1}$$

Strong semantics :

$$\begin{matrix}\{p_0,p_2\}\\ \nu(p_0)=0\\ \nu(p_2)=0\end{matrix} \xrightarrow{\epsilon(5)} \begin{matrix}\{p_0,\hat{p}_2\}\\ \nu(p_0)=5\\ \nu(p_2)=5\end{matrix} \xrightarrow{t_0} \begin{matrix}\{p_1,\hat{p}_2\}\\ \nu(p_1)=0\\ \nu(p_2)=5\end{matrix} \xrightarrow{\epsilon(\cdots)}$$

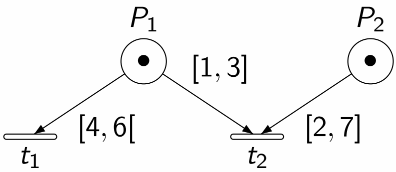
P-time Petri Nets (A-TPN)

Example (A TPN $\in$ A-TPN)



P-time Petri Nets (*A-TPN*)

Example (A TPN $\in$ *A-TPN*)

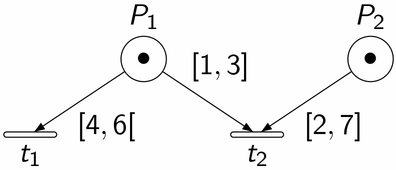


Strong semantics :

$$\begin{array}{c} \{p_1,p_2\} \\ \nu(p_1)=0 \\ \nu(p_2)=0 \end{array} \xrightarrow{\epsilon(3)} \begin{array}{c} \{p_1,p_2\} \\ \nu(p_1)=3 \\ \nu(p_2)=3 \end{array} \xrightarrow{t_2} \begin{array}{c} \{\} \\ . \\ . \end{array} \xrightarrow{\epsilon(\cdots)}$$

P-time Petri Nets ($A\text{-}TPN$)

Example (A TPN $\in A\text{-}TPN$)



Strong semantics :

$$\begin{array}{c}\{p_1,p_2\}\\ \nu(p_1)=0\\ \nu(p_2)=0\end{array} \overset{\epsilon(3)}{\rightarrow} \begin{array}{c}\{p_1,p_2\}\\ \nu(p_1)=3\\ \nu(p_2)=3\end{array} \overset{t_2}{\rightarrow} \begin{array}{c}\{\}\\ .\\ .\end{array} \overset{\epsilon(\cdots)}{\rightarrow}$$

Weak semantics :

$$\begin{array}{c}\{p_1,p_2\}\\ \nu(p_1)=0\\ \nu(p_2)=0\end{array} \overset{\epsilon(5)}{\rightarrow} \begin{array}{c}\{p_1,p_2\}\\ \nu(p_1)=5\\ \nu(p_2)=5\end{array} \overset{t_1}{\rightarrow} \begin{array}{c}\{p_2\}\\ \nu(p_2)=5\\ .\end{array} \overset{\epsilon(3)}{\rightarrow} \begin{array}{c}\{\hat{p}_2\}\\ \nu(p_2)=8\\ .\end{array} \overset{\epsilon(\cdots)}{\rightarrow}$$

Strong vs Weak Semantics

Let S, be a state of a Petri Nets, the strong semantics is defined by:

$$t \notin \mathit{firable}(S + d) \Rightarrow \forall d' \in [0, d] : t \notin \mathit{firable}(S + d') \qquad (2)$$

Strong vs Weak Semantics

Let S, be a state of a Petri Nets, the strong semantics is defined by:

$$t \notin \mathit{firable}(S + d) \Rightarrow \forall d' \in [0, d] : t \notin \mathit{firable}(S + d') \qquad (2)$$

With strong semantics, time elapsing cannot disable transition.

Plan

- **Petri nets with time**
- **T-time Petri Nets**
- **Other Semantics**
  - P-time Petri Nets
  - A-time Petri Nets
  - Strong vs Weak Semantics
- **Expressiveness and properties of TPN**
- **State Space of Time Petri Nets**
- **Model-checking of Time Petri Nets**
  - Temporal logics
  - Checking TPN with observers
- **Stopwatch Petri Nets**

Equivalence w.r.t. timed language and timed bisimulation.

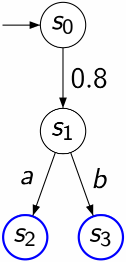
Two Timed Transition Systems (TTS)

Equivalence w.r.t. timed language and timed bisimulation.

Two Timed Transition Systems (TTS)

Equivalence w.r.t. timed language and timed bisimulation.

Two Timed Transition Systems (TTS)



Equivalence w.r.t. timed language and timed bisimulation.

Two Timed Transition Systems (TTS)

Equivalence w.r.t. timed language and timed bisimulation.

Two Timed Transition Systems (TTS)

- Both TTS accept the same timed language ie : the timed words $(a, 0.8)$ and $(b, 0.8)$

Equivalence w.r.t. timed language and timed bisimulation.

Two Timed Transition Systems (TTS)

- Both TTS accept the same timed language ie : the timed words $(a, 0.8)$ and $(b, 0.8)$
- These TTS are not timed bisimilar.

Comparison of the expressiveness of Time Petri Nets

Classes of TPN :

- Time Petri Nets where time is associated with places (*P-TPN*), arcs (*A-TPN*) or transitions (*T-TPN*).
- strong semantics ($\overline{T\text{-}TPN}$) or weak semantics ($\underline{T\text{-}TPN}$)
- safe nets (1-bounded)
- large or strict time constraints
- net with *label* and *epsilon* transition

Comparison of the expressiveness of Time Petri Nets

Classes of TPN :

- Time Petri Nets where time is associated with places (*P-TPN*), arcs (*A-TPN*) or transitions (*T-TPN*).
- strong semantics ($\overline{T\text{-}TPN}$) or weak semantics ($\underline{T\text{-}TPN}$)
- safe nets (1-bounded)
- large or strict time constraints
- net with *label* and *epsilon* transition

Comparison of expressiveness w.r.t. timed bisimilarity.

Results [BR08]

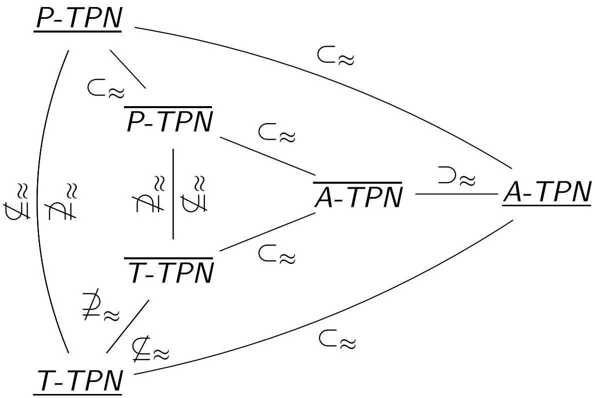


Figure: Comparison of expressiveness of TPN

Results [BR08]

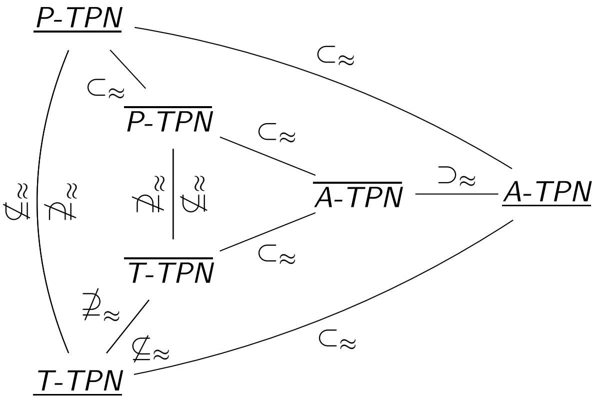


Figure: Comparison of expressiveness of TPN

However, *T-TPN* are easier than *A-TPN* for the modelling of synchronisations

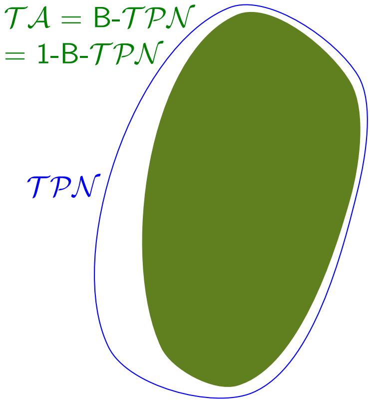
Expressivité : $\overline{T\text{-}TPN}$ *vs* *TA* (automates temporisés)

Timed language acceptance

Timed Bisimulation

Expressivité : $\overline{T\text{-}TPN}$ *vs* *TA* (automates temporisés)

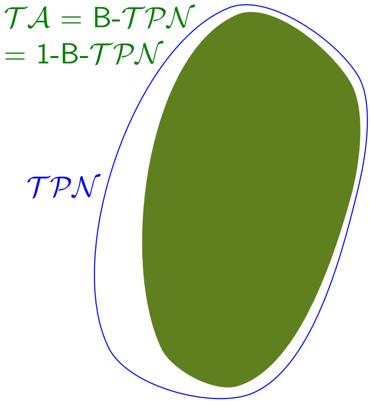
Timed language acceptance



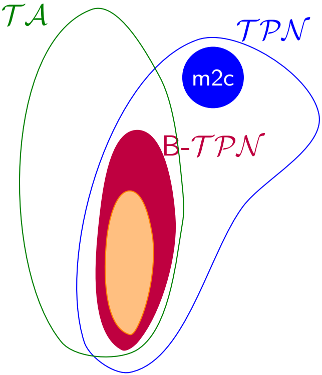
Timed Bisimulation

Expressivité : $\overline{T\text{-}TPN}$ *vs* *TA* (automates temporisés)

Timed language acceptance



Timed Bisimulation



B-$\mathcal{TPN}(\leq, \geq)$
= 1-B-$\mathcal{TPN}(\leq, \geq)$

Main Decidability results

T-time semantics with strong semantics ($\overline{T\text{-}TPN}$).

Problem	$\overline{\text{T-TPN}}$ (not bounded)	$\overline{\text{T-TPN}}$ (bounded)
Boundedness		
k-Boundedness		
Accessibility empty language		
Universality language inclusion		
Model-checking de TCTL		

Main Decidability results

T-time semantics with strong semantics ($\overline{T\text{-}TPN}$).

Problem	$\overline{\text{T-TPN}}$ (not bounded)	$\overline{\text{T-TPN}}$ (bounded)
Boundedness	Undecidable [JLL77]	
k-Boundedness	Decidable [BD91]	
Accessibility empty language	Undecidable [JLL77]	
Universality language inclusion	Undecidable [BCH+13]	
Model-checking de TCTL	Undecidable	

Main Decidability results

T-time semantics with strong semantics ($\overline{T\text{-}TPN}$).

Problem	$\overline{\textbf{T-TPN}}$ (not bounded)	$\overline{\textbf{T-TPN}}$ (bounded)
Boundedness	Undecidable [JLL77]	-
k-Boundedness	Decidable [BD91]	Decidable [BD91]
Accessibility empty language	Undecidable [JLL77]	Decidable [BD91]
Universality language inclusion	Undecidable [BCH+13]	Undecidable [BCH+13]
Model-checking de TCTL	Undecidable	Decidable [CR06, BGR09]

Exercise

TPN *vs* PN

- How can we simulate the behaviour of a PN by aTPN ?

Plan

- Petri nets with time
- T-time Petri Nets
- Other Semantics
- Expressiveness and properties of TPN
- **State Space of Time Petri Nets**
- Model-checking of Time Petri Nets
- Stopwatch Petri Nets

Model checking Problem

$\Rightarrow$ Explore the state space

Model checking Problem

$\Rightarrow$ Explore the state space

Problem

The state space of a TPN is infinite

Model checking Problem

$\Rightarrow$ Explore the state space

Problem

The state space of a TPN is infinite

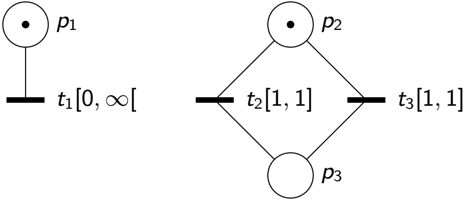
$\Rightarrow$ Group the states in equivalence classes (abstraction)

Exploration of the state space

- simulation

Exploration of the state space

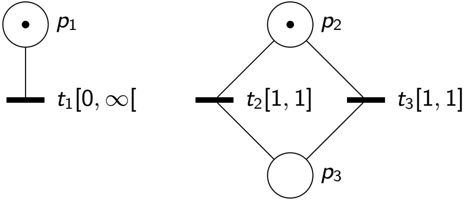
- simulation



$$\begin{pmatrix} 1 & \vline & 0 \\ 1, & \vline & 0 \\ 0 & \vline & \times \end{pmatrix}$$

Exploration of the state space

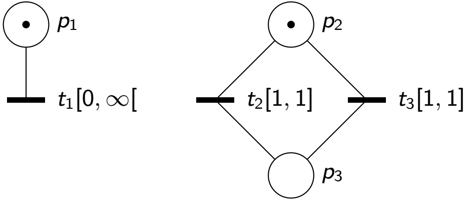
- simulation



$$\begin{pmatrix} 1 & 0 \\ 1, & 0 \\ 0 & \times \end{pmatrix} \xrightarrow{0} \begin{pmatrix} 1 & \delta \\ 1, & \delta \\ 0 & \times \end{pmatrix}$$

Exploration of the state space

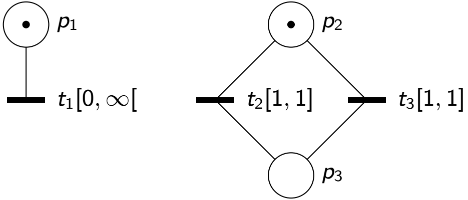
- simulation



$$\left(\begin{array}{l|l}1 & 0\\ 1, & 0\\ 0 & \times\end{array}\right) \xrightarrow{0} \xrightarrow{0.7} \left(\begin{array}{l|l}1 & \delta\\ 1, & \delta\\ 0 & \times\end{array}\right)$$

Exploration of the state space

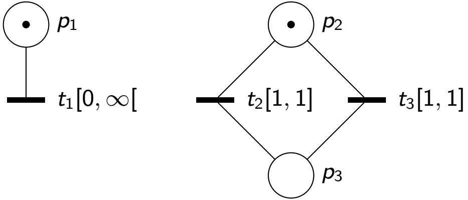
- simulation



$$\left(\begin{array}{l|l} 1 & 0 \\ 1, & 0 \\ 0 & \times \end{array}\right) \quad \begin{array}{c} \xrightarrow{0} \\ \xrightarrow{0.7} \\ \cdots \\ \xrightarrow{1} \end{array} \quad \left(\begin{array}{l|l} 1 & \delta \\ 1, & \delta \\ 0 & \times \end{array}\right)$$

Exploration of the state space

- simulation

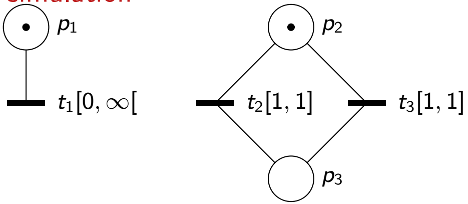


$$\begin{pmatrix} 1 & \vert & 0 \\ 1, & \vert & 0 \\ 0 & \vert & \times \end{pmatrix} \quad \begin{array}{c} \xrightarrow{0} \\ \xrightarrow{0.7} \\ \cdots \\ \xrightarrow{1} \end{array} \quad \begin{pmatrix} 1 & \vert & \delta \\ 1, & \vert & \delta \\ 0 & \vert & \times \end{pmatrix}$$

$\rightarrow$ Infinity of branchings, states

Exploration of the state space

- simulation



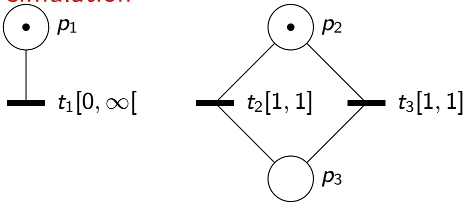
$$\left(\begin{array}{c|c} 1 & 0 \\ 1, & 0 \\ 0 & \times \end{array}\right) \begin{array}{c} \xrightarrow{0} \\ \xrightarrow{0.7} \\ \cdots \\ \xrightarrow{1} \end{array} \left(\begin{array}{c|c} 1 & \delta \\ 1, & \delta \\ 0 & \times \end{array}\right)$$

$\rightarrow$ Infinity of branchings, states

- symbolic

Exploration of the state space

- simulation



$$\left(\begin{array}{l|l} 1 & 0 \\ 1, & 0 \\ 0 & \times \end{array}\right) \begin{array}{c} \xrightarrow{0} \\ \xrightarrow{0.7} \\ \cdots \\ \xrightarrow{1} \end{array} \left(\begin{array}{l|l} 1 & \delta \\ 1, & \delta \\ 0 & \times \end{array}\right)$$

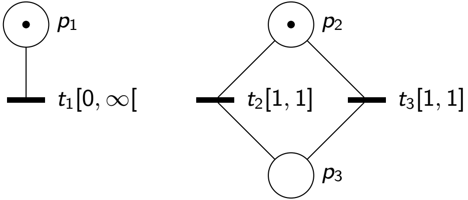
$\rightarrow$ Infinity of branchings, states

- symbolic
  - symbolic states:

$$\begin{pmatrix} 1 \\ 1, x_1 = x_2 = 0 \\ 0 \end{pmatrix}$$

Exploration of the state space

- simulation



$$\left(\begin{array}{l|l} 1 & 0 \\ 1, & 0 \\ 0 & \times \end{array}\right) \begin{array}{c} \xrightarrow{0} \\ \xrightarrow{0.7} \\ \cdots \\ \xrightarrow{1} \end{array} \left(\begin{array}{l|l} 1 & \delta \\ 1, & \delta \\ 0 & \times \end{array}\right)$$

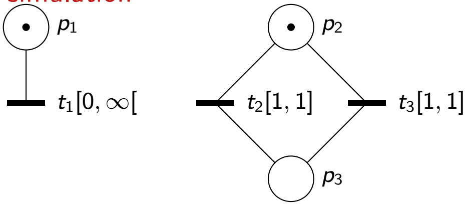
$\rightarrow$ Infinity of branchings, states

- symbolic
  - symbolic states:

$$\begin{pmatrix} 1 \\ 1, x_1 = x_2 = 0 \\ 0 \end{pmatrix} \xrightarrow{\delta}$$

Exploration of the state space

- simulation



$$\begin{pmatrix} 1 & 0 \\ 1, & 0 \\ 0 & \times \end{pmatrix} \begin{array}{c} \xrightarrow{0} \\ \xrightarrow{0.7} \\ \cdots \\ \xrightarrow{1} \end{array} \begin{pmatrix} 1 & \delta \\ 1, & \delta \\ 0 & \times \end{pmatrix}$$

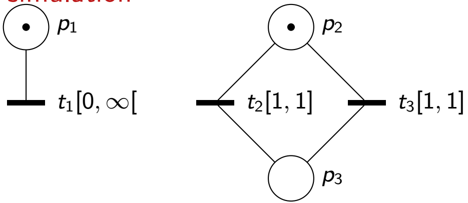
$\rightarrow$ Infinity of branchings, states

- symbolic
  - symbolic states:

$$\begin{pmatrix} 1 \\ 1, x_1 = x_2 = 0 \\ 0 \end{pmatrix} \xrightarrow{\delta} \begin{pmatrix} 1 \\ 1, x_1 = x_2 \in [0, 1] \\ 0 \end{pmatrix}$$

Exploration of the state space

- simulation



$$\left(\begin{array}{l|l} 1 & 0 \\ 1, & 0 \\ 0 & \times \end{array}\right) \begin{array}{c} \xrightarrow{0} \\ \xrightarrow{0.7} \\ \cdots \\ \xrightarrow{1} \end{array} \left(\begin{array}{l|l} 1 & \delta \\ 1, & \delta \\ 0 & \times \end{array}\right)$$

$\rightarrow$ Infinity of branchings, states

- symbolic
  - symbolic states:

$$\begin{pmatrix} 1 \\ 1, x_1 = x_2 = 0 \\ 0 \end{pmatrix} \xrightarrow{\delta} \begin{pmatrix} 1 \\ 1, x_1 = x_2 \in [0, 1] \\ 0 \end{pmatrix}$$

$\rightarrow$ State class graph
$\rightarrow$ Regions graph
$\rightarrow$ Zones graph

State space computation

Definition (Symbolic state)

$\mathcal{Q} = (M, \mathcal{V})$

- M a marking,
- $\mathcal{V}$ a set of valuation such that M exists.

Basic Algorithm for the state space computation

Basic Algorithm

The set of state to explore: $Waiting \leftarrow \mathcal{Q}_0 = (M_0, \mathcal{V}_0)$.

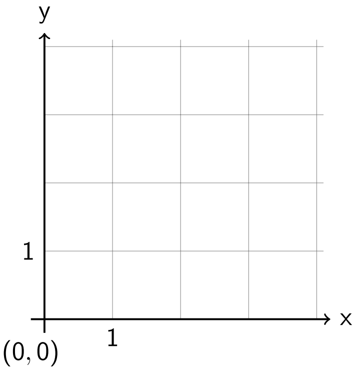
The set of explored states: $Visited \leftarrow \emptyset$

While $Waiting \neq \emptyset$

- $\mathcal{Q} \leftarrow pop(Waiting)$
- Computation of the fireable transitions from $\mathcal{Q}$: $firable(\mathcal{Q})$
- for all transition $t \in firable(\mathcal{Q})$
  - Compute the successor of $\mathcal{Q}$ by the firing of t : $next(\mathcal{Q}, t)$
  - if $next(\mathcal{Q}, t) \notin (Waiting \cup Visited)$
    then $Waiting \leftarrow Waiting \cup next(\mathcal{Q}, t)$
- $Visited \leftarrow Visited \cup \mathcal{Q}$

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region r should

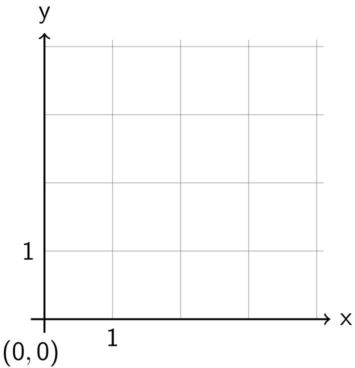


Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions

all clock valuations of a region r should

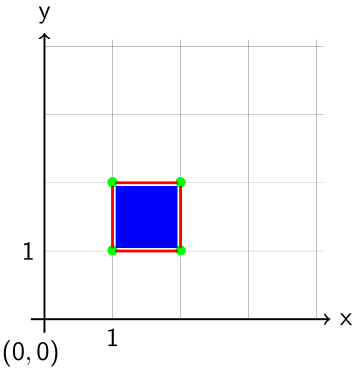
1. satisfy the same clock constraints



Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region r should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing

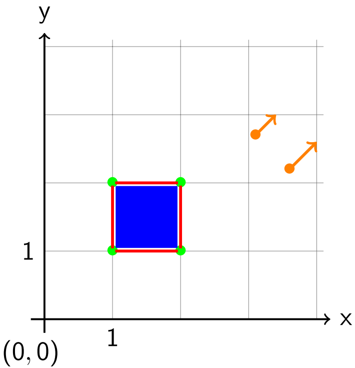


1. rectangular regions of types points, open lines, open squares satisfy (1)
$x = 1 \wedge 1 < y < 2$,
$1 < x < 2 \wedge 1 < y < 2$, $x = 2 \wedge y = 2$

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region r should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing

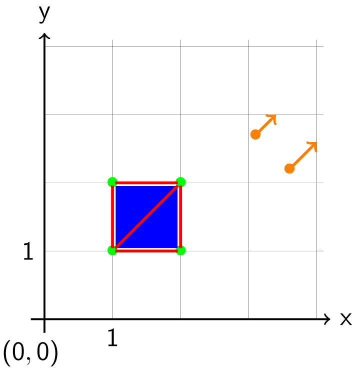


1. rectangular regions of types points, open lines, open squares satisfy (1)
$x = 1 \land 1 < y < 2$,
$1 < x < 2 \land 1 < y < 2$, $x = 2 \land y = 2$

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region r should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing



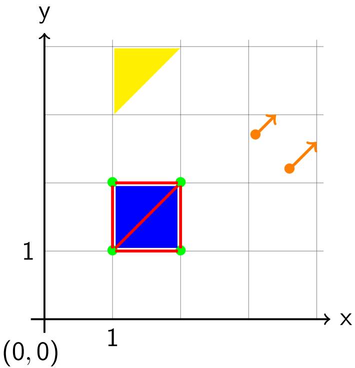
1. rectangular regions of types points, open lines, open squares satisfy (1)
$x = 1 \wedge 1 < y < 2$,
$1 < x < 2 \wedge 1 < y < 2$, $x = 2 \wedge y = 2$

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions

all clock valuations of a region r should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing

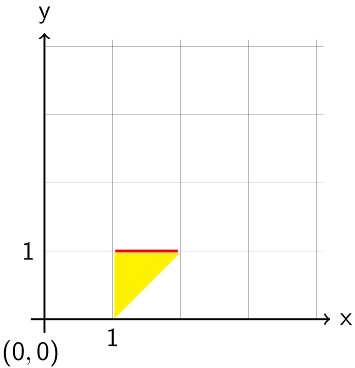


1. rectangular regions of types points, open lines, open squares satisfy (1)
   $x = 1 \wedge 1 < y < 2$,
   $1 < x < 2 \wedge 1 < y < 2$, $x = 2 \wedge y = 2$

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region *r* should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing

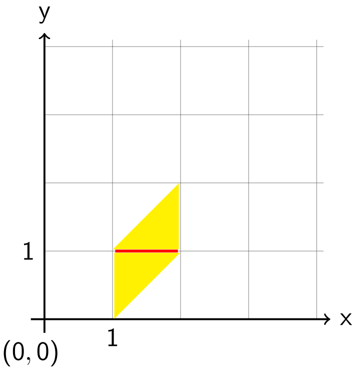


1. rectangular regions of types points, open lines, open squares satisfy (1)
   $x = 1 \wedge 1 < y < 2$,
   $1 < x < 2 \wedge 1 < y < 2$, $x = 2 \wedge y = 2$
2. = region defined by constraint
   $1 < x < 2 \wedge 0 < y < 1 \wedge x - y < 0$

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region r should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing

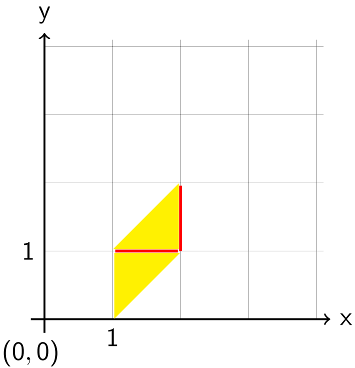


1. rectangular regions of types points, open lines, open squares satisfy (1)
   $x = 1 \wedge 1 < y < 2$,
   $1 < x < 2 \wedge 1 < y < 2$, $x = 2 \wedge y = 2$
2. = region defined by constraint
   $1 < x < 2 \wedge 0 < y < 1 \wedge x - y < 0$
3. regions encountered by time elapsing

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region r should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing

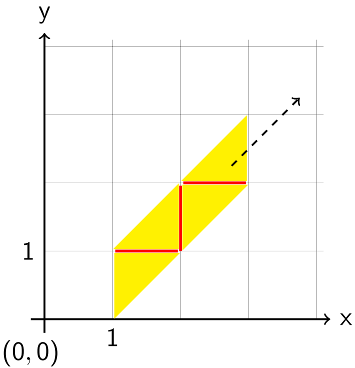


1. rectangular regions of types points, open lines, open squares satisfy (1) $x = 1 \wedge 1 < y < 2$, $1 < x < 2 \wedge 1 < y < 2$, $x = 2 \wedge y = 2$
2. = region defined by constraint $1 < x < 2 \wedge 0 < y < 1 \wedge x - y < 0$
3. regions encountered by time elapsing

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region r should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing

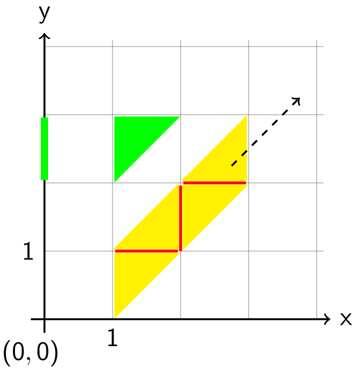


1. rectangular regions of types points, open lines, open squares satisfy (1)
$x = 1 \wedge 1 < y < 2$,
$1 < x < 2 \wedge 1 < y < 2$, $x = 2 \wedge y = 2$
2. = region defined by constraint
$1 < x < 2 \wedge 0 < y < 1 \wedge x - y < 0$
3. regions encountered by time elapsing

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region *r* should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing

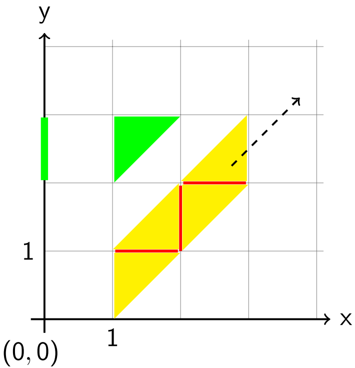


1. rectangular regions of types points, open lines, open squares satisfy (1)
   $x = 1 \wedge 1 < y < 2$,
   $1 < x < 2 \wedge 1 < y < 2$, $x = 2 \wedge y = 2$
2. = region defined by constraint
   $1 < x < 2 \wedge 0 < y < 1 \wedge x - y < 0$
3. regions encountered by time elapsing
4. regions obtained after a reset

Region Abstraction [ACD90]

Idea: group clock valuations into equivalence classes: regions
all clock valuations of a region r should

1. satisfy the same clock constraints
2. be able to reach the same regions by time elapsing



1. rectangular regions of types points, open lines, open squares satisfy (1) $x = 1 \wedge 1 < y < 2$, $1 < x < 2 \wedge 1 < y < 2$, $x = 2 \wedge y = 2$
2. = region defined by constraint $1 < x < 2 \wedge 0 < y < 1 \wedge x - y < 0$
3. regions encountered by time elapsing
4. regions obtained after a reset

Still infinitely many regions

Zone Graph

Region Graph → theoretical method

Zone Graph

Region Graph $\rightarrow$ theoretical method

Zone graph

- Grouping of regions
- Good results in practice

Definition (Zone)

$\mathcal{Z} = (M, Z)$ where M is a marking and Z is a polyhedron.

$$Z = \begin{cases} \forall i, j \text{ t.q. } t_i, t_j \in enabled(M), \\ x_i \leq z_i, \\ x_j \leq z_j, \\ x_i - x_j \leq z_{ij} \end{cases}$$

(x_i represents the clock associated with the transition i)

$\Rightarrow$ Difference Bound Matrix (DBM)

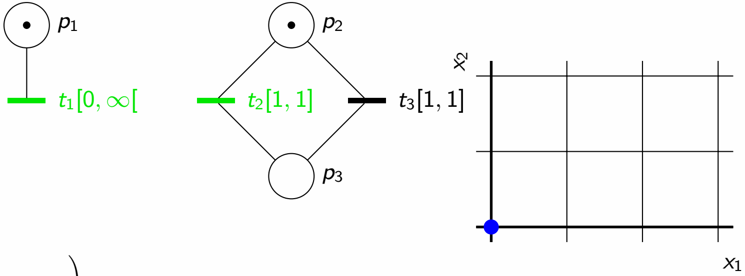
Computation of the state space

$\mathcal{Z}_0 = (M_0, Z_0)$
Computation of the states reachable by time elapsing (futur)

Computation of the state space

$\mathcal{Z}_0 = (M_0, Z_0)$
Computation of the states reachable by time elapsing (futur)

Example



$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 = 0\right)$$

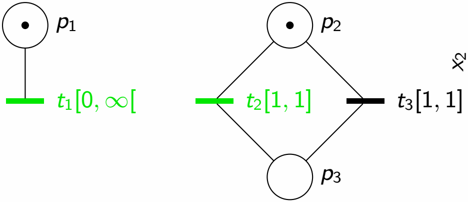
$\overrightarrow{Z_0} \cap Inv\,(M_0)$, with $Inv\,(M_0) = \wedge_{t_i}\{x_i \leq \beta_i\}$, $t_i \in enabled\,(M_0)$

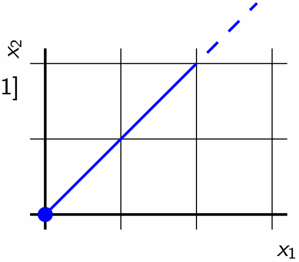
Computation of the state space

$\mathcal{Z}_0 = (M_0, Z_0)$

Computation of the states reachable by time elapsing (futur)

Example





$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 = 0\right) \rightarrow \left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 \in [0, \infty[\right)$$

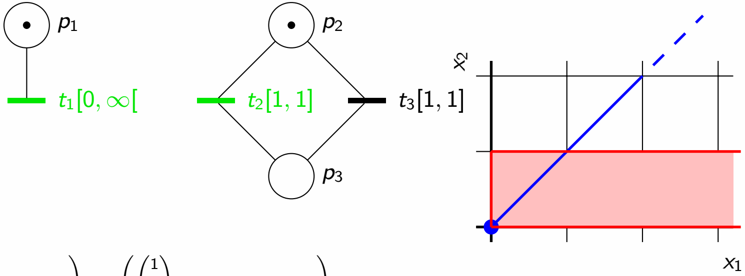
$\overrightarrow{Z_0} \cap Inv\,(M_0)$, with $Inv\,(M_0) = \wedge_{t_i}\{x_i \leq \beta_i\}$, $t_i \in enabled\,(M_0)$

Computation of the state space

$\mathcal{Z}_0 = (M_0, Z_0)$

Computation of the states reachable by time elapsing (futur)

Example



$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 = 0\right) \rightarrow \left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 \in [0, \infty[\right)$$

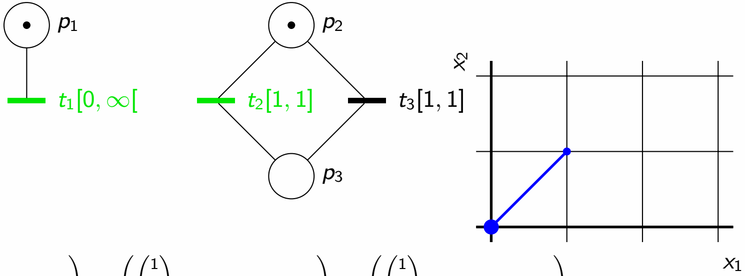
$\overrightarrow{Z_0} \cap Inv\,(M_0)$, with $Inv\,(M_0) = \wedge_{t_i}\{x_i \leq \beta_i\}, t_i \in enabled\,(M_0)$

Computation of the state space

$\mathcal{Z}_0 = (M_0, Z_0)$

Computation of the states reachable by time elapsing (futur)

Example



$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 = 0\right) \rightarrow \left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 \in [0, \infty[\right) \rightarrow \left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 \in [0, 1]\right)$$

$\overrightarrow{Z_0} \cap Inv\,(M_0)$, with $Inv\,(M_0) = \wedge_{t_i}\{x_i \leq \beta_i\}$, $t_i \in enabled\,(M_0)$

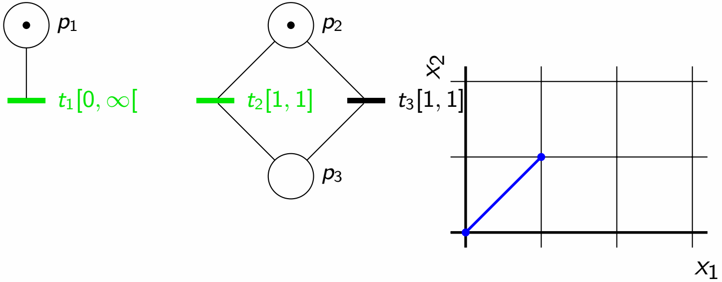
Computation of the state space

Computation of fireable transitions.

Computation of the state space

Computation of fireable transitions.

Example

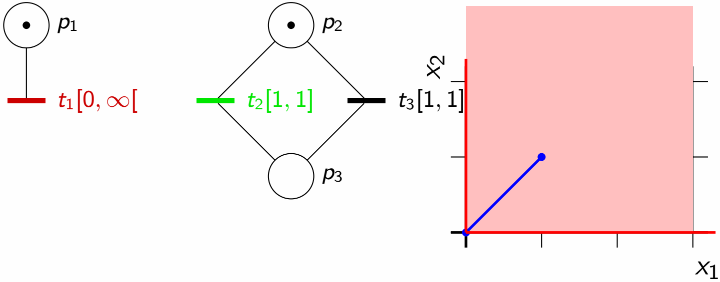


$$\{t_i \mid (\overrightarrow{Z_0} \cap Inv\,(M_0) \cap x_i \geq \alpha_i) \neq \emptyset\}$$

Computation of the state space

Computation of fireable transitions.

Example

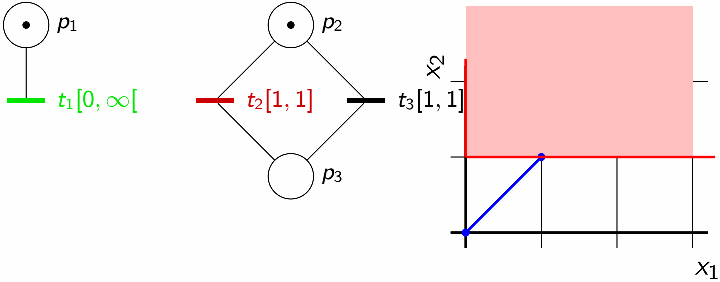


$$\{t_i \mid (\overrightarrow{Z_0} \cap Inv\,(M_0) \cap x_i \geq \alpha_i) \neq \emptyset\}$$

Computation of the state space

Computation of fireable transitions.

Example



$$\{t_i \mid (\overrightarrow{Z_0} \cap Inv\,(M_0) \cap x_i \geq \alpha_i) \neq \emptyset\}$$

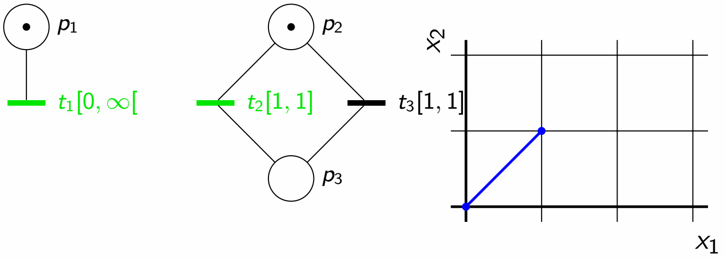
Computation of the state space

Firing of transitions $\rightarrow (M_i, Z_i)$.

Computation of the state space

Firing of transitions $\rightarrow (M_i, Z_i)$.

Example

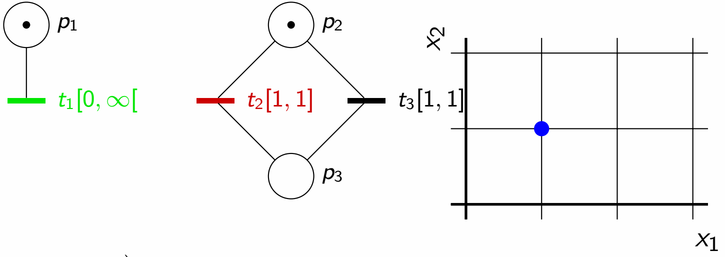


$(M_i, Z_i) = \left(M_0 - {}^\bullet t_i + t_i^\bullet, \left(\overrightarrow{Z_0} \cap Inv\,(M_0) \cap x_i \geq \alpha_i \right) [X_{ne} \leftarrow 0] \right)$ X_{ne} : ensemble des transitions nouvellement sensibilisées

Computation of the state space

Firing of transitions $\rightarrow (M_i, Z_i)$.

Example



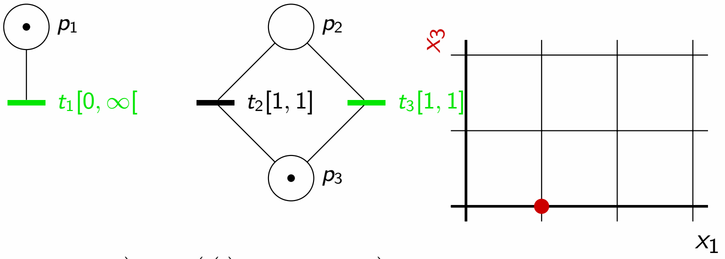
$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 \in [1,1]\right)$$

$(M_i, Z_i) = \left(M_0 - {}^\bullet t_i + t_i^\bullet, \left(\overrightarrow{Z_0} \cap Inv\left(M_0\right) \cap x_i \geq \alpha_i\right)[X_{ne} \leftarrow 0]\right)$ X_{ne} : ensemble des transitions nouvellement sensibilisées

Computation of the state space

Firing of transitions $\rightarrow (M_i, Z_i)$.

Example



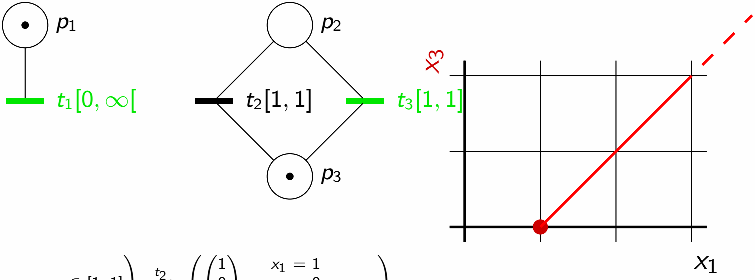
$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 \in [1,1]\right) \xrightarrow{t_2} \left(\begin{pmatrix}1\\0\\1\end{pmatrix}, x_1 = 1 \wedge x_3 = 0\right)$$

$(M_i, Z_i) = \left(M_0 - {}^\bullet t_i + t_i^\bullet, \left(\overrightarrow{Z_0} \cap Inv\left(M_0\right) \cap x_i \geq \alpha_i\right)[X_{ne} \leftarrow 0]\right)$ X_{ne} : ensemble des transitions nouvellement sensibilisées

Computation of the state space

Firing of transitions $\rightarrow (M_i, Z_i)$.

Example



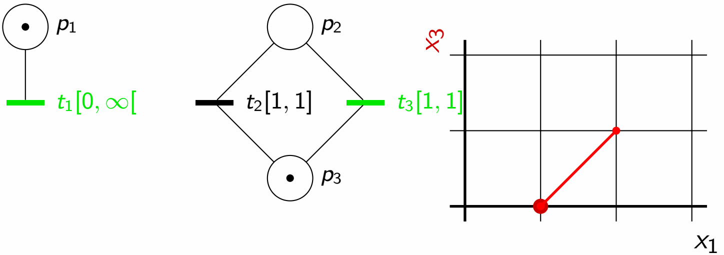
$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, x_1 = x_2 \in [1,1]\right) \xrightarrow{t_2} \left(\begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{array}{l} x_1 = 1 \\ x_3 = 0 \\ x_1 - x_3 = 1 \end{array}\right)$$

$(M_i, Z_i) = \left(M_0 - {}^\bullet t_i + t_i^\bullet, \left(\overrightarrow{Z_0} \cap Inv\,(M_0) \cap x_i \geq \alpha_i\right)[X_{ne} \leftarrow 0]\right)$ X_{ne} : ensemble des transitions nouvellement sensibilisées

Computation of the state space

Firing of transitions $\rightarrow (M_i, Z_i)$.

Example



$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, x_1 = x_2 \in [1, 1] \right) \xrightarrow{t_2} \left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{array}{l} x_1 = 1 \\ x_3 = 0 \\ x_1 - x_3 = 1 \end{array} \right)$$

$(M_i, Z_i) = \left(M_0 - {}^\bullet t_i + t_i^\bullet, \left(\overrightarrow{Z_0} \cap Inv\left(M_0\right) \cap x_i \geq \alpha_i \right) [X_{ne} \leftarrow 0] \right)$ X_{ne} : ensemble des transitions nouvellement sensibilisées

Next(t)

Let t a transition with the firing interval $[\alpha, \beta]$ fireable from (M_i, Z_i). The computation of the successor of (M_i, Z_i) by the firing of t is $next((M_i, Z_i), t) = (M_j, Z_j)$ where Z_j is computed as follows:

- compute the firing space of the transition t : $Z_i \cap (x_t \geq \alpha)$ where x_t is the clock associated with t
- eliminate x_t (for example by using Fourier-Motzkin method)
- add (or reset) the clocks of the newly enabled transitions: x_{new} and for all other clocks x_{old} (with $min \leq x_{old} \leq max$), add the new diagonal constraints $min \leq x_{old} - x_{new} \leq max$)
- compute the futur (time elapsing)
- add the constraints $x \leq \beta$
- compute the canonical form

Computation of the state space

Exercise

Go back to the previous example and add $t_1 \to P_4 \to t_4[2,3]$ to the net.

- Compute the initial zone Z_0 and its successor Z_1 by the firing of t_1 and give the detail of the method (Conjonction with the guard $x_1 \geq 0$. Elimination of x_1 by Fourier Motzkin. Add x_4. Futur. Conjonction with invariants. Canonical form).
- Compute (literal expression and graphical representation) all the zones of the following sequence: $Z_0 \xrightarrow{t_1} Z_1 \xrightarrow{t_2} Z_2 \xrightarrow{t_3} Z_3 \xrightarrow{t_2} Z_4 \xrightarrow{t_3} Z_5$ and give for each zone, the list of fireable transitions.
  Give the transition fireable from Z_5 ?
- Simulate the TPN with the tool Roméo on example: Ex1-a-Master.xml

Computation of the state space

Theorem (Convergence)

The algorithm converges for time Petri Nets:

- *bounded*
- $\beta : T \to \mathbb{Q}^+$

Computation of the state space

Theorem (Convergence)

The algorithm converges for time Petri Nets:

- *bounded*
- $\beta : T \to \mathbb{Q}^+$

Implementation with DBM :

- previous example with $t_1[4, 4]$ with Romeo

Computation of the state space

Theorem (Convergence)

The algorithm converges for time Petri Nets:

- *bounded*
- $\beta : T \to \mathbb{Q}^+$

Implementation with DBM :

- previous example with $t_1[4,4]$ with Romeo

$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 = x_2 \end{pmatrix} \right)$$

Computation of the state space

Theorem (Convergence)

The algorithm converges for time Petri Nets:

- *bounded*
- $\beta : T \to \mathbb{Q}^+$

Implementation with DBM :

- previous example with $t_1[4,4]$ with Romeo

$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 = x_2 \end{pmatrix} \right) \xrightarrow{t_2} \left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \leq x_1 \leq 2 \\ 0 \leq x_3 \leq 1 \\ x_1 - x_3 = 1 \end{pmatrix} \right)$$

Computation of the state space

Theorem (Convergence)

The algorithm converges for time Petri Nets:

- *bounded*
- $\beta : T \to \mathbb{Q}^+$

Implementation with DBM :

- previous example with $t_1[4,4]$ with Romeo

$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}0 \leq x_2 \leq 1\\ x_1 = x_2\end{pmatrix}\right) \xrightarrow{t_2} \left(\begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}1 \leq x_1 \leq 2\\ 0 \leq x_3 \leq 1\\ x_1 - x_3 = 1\end{pmatrix}\right) \xrightarrow{t_3} \left(\begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}2 \leq x_1 \leq 3\\ 0 \leq x_2 \leq 1\\ x_1 - x_2 = 2\end{pmatrix}\right) \dots$$

Computation of the state space

Theorem

The algorithm converges for time Petri Nets:

- *bounded*
- $\beta : T \to \mathbb{Q}^+$

Computation of the state space

Theorem

The algorithm converges for time Petri Nets:

- *bounded*
- $\beta : T \to \mathbb{Q}^+$

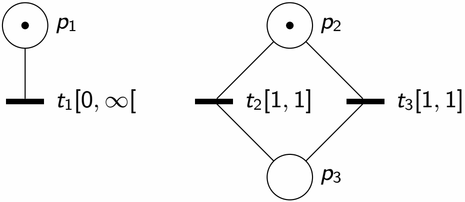
Problem

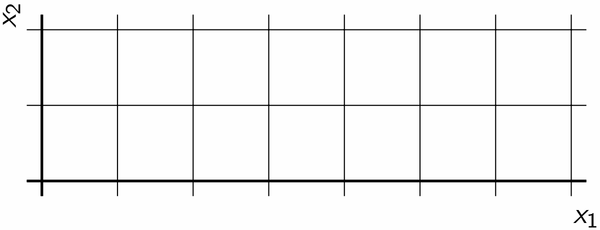
Termination for $\beta : T \to \mathbb{Q}^+ \cup \{\infty\}$?

Computation of the state space

Example

with $\beta = \infty$

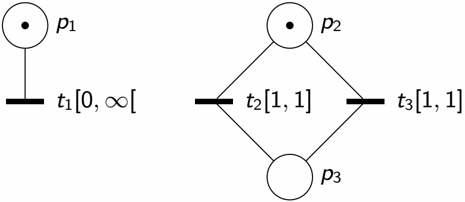


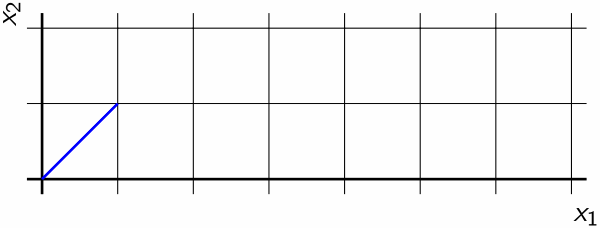


Computation of the state space

Example

with $\beta = \infty$

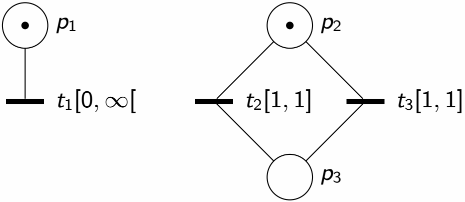


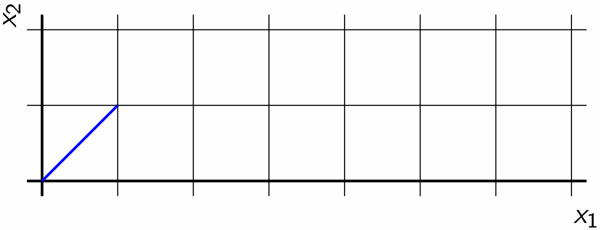


Computation of the state space

Example

with $\beta = \infty$

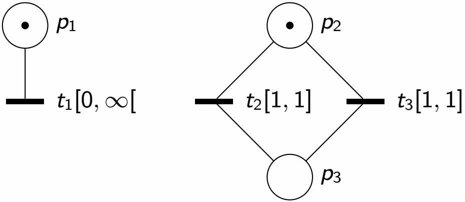


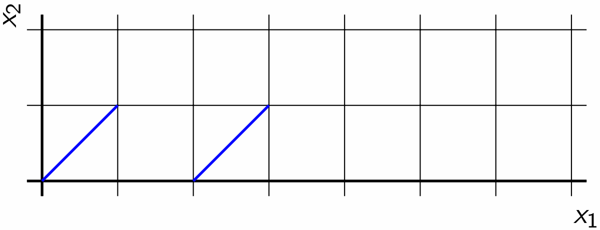


Computation of the state space

Example

with $\beta = \infty$

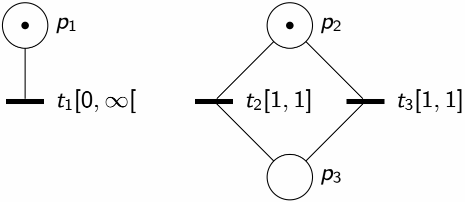


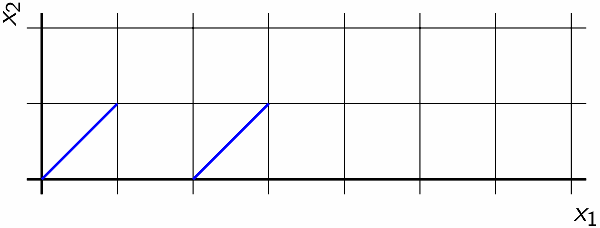


Computation of the state space

Example

with $\beta = \infty$

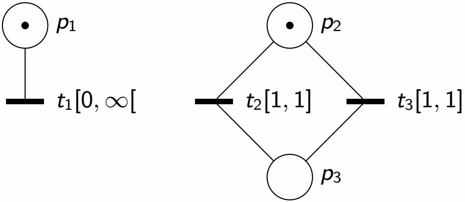


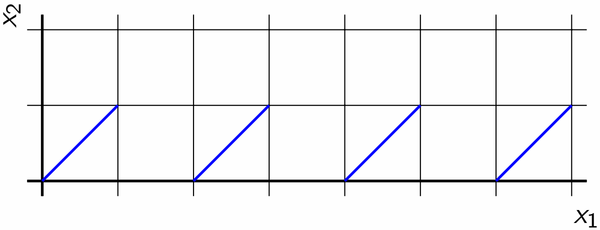


Computation of the state space

Example

with $\beta = \infty$





Computation of the state space

Transition with interval $[a, \infty[$:

- Information importante : $x \geq a$

$\rightarrow$ Utilisation d'un opérateur d'approximation *k-approx*

- Choix de k :

$$k = max_{t \in T \mid \beta(t) \neq \infty} (\alpha(t), \beta(t))$$

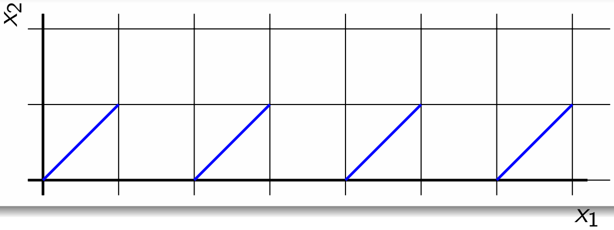
Computation of the state space

Apply *approx* in the computation of the successor

Computation of the state space

Apply *approx* in the computation of the successor

Example

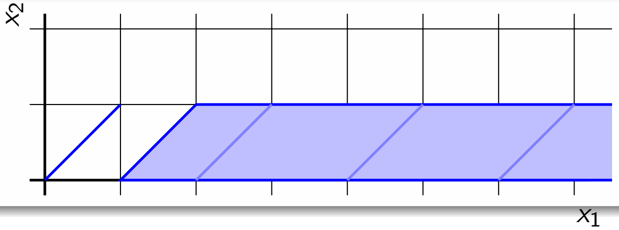


$$(M_i, Z_i) = \left(M_0 -^{\bullet} t_i + t_i^{\bullet}, k - approx\left(\left(\overrightarrow{Z_0} \cap Inv\left(M_0\right) \cap x_i \geq \alpha_i\right)\left[X_{ne} := 0\right]\right)\right)$$

Computation of the state space

Apply *approx* in the computation of the successor

Example



$$(M_i, Z_i) =$$
$$\left(M_0 -^{\bullet} t_i + t_i^{\bullet}, k - approx\left(\left(\overrightarrow{Z_0} \cap Inv\,(M_0) \cap x_i \geq \alpha_i\right)[X_{ne} := 0]\right)\right)$$

Computation of the state space

Theorem

The zone graph algorithm with k-approximation is exact with respect to marking reachability for bounded TPN with latest firing time in the set $\mathbb{Q}^+ \cup \{\infty\}$.

Computation of the state space

Theorem

The zone graph algorithm with k-approximation is exact with respect to marking reachability for bounded TPN with latest firing time in the set $\mathbb{Q}^+ \cup \{\infty\}$.

Implementation with DBM :

- previous example with $t_1[4, \infty]$

Computation of the state space

Theorem

The zone graph algorithm with k-approximation is exact with respect to marking reachability for bounded TPN with latest firing time in the set $\mathbb{Q}^+ \cup \{\infty\}$.

Implementation with DBM :

- previous example with $t_1[4, \infty]$

$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 = x_2 \end{pmatrix} \right)$$

Computation of the state space

Theorem

The zone graph algorithm with k-approximation is exact with respect to marking reachability for bounded TPN with latest firing time in the set $\mathbb{Q}^+ \cup \{\infty\}$.

Implementation with DBM :

- previous example with $t_1[4, \infty]$

$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 = x_2 \end{pmatrix} \right) \xrightarrow{t_2} \left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \leq x_1 \leq 2 \\ 0 \leq x_3 \leq 1 \\ x_1 - x_3 = 1 \end{pmatrix} \right)$$

Computation of the state space

Theorem

The zone graph algorithm with k-approximation is exact with respect to marking reachability for bounded TPN with latest firing time in the set $\mathbb{Q}^+ \cup \{\infty\}$.

Implementation with DBM :

- previous example with $t_1[4, \infty]$

$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}0 \leq x_2 \leq 1\\ x_1 = x_2\end{pmatrix}\right) \xrightarrow{t_2} \left(\begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}0 \leq x_1 \leq 2\\ 0 \leq x_3 \leq 1\\ x_1 - x_3 = 1\end{pmatrix}\right) \xrightarrow{t_3} \left(\begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}0 \leq x_1 \leq 3\\ 0 \leq x_2 \leq 1\\ x_1 - x_2 = 2\end{pmatrix}\right) \dots$$

Computation of the state space

Theorem

The zone graph algorithm with k-approximation is exact with respect to marking reachability for bounded TPN with latest firing time in the set $\mathbb{Q}^+ \cup \{\infty\}$.

Implementation with DBM :

- previous example with $t_1[4, \infty]$

$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 = x_2 \end{pmatrix} \right) \xrightarrow{t_2} \left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \leq x_1 \leq 2 \\ 0 \leq x_3 \leq 1 \\ x_1 - x_3 = 1 \end{pmatrix} \right) \xrightarrow{t_3} \left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \leq x_1 \leq 3 \\ 0 \leq x_2 \leq 1 \\ x_1 - x_2 = 2 \end{pmatrix} \right) \dots$$

- with $t_1[0, \infty]$

Computation of the state space

Theorem

The zone graph algorithm with k-approximation is exact with respect to marking reachability for bounded TPN with latest firing time in the set $\mathbb{Q}^+ \cup \{\infty\}$.

Implementation with DBM :

- previous example with $t_1[4, \infty]$

$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix} \begin{pmatrix}0 \le x_2 \le 1\\ x_1 = x_2\end{pmatrix}\right) \xrightarrow{t_2} \left(\begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}0 \le x_1 \le 2\\ 0 \le x_3 \le 1\\ x_1 - x_3 = 1\end{pmatrix}\right) \xrightarrow{t_3} \left(\begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}0 \le x_1 \le 3\\ 0 \le x_2 \le 1\\ x_1 - x_2 = 2\end{pmatrix}\right) \dots$$

- with $t_1[0, \infty]$ $\left(\begin{pmatrix}1\\1\\0\end{pmatrix} \begin{pmatrix}0 \le x_2 \le 1\\ x_1 = x_2\end{pmatrix}\right)$

Computation of the state space

Theorem

The zone graph algorithm with k-approximation is exact with respect to marking reachability for bounded TPN with latest firing time in the set $\mathbb{Q}^+ \cup \{\infty\}$.

Implementation with DBM :

- previous example with $t_1[4, \infty]$

$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 = x_2 \end{pmatrix} \right) \xrightarrow{t_2} \left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \leq x_1 \leq 2 \\ 0 \leq x_3 \leq 1 \\ x_1 - x_3 = 1 \end{pmatrix} \right) \xrightarrow{t_3} \left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \leq x_1 \leq 3 \\ 0 \leq x_2 \leq 1 \\ x_1 - x_2 = 2 \end{pmatrix} \right) \ldots$$

- with $t_1[0, \infty]$ $\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 = x_2 \end{pmatrix} \right) \xrightarrow{t_2} \left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \leq x_3 \leq 1 \\ x_1 - x_3 \geq 0 \end{pmatrix} \right)$

Computation of the state space

Theorem

The zone graph algorithm with k-approximation is exact with respect to marking reachability for bounded TPN with latest firing time in the set $\mathbb{Q}^+ \cup \{\infty\}$.

Implementation with DBM :

- previous example with $t_1[4, \infty]$

$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 = x_2 \end{pmatrix} \right) \xrightarrow{t_2} \left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \leq x_1 \leq 2 \\ 0 \leq x_3 \leq 1 \\ x_1 - x_3 = 1 \end{pmatrix} \right) \xrightarrow{t_3} \left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \leq x_1 \leq 3 \\ 0 \leq x_2 \leq 1 \\ x_1 - x_2 = 2 \end{pmatrix} \right) \ldots$$

- with $t_1[0, \infty]$

$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 = x_2 \end{pmatrix} \right) \xrightarrow{t_2} \left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \leq x_3 \leq 1 \\ x_1 - x_3 \geq 0 \end{pmatrix} \right) \xrightarrow{t_3} \left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \leq x_2 \leq 1 \\ x_1 - x_2 \geq 0 \end{pmatrix} \right) \ldots$$

Computation of the state space - State class graph

Definition (State class [BD91])

A State class C is a pair (M, D) where M is a marking and D is a set of inequalities (a polyhedron) called the firing domain.

$$D = \begin{cases} a_i \leq \theta_i \leq b_i, \ \forall i \text{ s.t. } t_i \text{ is enabled}, \\ -c_{kj} \leq \theta_j - \theta_k \leq c_{jk}, \ \forall j, k \text{ s.t. } \begin{cases} j \neq k \\ t_j, t_k \in enabled(M) \end{cases} \end{cases}$$

(θ_i represents the firing time of t_i relatively to the time when the class C was entered in)

$\Rightarrow$ Difference Bound Matrix (DBM)

Computation of the state space - State class graph

Intuition

Let the following Petri net: $P1 \rightarrow T1$, $P2 \rightarrow T2$, $P3 \rightarrow T3$
avec $M_o = \{P1, P2, P3\}$ $T1[3,5]$, $T2[7,9]$, $T3[4,6]$

- Give the firing intervals of the transitions T_1, T_2 and T_3 with the variables θ_1, θ_2 and θ_3
- Deduce the fireable transitions.
- Deduce intuitively (without diagonal constraints)the firing intervals of the transitions T_2 and T_3 after the firing of T_1.

Computation of the state space - State class graph

$\mathcal{C}_0 = (M_0, D_0)$

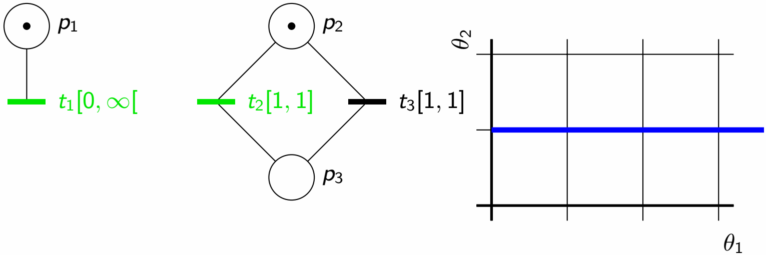
Computation of the fireable transitions

Computation of the state space - State class graph

$\mathcal{C}_0 = (M_0, D_0)$

Computation of the fireable transitions

Example



$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array} \right. \right)$$

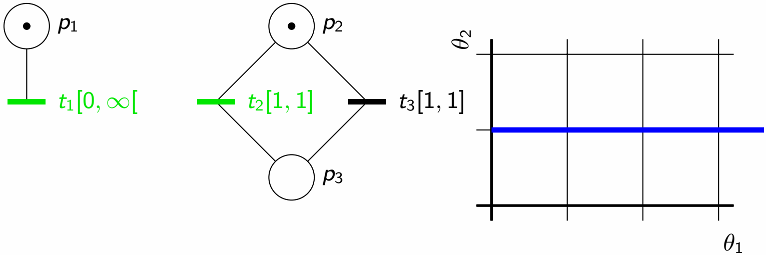
$LSE = min(b_i)$, Fireable transitions: $\{t_j \mid a_j \leq LSE\}$

Computation of the state space - State class graph

$\mathcal{C}_0 = (M_0, D_0)$

Computation of the fireable transitions

Example



$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array} \right. \right) \rightarrow LSE = b_2 = 1 \text{(From the initial class)}$$

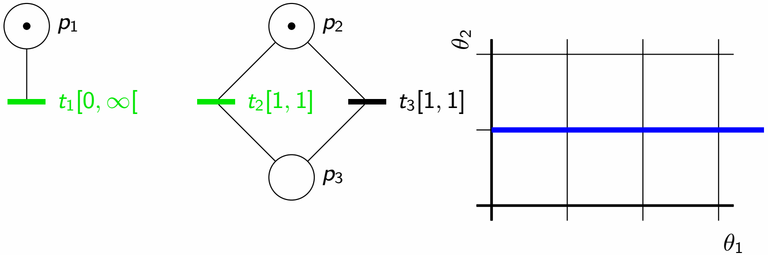
$LSE = min(b_i)$, Fireable transitions: $\{t_j \mid a_j \leq LSE\}$

Computation of the state space - State class graph

$\mathcal{C}_0 = (M_0, D_0)$

Computation of the fireable transitions

Example



$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, \left\{\begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array}\right.\right) \rightarrow LSE = b_2 = 1$(From the initial class) $\rightarrow t_1$ and t_2 are fireable

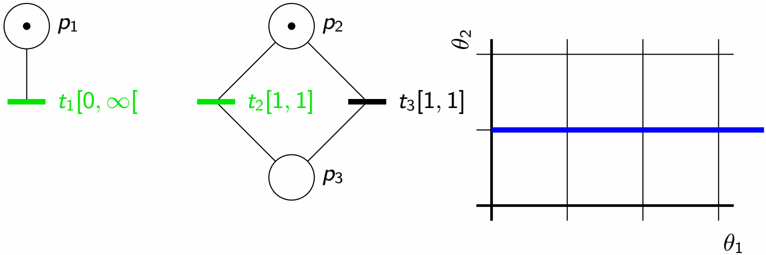
$LSE = min(b_i)$, Fireable transitions: $\{t_j \mid a_j \leq LSE\}$

Computation of the state space - State class graph

$\mathcal{C}_0 = (M_0, D_0)$

Computation of the fireable transitions

Example



$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array} \right. \right) \rightarrow t_1 \text{ and } t_2 \text{ are fireable}$$

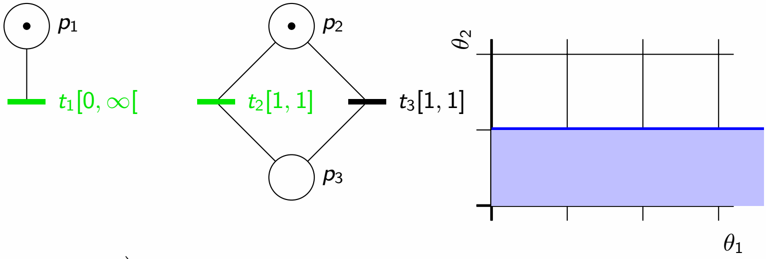
A transition t_i is fireable if, by elapsing time, one can reach $\theta_i = 0$

Computation of the state space - State class graph

$\mathcal{C}_0 = (M_0, D_0)$

Computation of the fireable transitions

Example



$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array} \right. \right) \rightarrow t_1 \text{ and } t_2 \text{ are fireable}$$

A transition t_i is fireable if, by elapsing time, one can reach $\theta_i = 0$

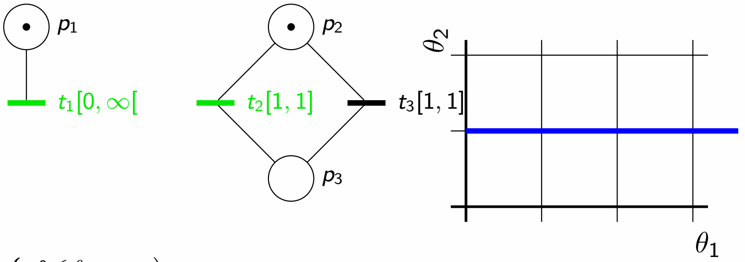
Computation of the state space - State class graph

Firing of transitions $\rightarrow (M_i, D_i)$.

Computation of the state space - State class graph

Firing of transitions $\rightarrow (M_i, D_i)$.

Example



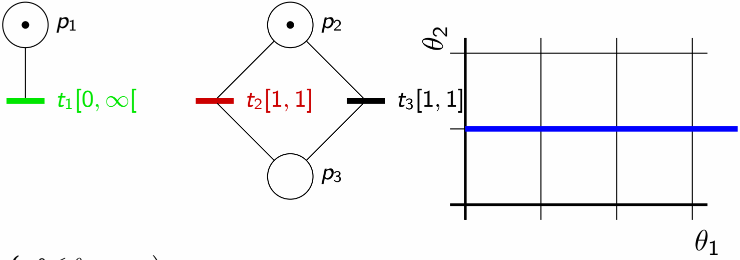
$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{cases} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{cases} \right)$$

$$(M_i, D_i) = (M_0 - {}^\bullet t_i + t_i^\bullet, (D_i = next(D_0, t_i)))$$

Computation of the state space - State class graph

Firing of transitions $\rightarrow (M_i, D_i)$.

Example



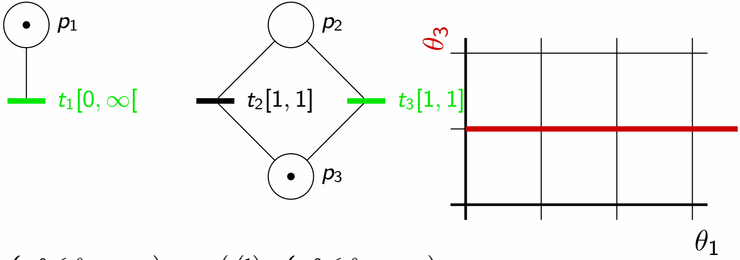
$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array} \right. \right)$$

$$(M_i, D_i) = (M_0 - {}^{\bullet}t_i + t_i^{\bullet}, (D_i = next(D_0, t_i)))$$

Computation of the state space - State class graph

Firing of transitions $\rightarrow (M_i, D_i)$.

Example



$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array} \right. \right) \xrightarrow{t_2} \left(\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_3 \leq 1 \\ \theta_3 - \theta_1 \leq 1 \end{array} \right. \right)$$

$$(M_i, D_i) = (M_0 - {}^\bullet t_i + t_i^\bullet, (D_i = next(D_0, t_i)))$$

State class graph - algorithm

Let $C = (M, D)$, a class and t_f, a fireable transition. The class $C' = (M', D')$, successor of C by the firing of t_f is:

- $M' = M - {}^\bullet t_f + t_f^\bullet$
- $D' = next(D, t_f)$ is computed as follows:
  1. variable change: $\forall j, \theta_j = \theta_f + \theta'_j$;
  2. $\forall t_j \neq t_f$, adding the constraints $\theta'_j \geq 0$;
  3. elimination of the variables associated with the transitions disabled by the firing of t_f (including t_f), by using the Fourier-Motzkin method;
  4. adding the news inequalities of the newly enabled transitions t_k:

     $$\forall t_k \in \uparrow enabled(M, t_f), \alpha(t_k) \leq \theta'_k \leq \beta(t_k).$$

  5. Computation of the canonical form D'^* of D'.

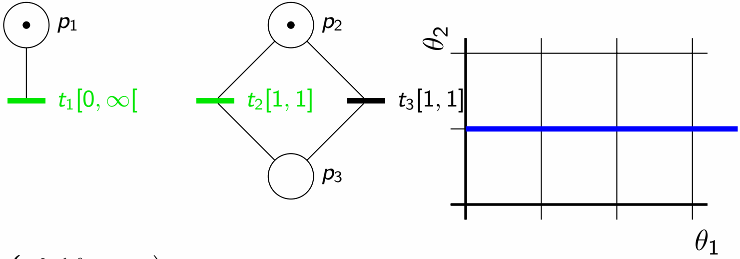
Computation of the state space

Firing of transitions $\rightarrow (M_i, D_i)$.

Computation of the state space

Firing of transitions $\rightarrow (M_i, D_i)$.

Example (Firing of t_1)



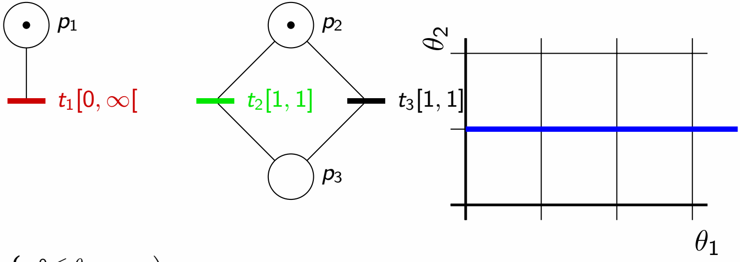
$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array} \right. \right)$$

$$(M_i, D_i) = (M_0 - {}^\bullet t_i + t_i^\bullet, (D_i = next(D_0, t_i)))$$

Computation of the state space

Firing of transitions $\rightarrow (M_i, D_i)$.

Example (Firing of t_1)



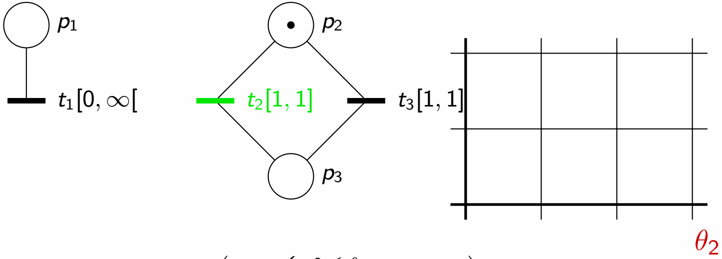
$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array} \right. \right)$$

$$(M_i, D_i) = (M_0 - {}^\bullet t_i + t_i^\bullet, (D_i = next(D_0, t_i)))$$

Computation of the state space

Firing of transitions $\rightarrow (M_i, D_i)$.

Example (Firing of t_1)



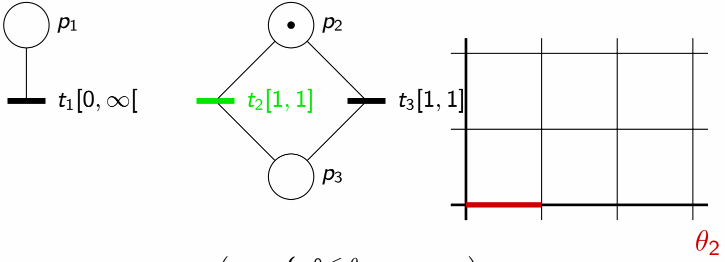
$$\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array} \right. \right) \xrightarrow{t_1} \left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \left\{ \begin{array}{l} 0 \leq \theta_1 \\ \theta_2' \geq 0 \\ 1 \leq \theta_2' + \theta_1 \leq 1 \\ \theta_2' + \theta_1 - \theta_1 \leq 1 \end{array} \right. \right)$$

$$(M_i, D_i) = (M_0 -^\bullet t_i + t_i^\bullet, (D_i = next(D_0, t_i)))$$

Computation of the state space

Firing of transitions $\rightarrow (M_i, D_i)$.

Example (Firing of t_1)



$$\left(\begin{pmatrix}1\\1\\0\end{pmatrix}, \left\{\begin{array}{l} 0 \leq \theta_1 \\ 1 \leq \theta_2 \leq 1 \\ \theta_2 - \theta_1 \leq 1 \end{array}\right.\right) \xrightarrow{t_1} \left(\begin{pmatrix}0\\1\\0\end{pmatrix}, \left\{\begin{array}{l} 0 \leq \theta_1 \\ \theta_2' \geq 0 \\ 1 \leq \theta_2' + \theta_1 \leq 1 \\ \theta_2' + \theta_1 - \theta_1 \leq 1 \end{array}\right.\right) = \left(\begin{pmatrix}0\\1\\0\end{pmatrix}, 0 \leq \theta_2' \leq 1\right)$$

$$(M_i, D_i) = (M_0 - {}^\bullet t_i + t_i^\bullet, (D_i = next(D_0, t_i)))$$

Theorem ([BD91])

The State Class Graph algorithm converges for bounded time Petri Nets:

Theorem ([BD91])

The State Class Graph algorithm converges for bounded time Petri Nets:

Implementations in tools:

- Tina
- Roméo

Computation of the state class graph

Exercise : Ex2-Master.xml

Let the TPN: $P1 \rightarrow T1$, $P2 \rightarrow T2$, $P3 \rightarrow T3$
with $M_o = \{P1, P2, P3\}$ $T1[3,5]$, $T2[7,9]$, $T3[4,6]$

- Compute the initial class C_0
  - Draw the projections of C_0 other 3 planes (θ_1, θ_2), (θ_1, θ_3) and (θ_2, θ_3).
  - Deduce the fireable transitions (Does the elapsing of time makes possible to reach $\theta = 0$?).
- Compute the successor of C_0 by the firing of T_1 (Fourier Motskin) : $C_0 \xrightarrow{T_1} C_1$
  - Draw the polyhedron in the plane (θ_2, θ_3).
  - Is the transition T_2 fireable from C_1 ? (Does the elapsing of time makes possible to reach $\theta_2 = 0$?).
- Same question with T_3 : $C_0 \xrightarrow{T_3} C_2$ (canonical form).

Computation of the state class graph - Convergence

In the state space computation algorithm given in page 34, the convergence criterion is given by $next(\mathcal{Q}, t) \in (Waiting \cup Visited)$

Convergence

For the computation of the state class graph, the convergence can be:

- by equality of classes (the graph preserves the markings and the language)
- by inclusion of the domain of classes (the graph preserves only reachability of markings).

Go back to the previous example. Compute the state class graph obtain:

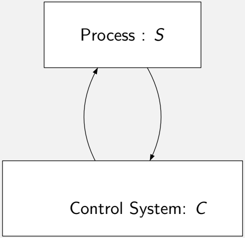
- with convergence by equality
- with convergence by inclusion

Plan

- **Petri nets with time**
- **T-time Petri Nets**
- **Other Semantics**
  - P-time Petri Nets
  - A-time Petri Nets
  - Strong vs Weak Semantics
- **Expressiveness and properties of TPN**
- **State Space of Time Petri Nets**
- **Model-checking of Time Petri Nets**
  - Temporal logics
  - Checking TPN with observers
- **Stopwatch Petri Nets**

Introduction

Design of a real time system



- Closed system : $S \parallel C$
- Model of the process S
- Specification of control properties: φ

Design of a real time system

- Closed system : $S \parallel C$
- Model of the process S
- Specification of control properties: φ

Design of a real time system

- Closed system : $S \parallel C$
- Model of the process S
- Specification of control properties: φ

Model-checking problem:

- Model of the control system C.

$$\text{Does } S \parallel C \models \varphi \quad ? \tag{3}$$

Design of a real time system

- Closed system : $S \parallel C$
- Model of the process S
- Specification of control properties: φ

Model-checking problem:

- Model of the control system C.

$$\text{Does } S \parallel C \models \varphi \quad ? \tag{3}$$

Control problem:

$$\text{Is there } C \text{ such that } S \parallel C \models \varphi \quad ? \tag{4}$$

- If yes : synthesise this controller.

For real time systems:

- Functional specification
- Timed specification

$\Rightarrow$ logical time is not sufficient.

For real time systems:

- Functional specification
- Timed specification

$\Rightarrow$ logical time is not sufficient.

Formal model

Timed extensions of

- Process algebra
- Petri Nets
- Finite Automata

For real time systems:

- Functional specification
- Timed specification

$\Rightarrow$ logical time is not sufficient.

Formal model

Timed extensions of

- Process algebra
- Petri Nets
- Finite Automata

Specification

- observers
- temporal logics (LTL, CTL)
- timed temporal logics (TCTL)

Properties

During the execution of a system,

Safety

nothing bad happens

Properties

During the execution of a system,

Safety

nothing bad happens

Liveness

something good eventually happens

Temporal logic CTL*

The temporal operators

Quantifiers over paths

A : *for all* means 'along All paths' (Inevitably)

E : *for some* means 'along at least (there Exists) one path' (possibly)

Path-specific quantifiers

X : next

U : Until

F : (Finally) eventually

G : (Globally) always

Computation tree logic CTL * = State formulae

State formulae (sf):

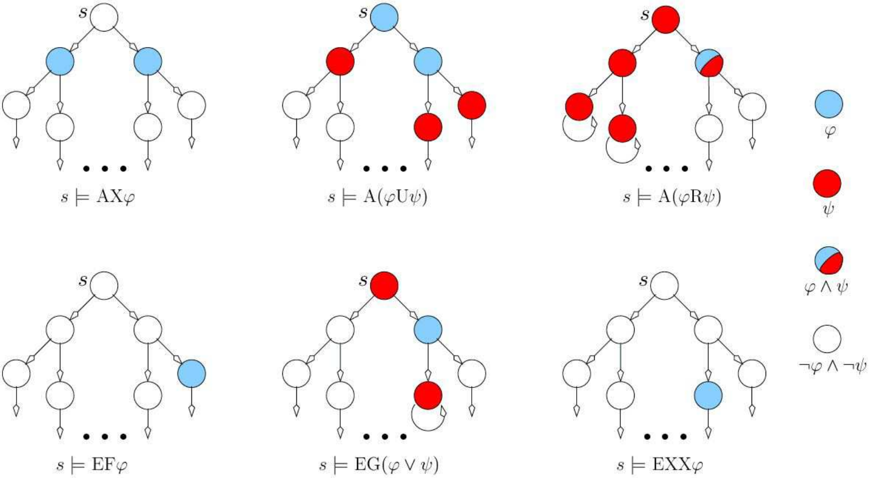
$sf ::= p \mid \neg p \mid \ true \mid false \mid sf \vee sf \mid sf \wedge sf \mid A\ pf \mid E\ pf$

where p ranges over a set of atomic formulas (markings).

Path formulae (pf):

$pf ::= sf \mid sf \vee sf \mid sf \wedge sf \mid pf\ \ U\ pf \mid X\ sf \mid F\ sf \mid G\ sf$

Temporal logic CTL*



Temporal logic CTL*

Other notations:

$$\exists\Diamond\varphi = EF\varphi,$$
$$\forall\Diamond\varphi = AF\varphi,$$
$$\exists\Box\varphi = EG\varphi,$$
$$\forall\Box\varphi = AG\varphi;$$

Deadlock free : $\forall\Box\ \exists X\ \varphi$ with $\varphi = \mathit{true}$ that is to say $AG\ EX\ \mathit{true}$

Temporal logic CTL*

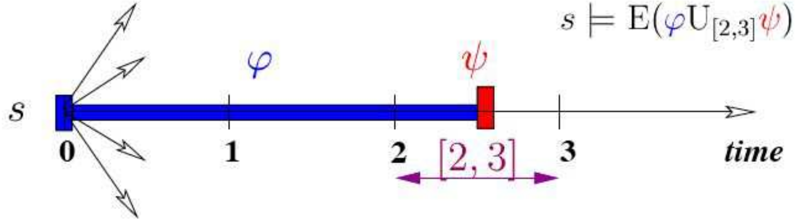
Duality:

$$AF\ \varphi = \neg EG \neg \varphi$$
$$AG\ \varphi = \neg EF \neg \varphi$$
$$AX\ \varphi = \neg EX \neg \varphi$$

F and G quantifiers can be defined from *Until*:

$$AF\ \varphi = A(true U \varphi)$$
$$EF\ \varphi = E(true U \varphi)$$
$$AG\ \varphi = \neg E(true U \neg \varphi)$$
$$EG\ \varphi = \neg A(true U \neg \varphi)$$

Timed Temporal logic TCTL



$\textsc{Gmec}$ = set of legal markings defined as the integer solutions of a convex set

$[\![\mathcal{N}]\!]$ = set of runs of a TPN $\mathcal{N}$

I = time interval

$$
\begin{array}{lll}
(M,v) \models \textsc{Gmec} & \text{iff} & M \models \textsc{Gmec} \\
(M,v) \not\models \textbf{false} & & \\
(M,v) \models \neg\varphi & \text{iff} & (M,v) \not\models \varphi \\
(M,v) \models \varphi \Rightarrow \psi & \text{iff} & (M,v) \not\models \varphi \text{ or } (M,v) \models \psi \\
(M,v) \models \exists\varphi\,\mathcal{U}_I\psi & \text{iff} & \exists\sigma \in [\![\mathcal{N}]\!] \ \textit{such that} \\
& & \begin{cases} (s_0,v_0) = (M,v) \\ \forall i \in [s1..n], \forall d \in [0,d_i), (s_i, v_i + d) \models \varphi \\ \left(\sum_{i=1}^{n} d_i\right) \in I \textit{ and } (s_n,v_n) \models \psi \end{cases} \\
(M,v) \models \forall\varphi\,\mathcal{U}_I\psi & \text{iff} & \forall\sigma \in [\![\mathcal{N}]\!] \ \textit{we have} \\
& & \begin{cases} (s_0,v_0) = (M,v) \\ \forall i \in [1..n], \forall d \in [0,d_i), (s_i, v_i + d) \models \varphi \\ \left(\sum_{i=1}^{n} d_i\right) \in I \textit{ and } (s_n,v_n) \models \psi \end{cases}
\end{array}
$$

Timed Temporal logic TCTL

Notations and shorthands :

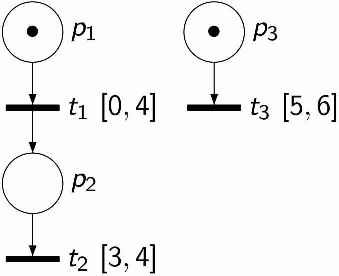
$$\exists\Diamond_I\phi = \exists \textbf{true}\, \mathcal{U}_I \phi,$$
$$\forall\Diamond_I\phi = \forall \textbf{true}\, \mathcal{U}_I \phi,$$
$$\exists\Box_I\phi = \neg\forall\Diamond_I\neg\phi,$$
$$\forall\Box_I\phi = \neg\exists\Diamond_I\neg\phi;$$

$$(\varphi \leadsto_I \psi) = \forall\Box(\varphi \Rightarrow \forall\Diamond_I\psi),$$

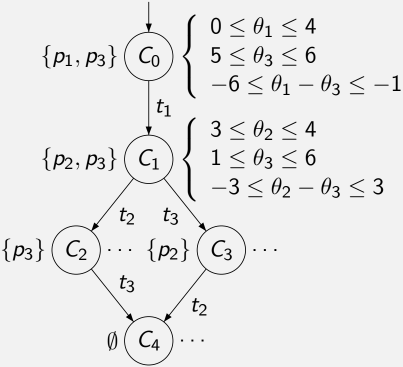
Abstraction and verification...

Example : state class graph [BD91]

Example (Net of L. Gallon)



State class graph



Problème

Find an abstraction for a given property

Problème

Find an abstraction for a given property

Non quantitative properties :

- Property over traces (LTL) : state class graph [BD91] or zone graph.
- Branching property (CTL) : atomic state class graph [BV03]

Problème

Find an abstraction for a given property

Non quantitative properties :

- Property over traces (LTL) : state class graph [BD91] or zone graph.
- Branching property (CTL) : atomic state class graph [BV03]

Quantitative Properties

- Observer and reachability
- For a subset of TCTL (on the fly computation) : state class graph or zone graph.

Problème

Find an abstraction for a given property

Non quantitative properties :

- Property over traces (LTL) : state class graph [BD91] or zone graph.
- Branching property (CTL) : atomic state class graph [BV03]

Quantitative Properties

- Observer and reachability
- For a subset of TCTL (on the fly computation) : state class graph or zone graph.

Checking TPN with observers

Observer

- non intrusive
  - Arc (post) from a transition of the net to a place of the observer
  - Read arc or inhibitor arc from a place of the net to a transition of the observer
  - Reset Arc from a place of the observer to a transition of the net
- turns the verification of a particular property into a marking reachability problem

Example : let a Petri Net with a transition t. Write an observer for the followings properties : Between two successive firings of the transition t, there is:

- always more than 10 time units
- never more than 10 time units

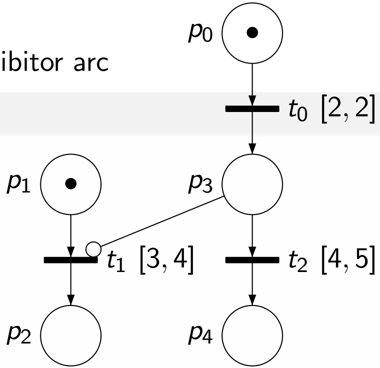
Exercise : ex1.xml

- $EF[0,40](M(P5) - M(P2) > 0)$
- $EF[0,20](M(P5) - M(P2) > 0)$
- $AG[0,20](M(P1) + M(P2) > 0)$
- $AG[0,30](M(P1) + M(P2) > 0)$
- $EG[0,30](M(P1) + M(P2) > 0)$
- $(M(P2) > 0) \rightarrow [0,40](M(P5) > 0)$
- $(M(P2) > 0) \rightarrow [0,35](M(P5) > 0)$
- $(M(P2) > 0) \rightarrow [0,60](M(P3) - M(P4) > 0)$
- Write an observer to check the properties :
  - $(M(P2) > 0) \rightarrow [0,45[(M(P6) > 0)$ or more precisely (firing of T0)$\rightarrow$ $[0,45[$(firing of T3)
  - (firing of T0)$\rightarrow]10,45[$(firing of T3)
  - idem considering that there may be several firings of T0 before a firing of T3 (we consider for the property only the last firing of T0)

Stopwatch Petri Nets

Petri net with timed inhibitor arc

Example (SwPN)

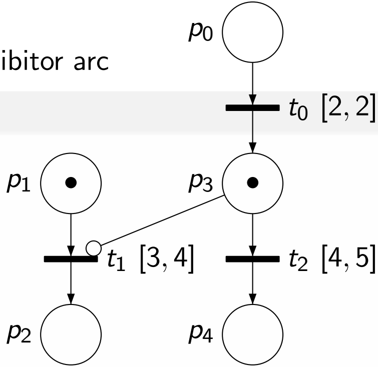


Is it possible to fire t_1 before t_2 ?

Stopwatch Petri Nets

Petri net with timed inhibitor arc

Example (SwPN)

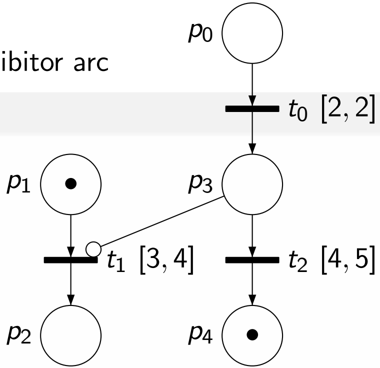


Is it possible to fire t_1 before t_2 ?

Stopwatch Petri Nets

Petri net with timed inhibitor arc

Example (SwPN)

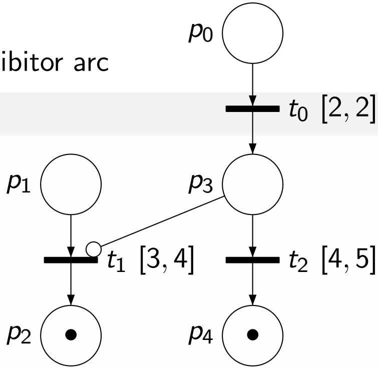


Is it possible to fire t_1 before t_2 ?

Stopwatch Petri Nets

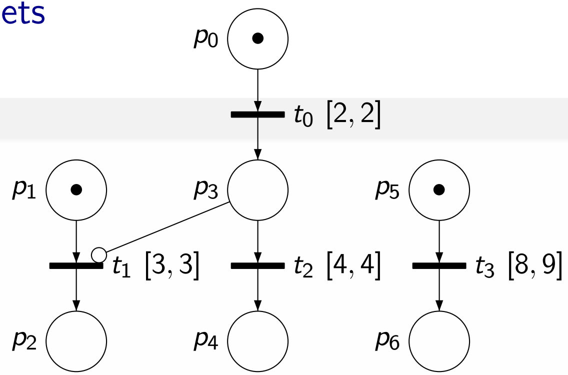
Petri net with timed inhibitor arc

Example (SwPN)



Stopwatch Petri Nets

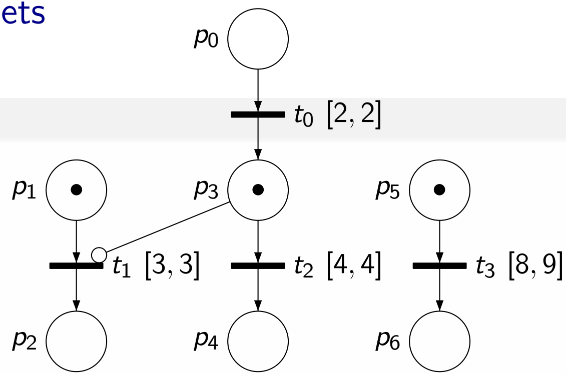
Example (SwPN)



Is it possible to fire t_3 before t_1 ?

Stopwatch Petri Nets

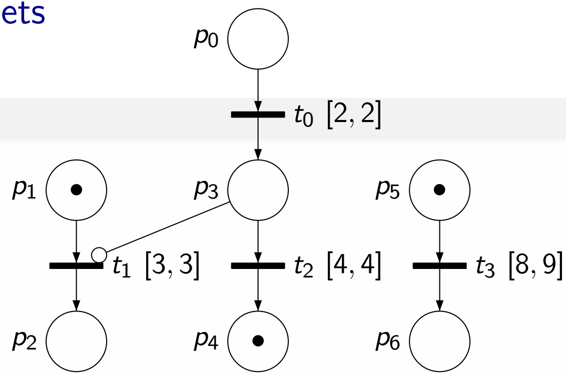
Example (SwPN)



Is it possible to fire t_3 before t_1 ?

Stopwatch Petri Nets

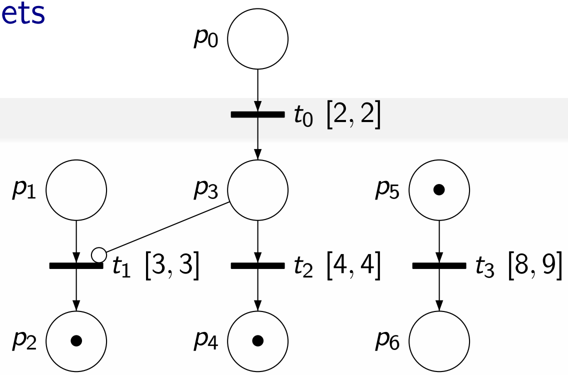
Example (SwPN)



Is it possible to fire t_3 before t_1 ?

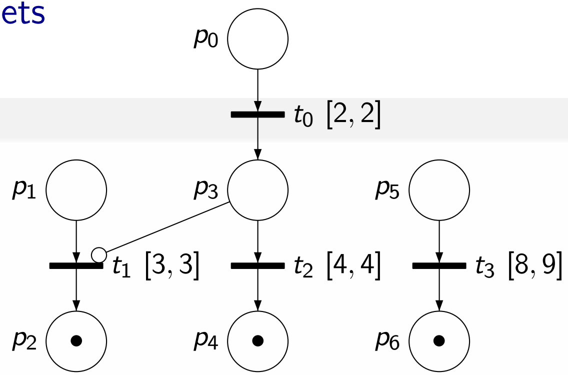
Stopwatch Petri Nets

Example (SwPN)



Stopwatch Petri Nets

Example (SwPN)



Some books

Penczek Wojciech and Polrola Agata
Advances in Verification of Time Petri Nets and Timed Automata
Editor: Springer Verlag, 2006.

Claude Jard and Olivier H. Roux
Communicating Embedded Systems - Software and Design
Editor: ISTE Publishing - John Wiley, 2009.

Claude Girault and Rüdiger Valk
Petri Nets for Systems Engineering
Editor: Springer Verlag, 2003

References

R. Alur, C. Courcoubetis, and D.L. Dill.
Model-checking for real-time systems.
In *5th IEEE Symposium on Logic in Computer*, pages 414–425. IEEE Computer Society Press, june 1990.

P.A. Abdulla and A. Nylén.
Timed Petri nets and BQOs.
In *ICATPN'01*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–72, Newcastle, United Kingdom, june 2001. Springer-Verlag.

Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux.
The expressive power of time Petri nets.
Theoretical Computer Science (TCS), 474:1–20, 2013.

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, 1991.

Hanifa Boucheneb, Guillaume Gardey, and Olivier H. Roux.
TCTL model checking of time Petri nets.
Journal of Logic and Computation, 19(6):1509–1540, December 2009.

Marc Boyer and Olivier H. Roux.
On the compared expressiveness of arc, place and transition time Petri nets.
Fundamenta Informaticae, 88(3):225–249, 2008.

B. Berthomieu and F. Vernadat.
State class constructions for branching analysis of time Petri nets.
In *TACAS'03*, pages 442–457. Springer–Verlag, Apr 2003.

Franck Cassez and Olivier (H.) Roux.
Structural translation from Time Petri Nets to Timed Automata – Model-Checking Time Petri Nets via Timed Automata.
The journal of Systems and Software, 79(10):1456–1468, 2006.

D. de Frutos Escrig, V. Valero Ruiz, and O. Marroquín Alonso.
Decidability of properties of timed-arc Petri nets.
In *ICATPN'00*, volume 1825 of *Lecture Notes in Computer Science*, pages 187–206, Aarhus, Denmark, june 2000.

H.M. Hanisch.
Analysis of place/transition nets with timed-arcs and its application to batch process control.
In *14th International Conference on Application and Theory of Petri Nets (ICATPN'93)*, volume 691 of *LNCS*, pages 282–299, 1993.

N.D. Jones, L.H. Landweber, and Y.E. Lien.
Complexity of some problems in Petri nets.
Theoretical Computer Science 4, pages 277–299, 1977.

W. Khansa, J.-P. Denat, and S. Collart-Dutilleul.
P-Time Petri Nets for manufacturing systems.
In *International Workshop on Discrete Event Systems, WODES'96*, pages 94–102, Edinburgh (U.K.), august 1996.

P.M. Merlin.
A study of the recoverability of computing systems.
PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA, 1974.

C. Ramchandani.
Analysis of asynchronous concurrent systems by timed Petri nets.
PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974.
Project MAC Report MAC-TR-120.