On Time Petri Nets and Scheduling

Didier Lime

IRCCyN / École Centrale de Nantes

5 septembre 2007

Plan I

Plan I

Introduction

Time Petri Nets with Stopwatches (and more)

Abstractions for Scheduling-TPNs

Verifying properties

Conclusion and Future Work

Introduction

- Results in this talk owe credit to Olivier H. Roux, Bernard Berthomieu, Morgan Magnin and François Vernadat;

Introduction

- Results in this talk owe credit to Olivier H. Roux, Bernard Berthomieu, Morgan Magnin and François Vernadat;
- Verification of real-time systems is a **tough problem**: time, uncertainty, complex synchronizations, preemptive scheduler, distribution...

Introduction

- Results in this talk owe credit to Olivier H. Roux, Bernard Berthomieu, Morgan Magnin and François Vernadat;
- Verification of real-time systems is a **tough problem**: time, uncertainty, complex synchronizations, preemptive scheduler, distribution...
- A lot of work has been devoted to **analytical results** of schedulability, giving worst-case response-time for tasks;

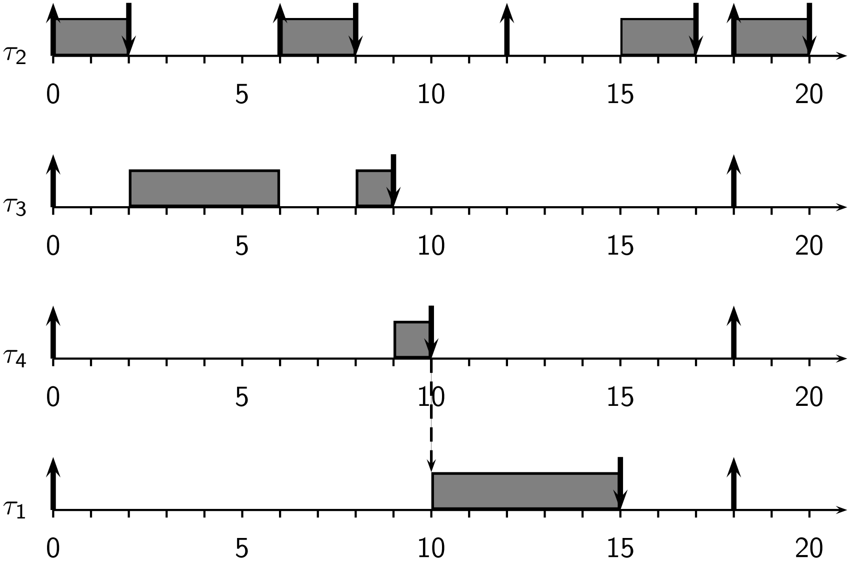
Introduction

- Results in this talk owe credit to Olivier H. Roux, Bernard Berthomieu, Morgan Magnin and François Vernadat;
- Verification of real-time systems is a **tough problem**: time, uncertainty, complex synchronizations, preemptive scheduler, distribution...
- A lot of work has been devoted to **analytical results** of schedulability, giving worst-case response-time for tasks;
- They are very good with simple settings;

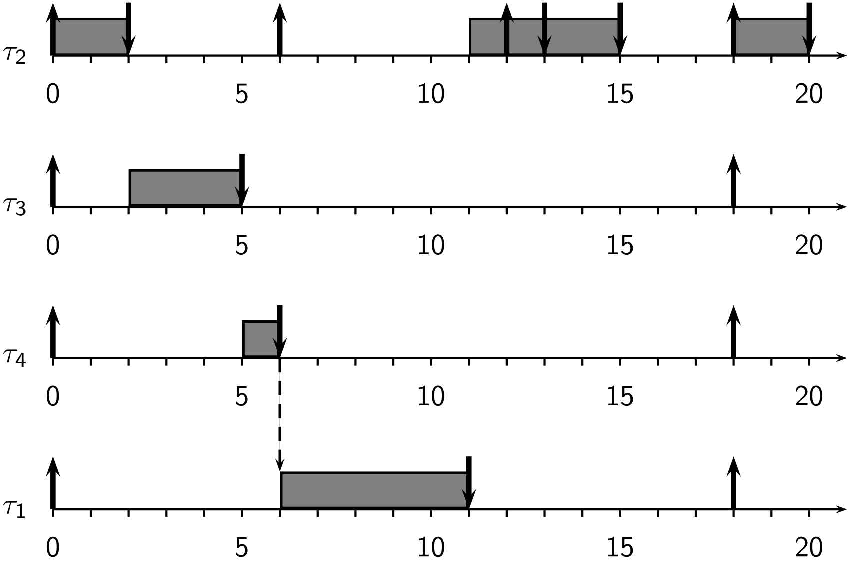
Introduction

- Results in this talk owe credit to Olivier H. Roux, Bernard Berthomieu, Morgan Magnin and François Vernadat;
- Verification of real-time systems is a **tough problem**: time, uncertainty, complex synchronizations, preemptive scheduler, distribution. . .
- A lot of work has been devoted to **analytical results** of schedulability, giving worst-case response-time for tasks;
- They are very good with simple settings;
- They are not so good with precedences, timing uncertainties, distribution, . . .

A well-known paradox

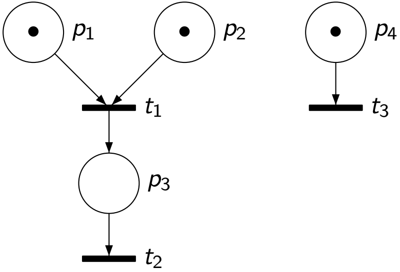


A well-known paradox

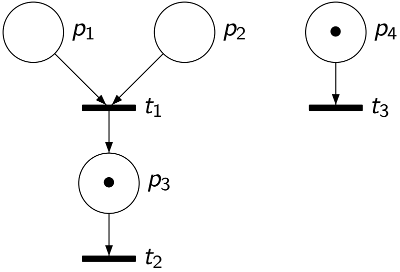


Plan I

Petri Nets

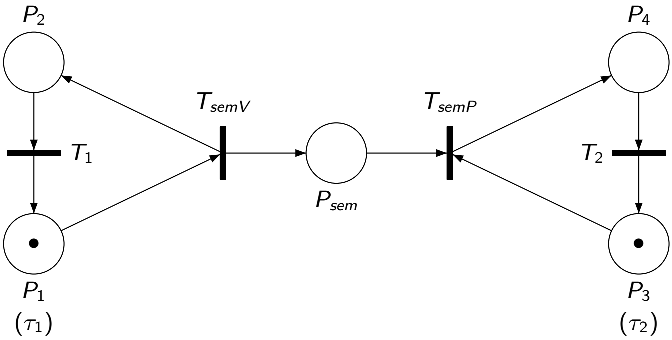


Petri Nets

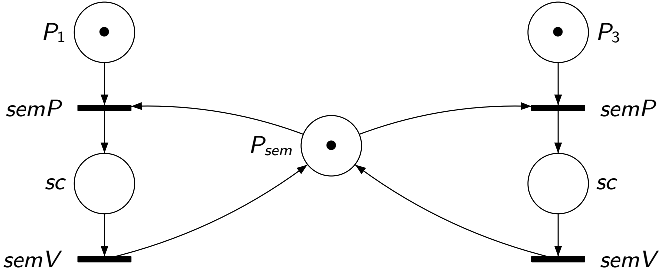


$$\boxed{\text{if } M \geq {}^\bullet t \text{ then } M' = M - {}^\bullet t + t^\bullet}$$

Petri Net Example: Basic Semaphore Synchronization



Petri Net Example: Mutual Exclusion



Petri Net Example: Peterson's Algorithm

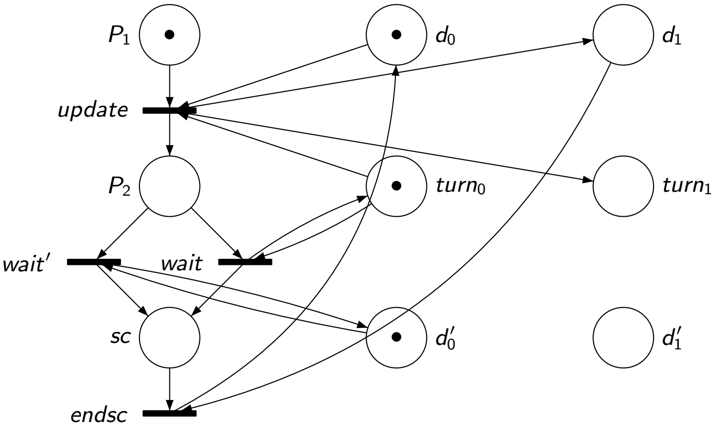
P:

```
d ← false
loop
    <non critical section>
    d ← true
    turn ← 1
    wait(¬d′∨ turn= 0)
    <critical section>
    d ← false
end_loop
```

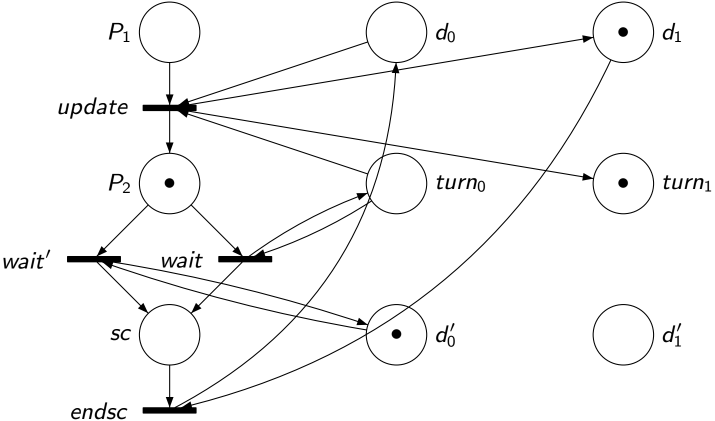
P':

```
d′ ← false
loop
    <non critical section>
    d′ ← true
    turn ← 0
    wait(¬d∨ turn= 1)
    <critical section>
    d′ ← false
end_loop
```

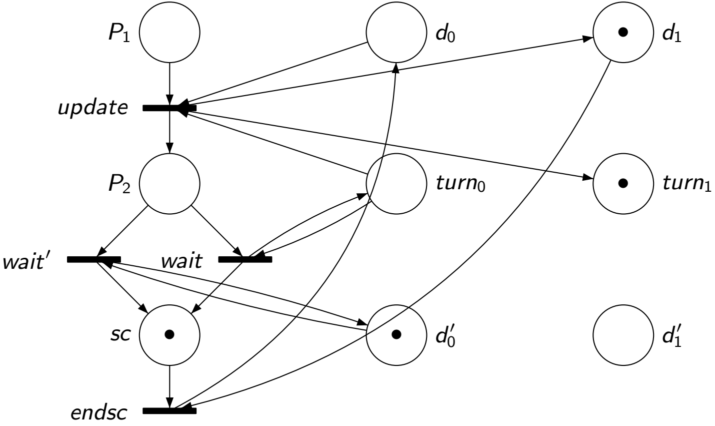
Petri Net Example: Peterson's Algorithm



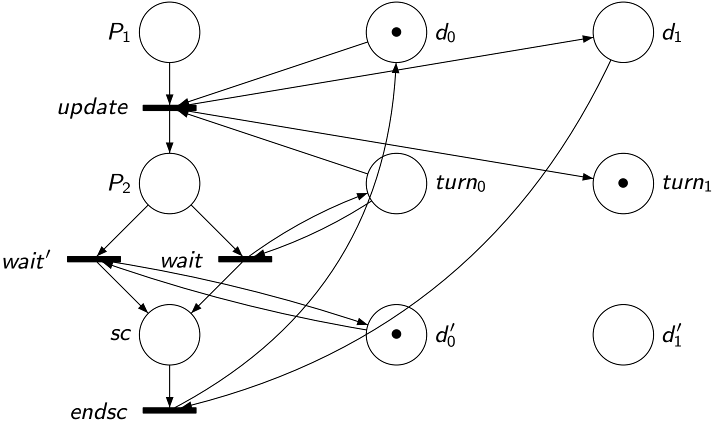
Petri Net Example: Peterson's Algorithm



Petri Net Example: Peterson's Algorithm



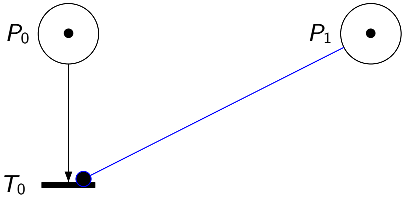
Petri Net Example: Peterson's Algorithm



Read Arcs

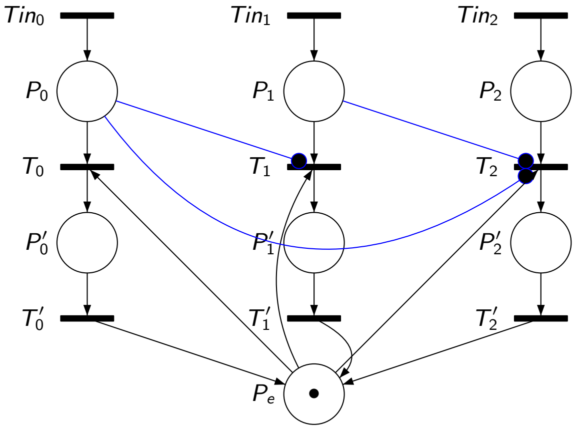
$$\boxed{\text{if } M \geq {}^\bullet t \text{ and } M \geq {}^\diamond t \text{ then } M' = M - {}^\bullet t + t^\bullet}$$

Inhibitor Arcs

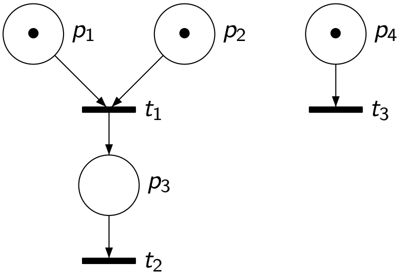


$$\boxed{\text{if } M \geq {}^{\bullet}t \text{ and } M < {}^{\circ}t \text{ then } M' = M - {}^{\bullet}t + t^{\bullet}}$$

Petri Net Example: Fixed Priority Scheduling (non-preemptive)

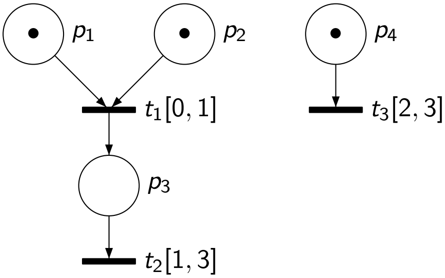


Time Petri Nets



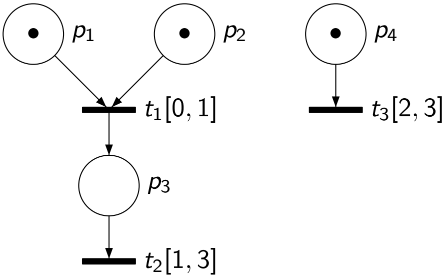
A Petri Net

Time Petri Nets



A **Time** Petri Net

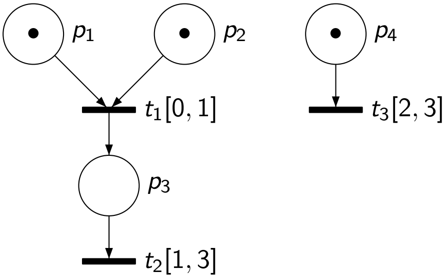
Time Petri Nets



A **Time** Petri Net

$$\left(M_0, \begin{array}{l} t_1 = 0, \\ t_3 = 0 \end{array} \right)$$

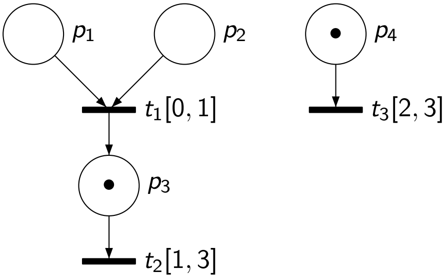
Time Petri Nets



A **Time** Petri Net

$$\left(M_0, \begin{array}{l} t_1 = 0, \\ t_3 = 0 \end{array}\right) \xrightarrow{0.62} \left(M_0, \begin{array}{l} t_1 = \mathbf{0.62}, \\ t_3 = \mathbf{0.62} \end{array}\right)$$

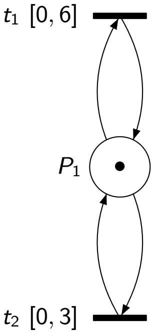
Time Petri Nets



A **Time** Petri Net

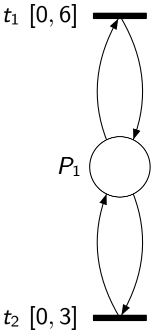
$$\left(M_0,\ \begin{array}{l} t_1 = 0, \\ t_3 = 0 \end{array}\right) \xrightarrow{0.62} \left(M_0,\ \begin{array}{l} t_1 = \mathbf{0.62}, \\ t_3 = \mathbf{0.62} \end{array}\right) \xrightarrow{t_1} \left(\mathbf{M_1},\ \begin{array}{l} t_1 = 0.62, \\ \mathbf{t_2} = \mathbf{0} \end{array}\right)$$

About newly enabled transitions



Fire t_1

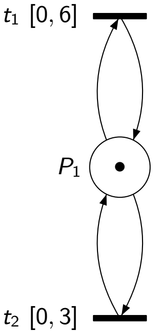
About newly enabled transitions



Fire t_1

t_1 and t_2 are not enabled by $M - {}^\bullet t_1$

About newly enabled transitions

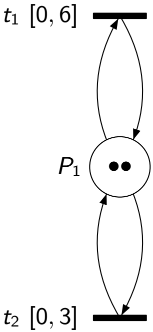


Fire t_1

t_1 and t_2 are not enabled by $M - {}^\bullet t_1$

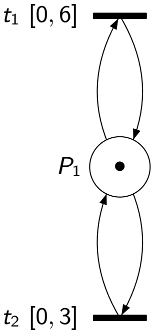
t_1 and t_2 are **newly** enabled

About newly enabled transitions



Fire t_1

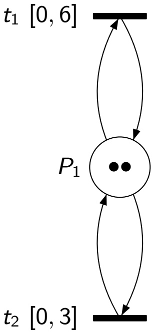
About newly enabled transitions



Fire t_1

t_1 and t_2 are enabled by $M - {}^\bullet t_1$ but t_1 is the fired transition

About newly enabled transitions

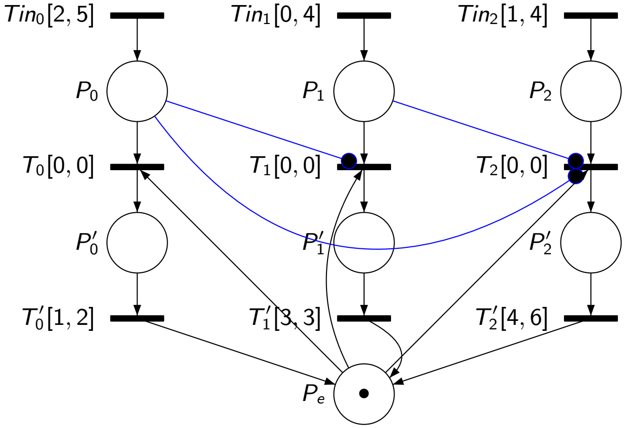


Fire t_1

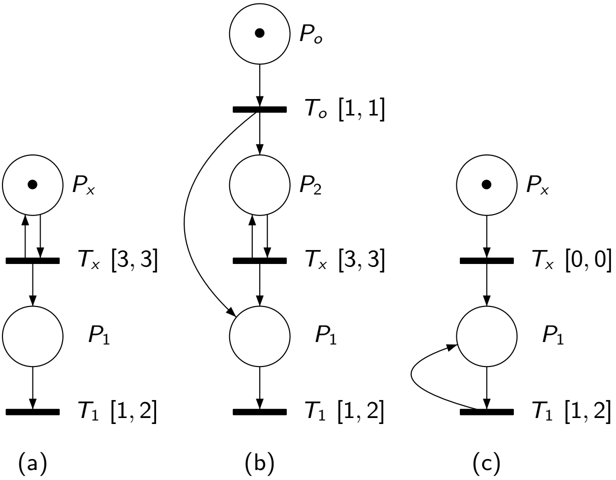
t_1 and t_2 are enabled by $M - {}^\bullet t_1$ but t_1 is the fired transition

t_2 **remains** enabled, t_1 is **newly** enabled

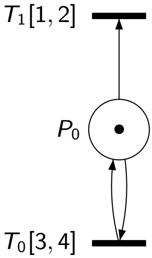
Time Petri Net Example: Fixed Priority Scheduling (non-preemptive)



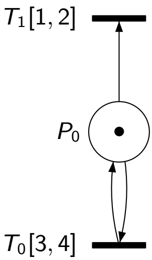
TPN Example: Task Activation



Read Arcs and Time



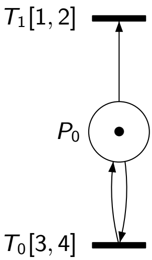
Read Arcs and Time



Read should not be destructive:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array} \right)$$

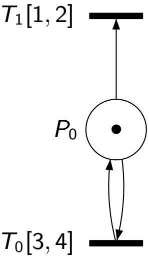
Read Arcs and Time



Read should not be destructive:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right)$$

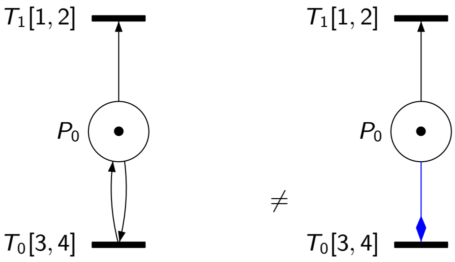
Read Arcs and Time



Read should not be destructive:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right) \xrightarrow{T_0} \left(M_0, \begin{array}{l} T_0 = 0, \\ \mathbf{T_1} = \mathbf{0} \end{array}\right)$$

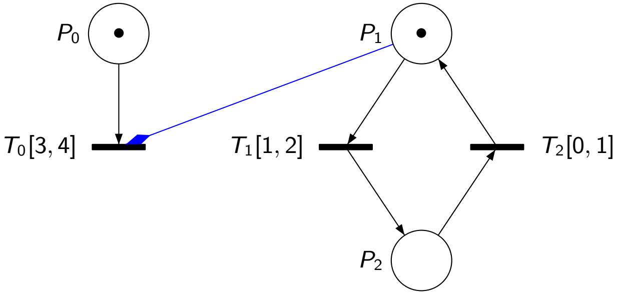
Read Arcs and Time



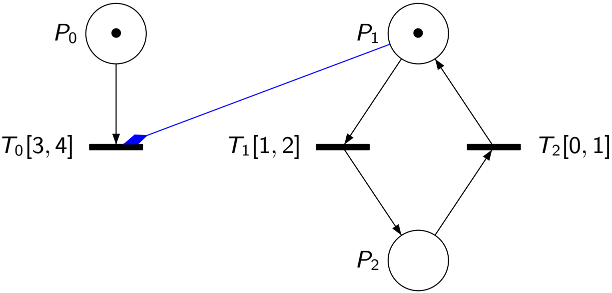
Read should not be destructive:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right) \xrightarrow{T_0} \left(M_0, \begin{array}{l} T_0 = 0, \\ \mathbf{T_1} = \mathbf{0} \end{array}\right)$$

From Read Arcs To Activator Arcs



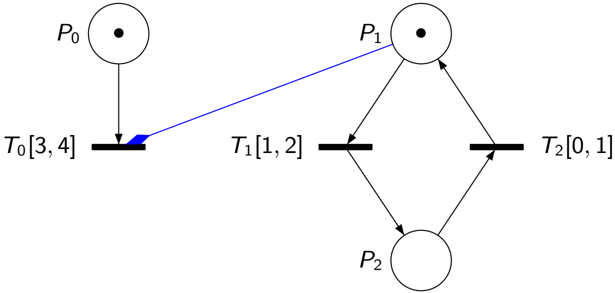
From Read Arcs To Activator Arcs



Memory / No Memory:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array} \right)$$

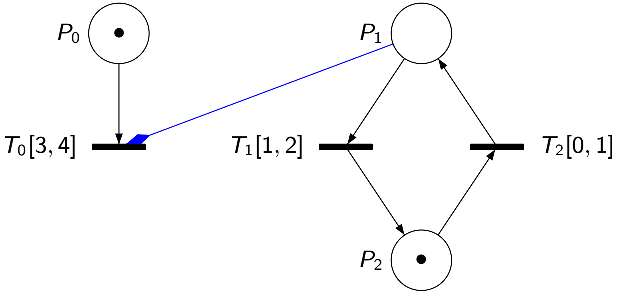
From Read Arcs To Activator Arcs



Memory / No Memory:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right)$$

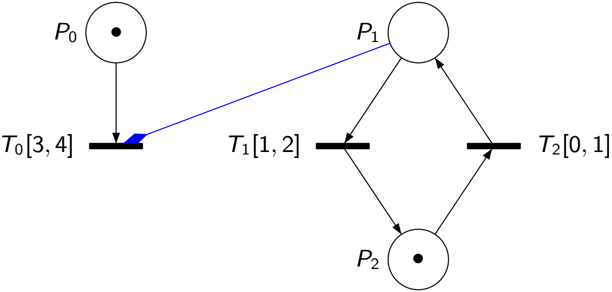
From Read Arcs To Activator Arcs



Memory / No Memory:

$$\left(M_0,\ \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0,\ \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right) \xrightarrow{T_1} (M_1, T_2 = 0)$$

From Read Arcs To Activator Arcs

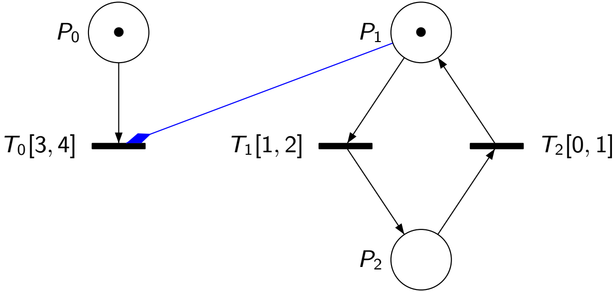


Memory / No Memory:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right) \xrightarrow{T_1} (M_1, T_2 = 0)$$

$$\xrightarrow{0.2} (M_1, T_2 = 0.2)$$

From Read Arcs To Activator Arcs

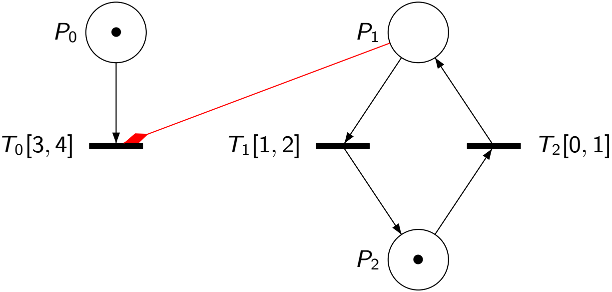


Memory / No Memory:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right) \xrightarrow{T_1} (M_1, T_2 = 0)$$

$$\xrightarrow{0.2} (M_1, T_2 = 0.2) \xrightarrow{T_2} \left(M_0, \begin{array}{l} T_0 = \mathbf{0}, \\ T_1 = 0 \end{array}\right)$$

From Read Arcs To Activator Arcs

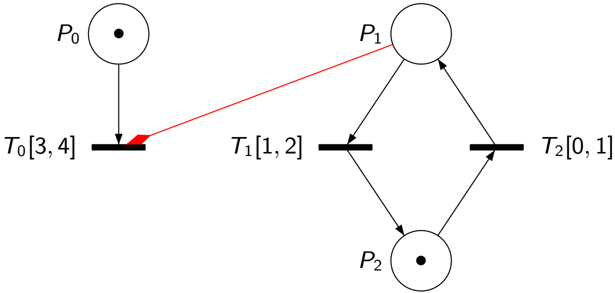


Memory / No Memory:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right) \xrightarrow{T_1} \left(M_1, \begin{array}{l} T_0 = 1.62, \\ T_2 = 0 \end{array}\right)$$

$$\xrightarrow{0.2} (M_1, T_2 = 0.2) \xrightarrow{T_2} \left(M_0, \begin{array}{l} T_0 = \mathbf{0}, \\ T_1 = 0 \end{array}\right) \xrightarrow{0.31} \left(M_0, \begin{array}{l} T_0 = 0.31, \\ T_1 = 0.31 \end{array}\right)$$

From Read Arcs To Activator Arcs

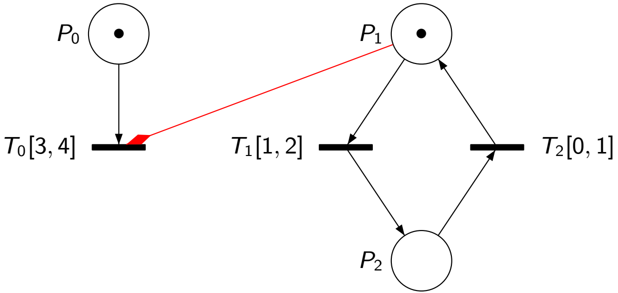


Memory / No Memory:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right) \xrightarrow{T_1} \left(M_1, \begin{array}{l} T_0 = 1.62, \\ T_2 = 0 \end{array}\right)$$

$$\xrightarrow{0.2} \left(M_1, \begin{array}{l} T_0 = 1.62, \\ T_2 = 0.2 \end{array}\right) \xrightarrow{T_2} \left(M_0, \begin{array}{l} T_0 = \mathbf{0}, \\ T_1 = 0 \end{array}\right) \xrightarrow{0.31} \left(M_0, \begin{array}{l} T_0 = 0.31, \\ T_1 = 0.31 \end{array}\right)$$

From Read Arcs To Activator Arcs

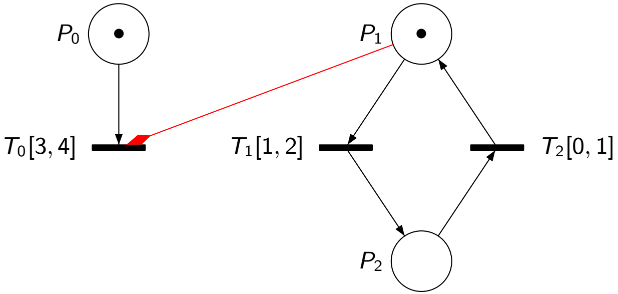


Memory / No Memory:

$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right) \xrightarrow{T_1} \left(M_1, \begin{array}{l} T_0 = 1.62, \\ T_2 = 0 \end{array}\right)$$

$$\xrightarrow{0.2} \left(M_1, \begin{array}{l} T_0 = 1.62, \\ T_2 = 0.2 \end{array}\right) \xrightarrow{T_2} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 0 \end{array}\right) \xrightarrow{0.31} \left(M_0, \begin{array}{l} T_0 = 0.31, \\ T_1 = 0.31 \end{array}\right)$$

From Read Arcs To Activator Arcs

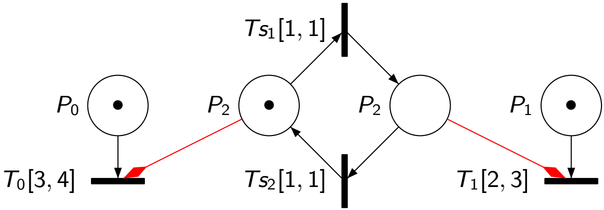


Memory / No Memory:

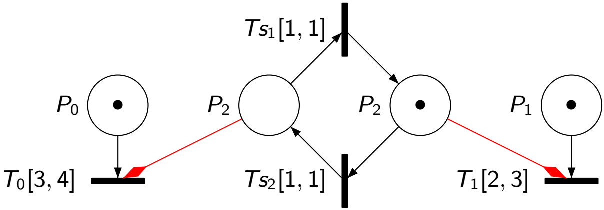
$$\left(M_0, \begin{array}{l} T_0 = 0, \\ T_1 = 0 \end{array}\right) \xrightarrow{1.62} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 1.62 \end{array}\right) \xrightarrow{T_1} \left(M_1, \begin{array}{l} T_0 = 1.62, \\ T_2 = 0 \end{array}\right)$$

$$\xrightarrow{0.2} \left(M_1, \begin{array}{l} T_0 = 1.62, \\ T_2 = 0.2 \end{array}\right) \xrightarrow{T_2} \left(M_0, \begin{array}{l} T_0 = 1.62, \\ T_1 = 0 \end{array}\right) \xrightarrow{0.31} \left(M_0, \begin{array}{l} T_0 = 1.93, \\ T_1 = 0.31 \end{array}\right)$$

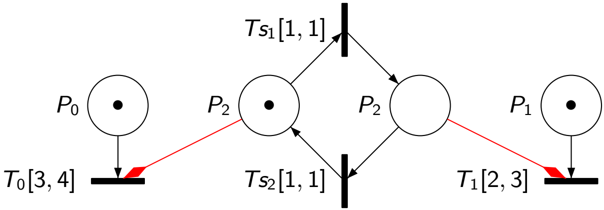
TPN with Stopwatches Example: Round-Robin Scheduling



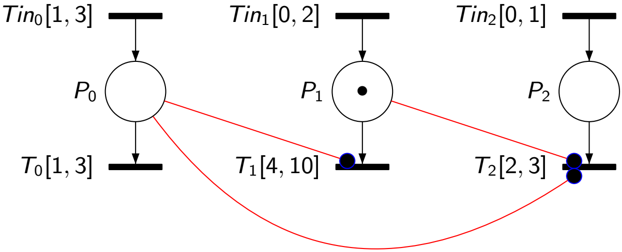
TPN with Stopwatches Example: Round-Robin Scheduling



TPN with Stopwatches Example: Round-Robin Scheduling

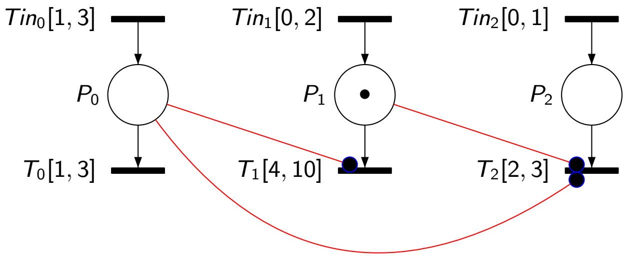


Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)



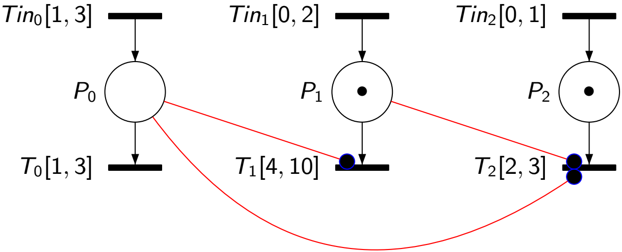
$$T_0 =? \quad T_1 = 0 \quad T_2 =?$$

Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)



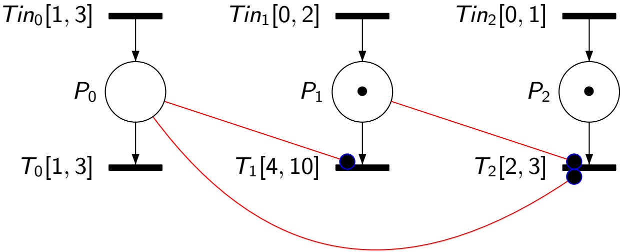
$T_0 = ?\quad T_1 = \mathbf{0.22}\quad T_2 = ?$

Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)



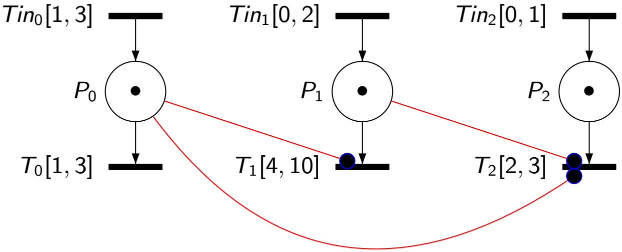
$$T_0 = ? \quad T_1 = 0.22 \quad T_2 = \mathbf{0}$$

Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)



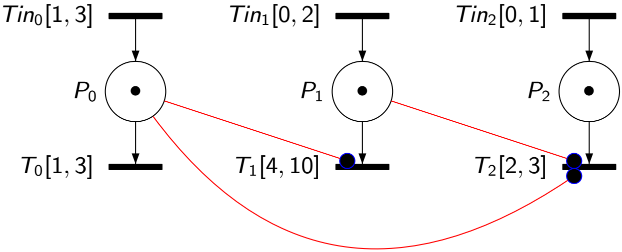
$$T_0 = ? \quad T_1 = \mathbf{1.6} \quad T_2 = 0$$

Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)



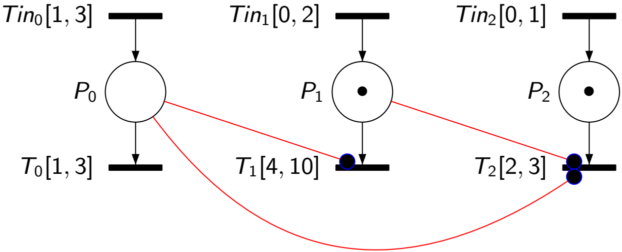
$$T_0 = \mathbf{0} \quad T_1 = 1.6 \quad T_2 = 0$$

Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)



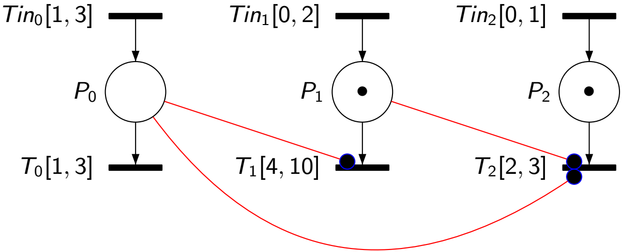
$T_0 = \mathbf{1.5} \quad T_1 = 1.6 \quad T_2 = 0$

Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)



$$T_0 =? \quad T_1 = 1.6 \quad T_2 = 0$$

Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)



$T_0 = ?\quad T_1 = \mathbf{1.8}\quad T_2 = 0$

The price of expressiveness

The **reachability** problem:

	General case	Bounded
Petri Nets	decidable	decidable
Petri Nets w/ Inhibitor	undecidable[8]	decidable
Time Petri Nets	undecidable [9]	decidable [1]
Stopwatch Time Petri Nets	undecidable [2]	undecidable[2]

Scheduling Time Petri Nets

- A model in the class of **Time Petri Nets w/ Stopwatches**;

Scheduling Time Petri Nets

- A model in the class of **Time Petri Nets w/ Stopwatches**;
- Dedicated to the modelling of **preemptive scheduling** policies;

Scheduling Time Petri Nets

- A model in the class of **Time Petri Nets w/ Stopwatches**;
- Dedicated to the modelling of **preemptive scheduling** policies;
- Places and transitions are associated with **tasks**;

Scheduling Time Petri Nets

- A model in the class of **Time Petri Nets w/ Stopwatches**;
- Dedicated to the modelling of **preemptive scheduling** policies;
- Places and transitions are associated with **tasks**;
- Tasks are given a **processor** and **priority** (or deadline or . . .);

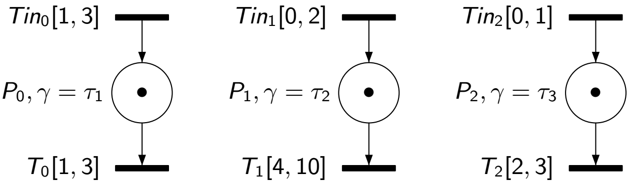
Scheduling Time Petri Nets

- A model in the class of **Time Petri Nets w/ Stopwatches**;
- Dedicated to the modelling of **preemptive scheduling** policies;
- Places and transitions are associated with **tasks**;
- Tasks are given a **processor** and **priority** (or deadline or . . .);
- Processors are given a **scheduling policy**.

Scheduling Time Petri Nets

- A model in the class of **Time Petri Nets w/ Stopwatches**;
- Dedicated to the modelling of **preemptive scheduling** policies;
- Places and transitions are associated with **tasks**;
- Tasks are given a **processor** and **priority** (or deadline or . . .);
- Processors are given a **scheduling policy**.
- With the state of the net of **progress rate for each transition** is computed at each change of marking.

Example: Fixed Priority Scehduling (Preemptive)

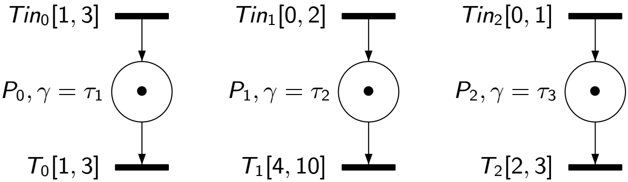


$Proc(\tau_1) = 1$
$Prio(\tau_1) = 3$

$Proc(\tau_2) = 1$
$Prio(\tau_2) = 2$

$Proc(\tau_3) = 1$
$Prio(\tau_3) = 1$

Example: Fixed Priority Scehduling (Preemptive)



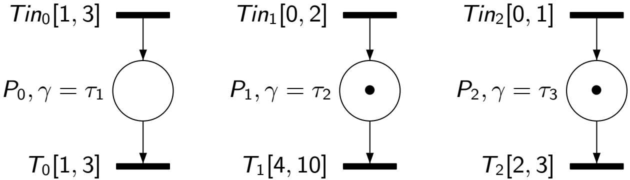
$Proc(\tau_1) = 1$ $Prio(\tau_1) = 3$

$Proc(\tau_2) = 1$ $Prio(\tau_2) = 2$

$Proc(\tau_3) = 1$ $Prio(\tau_3) = 1$

$$Flow(T_0) = \mathbf{1}, Flow(T_1) = 0, Flow(T_2) = 0$$

Example: Fixed Priority Scehduling (Preemptive)



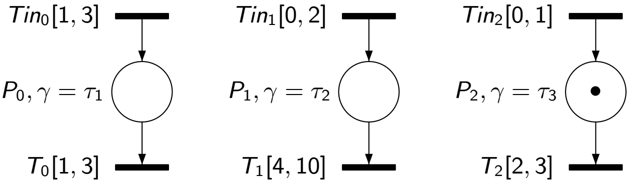
$Proc(\tau_1) = 1$
$Prio(\tau_1) = 3$

$Proc(\tau_2) = 1$
$Prio(\tau_2) = 2$

$Proc(\tau_3) = 1$
$Prio(\tau_3) = 1$

$$Flow(T_0) =?, Flow(T_1) = \mathbf{1}, Flow(T_2) = 0$$

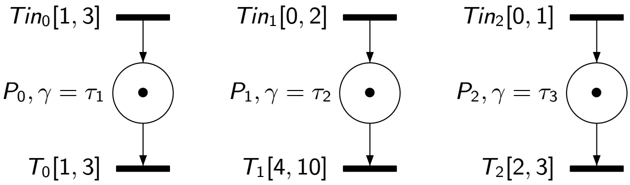
Example: Fixed Priority Scehduling (Preemptive)



$Proc(\tau_1) = 1$	$Proc(\tau_2) = 1$	$Proc(\tau_3) = 1$
$Prio(\tau_1) = 3$	$Prio(\tau_2) = 2$	$Prio(\tau_3) = 1$

$$Flow(T_0) = ?, Flow(T_1) = ?, Flow(T_2) = \mathbf{1}$$

Example: Round-Robin

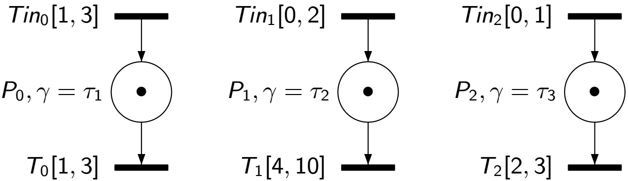


$Proc(\tau_1) = 1$
$Prio(\tau_1) = 1$

$Proc(\tau_2) = 1$
$Prio(\tau_2) = 1$

$Proc(\tau_3) = 1$
$Prio(\tau_3) = 1$

Example: Round-Robin

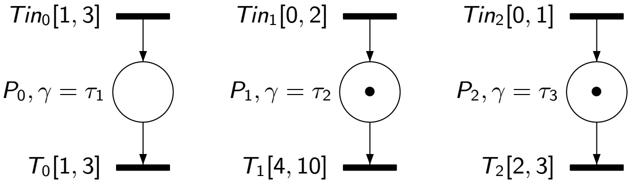


$Proc(\tau_1) = 1$ $Proc(\tau_2) = 1$ $Proc(\tau_3) = 1$

$Prio(\tau_1) = 1$ $Prio(\tau_2) = 1$ $Prio(\tau_3) = 1$

$$Flow(T_0) = \mathbf{\frac{1}{3}}, Flow(T_1) = \mathbf{\frac{1}{3}}, Flow(T_2) = \mathbf{\frac{1}{3}}$$

Example: Round-Robin

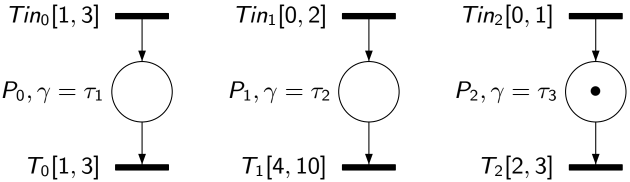


$Proc(\tau_1) = 1$ $Proc(\tau_2) = 1$ $Proc(\tau_3) = 1$

$Prio(\tau_1) = 1$ $Prio(\tau_2) = 1$ $Prio(\tau_3) = 1$

$$Flow(T_0) =?, Flow(T_1) = \mathbf{\frac{1}{2}}, Flow(T_2) = \mathbf{\frac{1}{2}}$$

Example: Round-Robin



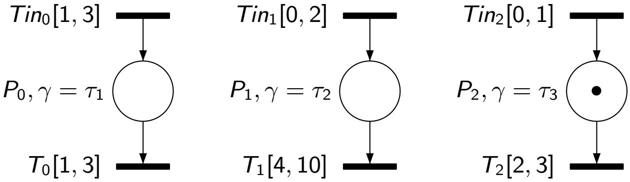
$Proc(\tau_1) = 1$ $Prio(\tau_1) = 1$

$Proc(\tau_2) = 1$ $Prio(\tau_2) = 1$

$Proc(\tau_3) = 1$ $Prio(\tau_3) = 1$

$$Flow(T_0) = ?, Flow(T_1) = ?, Flow(T_2) = \mathbf{1}$$

Example: Round-Robin

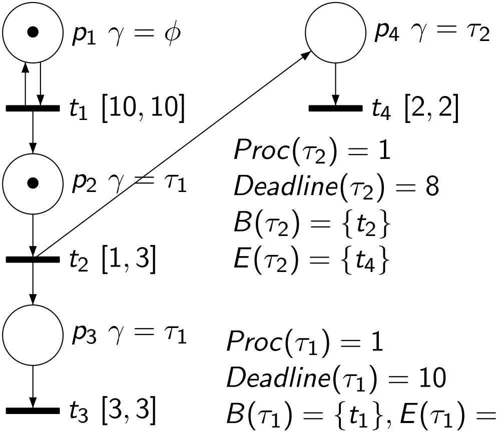


$Proc(\tau_1) = 1$ $Proc(\tau_2) = 1$ $Proc(\tau_3) = 1$

$Prio(\tau_1) = 1$ $Prio(\tau_2) = 1$ $Prio(\tau_3) = 1$

A **fluid** approach: minimize the number of discrete changes.

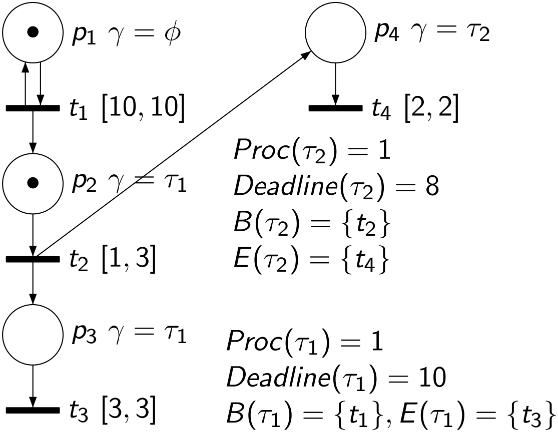
Example: Earliest Deadline First



$Proc(\tau_2) = 1$
$Deadline(\tau_2) = 8$
$B(\tau_2) = \{t_2\}$
$E(\tau_2) = \{t_4\}$

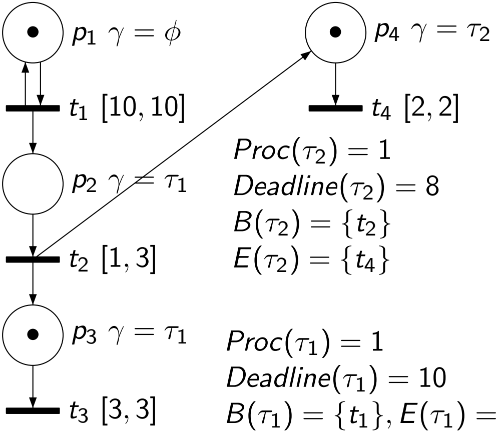
$Proc(\tau_1) = 1$
$Deadline(\tau_1) = 10$
$B(\tau_1) = \{t_1\}, E(\tau_1) = \{t_3\}$

Example: Earliest Deadline First



$Flow(t_1) = 1 \quad Flow(t_2) = 1 \quad Flow(t_3) =? \quad Flow(t_4) =?$

Example: Earliest Deadline First

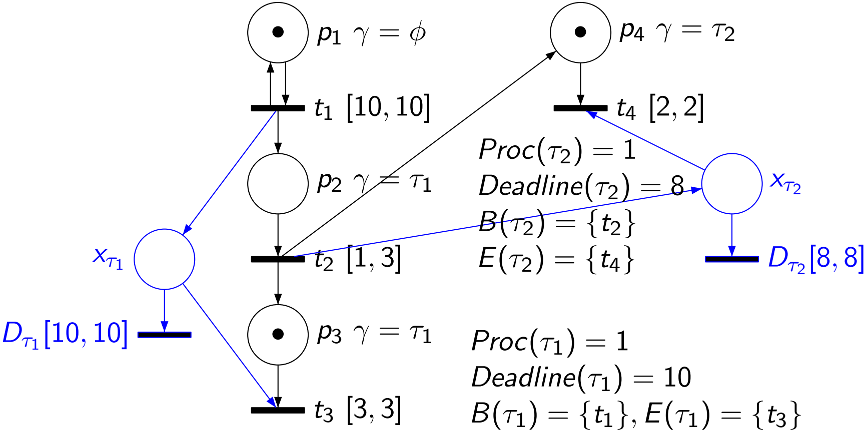


$Proc(\tau_2) = 1$
$Deadline(\tau_2) = 8$
$B(\tau_2) = \{t_2\}$
$E(\tau_2) = \{t_4\}$

$Proc(\tau_1) = 1$
$Deadline(\tau_1) = 10$
$B(\tau_1) = \{t_1\}, E(\tau_1) = \{t_3\}$

if t_2 was fired **before** 2
$Flow(t_1) = 1$ $Flow(t_2) = ?$ $Flow(t_3) = 0$ $Flow(t_4) = \mathbf{1}$
if t_2 was fired **after** 2
$Flow(t_1) = 1$ $Flow(t_2) = ?$ $Flow(t_3) = \mathbf{1}$ $Flow(t_4) = 0$

Example: Earliest Deadline First



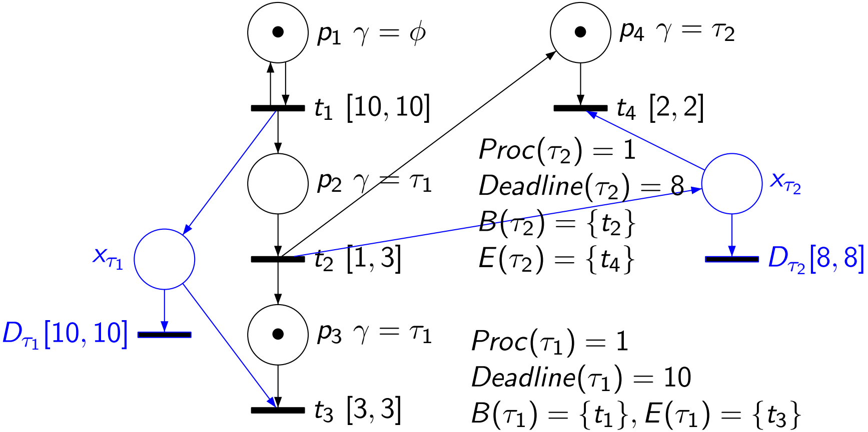
if t_2 was fired **before** 2

$Flow(t_1) = 1$ $\quad Flow(t_2) = ?$ $\quad Flow(t_3) = 0$ $\quad Flow(t_4) = \mathbf{1}$

if t_2 was fired **after** 2

$Flow(t_1) = 1$ $\quad Flow(t_2) = ?$ $\quad Flow(t_3) = \mathbf{1}$ $\quad Flow(t_4) = 0$

Example: Earliest Deadline First



if $\mathbf{D_{\tau_2}} \leq \mathbf{D_{\tau_1}}$

$Flow(t_1) = 1 \quad Flow(t_2) = ? \quad Flow(t_3) = 0 \quad Flow(t_4) = \mathbf{1}$

if $\mathbf{D_{\tau_1}} < \mathbf{D_{\tau_2}}$

$Flow(t_1) = 1 \quad Flow(t_2) = ? \quad Flow(t_3) = \mathbf{1} \quad Flow(t_4) = 0$

Plan I

Abstractions

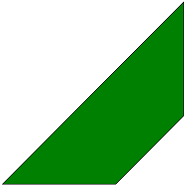
- **Infinite** state-space ⇒ Abstractions
- TPNs: Zone-based simulation graph [5]
- TPNs: **State class graph** [1]
- TPNs w/ stopwatches (IHTPNs,...): **State class graph** [10, 11, 3]

Basic Algorithm

begin
  $Passed = \emptyset$
  $Waiting = \{C_0\}$
  while $Waiting \neq \emptyset$
    $C = \text{pop}(Waiting)$
    $Passed = Passed \cup C$
    for t firable from C
      $C' = \text{AbstractSuccessor}(\mathbf{C}, \mathbf{t})$
      if $C' \notin Passed$
        $Waiting = Waiting \cup C'$
      end if
    end for
  end while
end

State Class

$$C = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, \right.$$



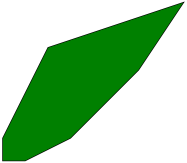
$$\left. \right\}$$

TPNs: Zone (encoded by a Difference Bound Matrix (DBM) $[d_{ij}]_{i,j \in [0..n]}$):

$$\begin{cases} -d_{0i} \leq \theta_i - \mathbf{0} \leq d_{i0}, \\ \theta_i - \theta_j \leq d_{ij} \end{cases}$$

State Class

$$C = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, \right.$$

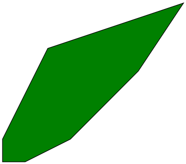


$$\left. \right\}$$

SwTPNs: General polyhedron: $A\bar{\Theta} \leq B$

Over-approximation

$$C = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, \right.$$

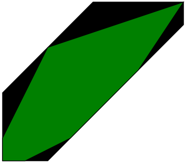


$$\left. \right\}$$

Over-approximation using the **smallest englobing** zone

Over-approximation

$$C = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, \right.$$



$$\left. \right\}$$

Over-approximation using the **smallest englobing** zone

Computing the state class graph (normal)

Let $C = (M, D)$ and $D = (A.\Theta \leq B)$. We fire t_f.

- $M' = M - {}^\bullet t_f + t_f{}^\bullet$
- D' is computed by:
  - for all enabled transitions t_i s.t. $Flow(t_i) \neq 0$, constrain by $\theta_f \leq \theta_i$
  - for all enabled transitions t_i s.t. $Flow(t_i) \neq 0$, $\theta_i' = \theta_i - \theta_f$
  - eliminate variables for **disabled** transitions (e.g. using Fourier-Motzkin method [4])
  - add new variables for **newly** enabled transitions t_i:

$$\alpha(t_i) \leq \theta_i \leq \beta(t_i)$$

Computing the state class graph (round-robin)

Let $C = (M, D)$ and $D = (A.\Theta \leq B)$. We fire t_f.

- $M' = M - {}^\bullet t_f + t_f{}^\bullet$
- D' is computed by:
  - for all enabled transitions t_i s.t. $Flow(t_i) \neq 0$, constrain by $\theta_f \leq \theta_i$
  - for all enabled transitions t_i s.t. $Flow(t_i) \neq 0$, $\theta'_i = \theta_i - \frac{\mathbf{Flow(t_j)}}{\mathbf{Flow(t_f)}}\theta_f$
  - eliminate variables for **disabled** transitions (e.g. using Fourier-Motzkin method [4])
  - add new variables for **newly** enabled transitions t_i:

$$\alpha(t_i) \leq \theta_i \leq \beta(t_i)$$

Computing the state class graph (earliest deadline first)

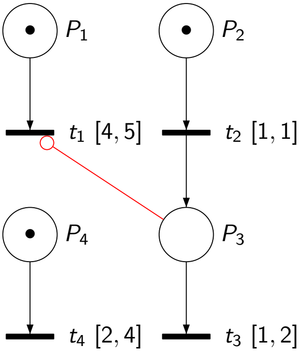
Let $C = (M, D)$ and $D = (A.\Theta \leq B)$. We fire t_f.

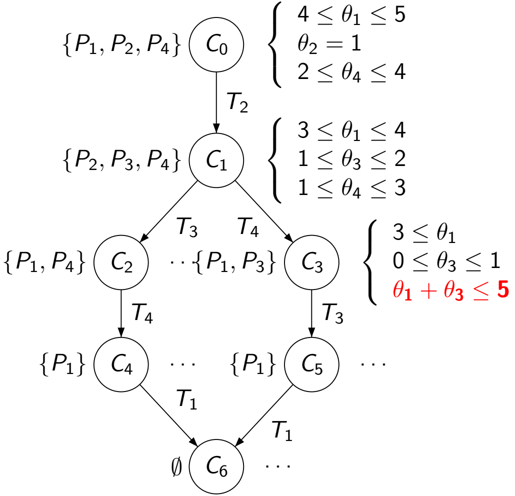
- $M' = M - {}^\bullet t_f + t_f{}^\bullet$
- D' is computed by:
  - for all enabled transitions t_i s.t. $Flow(t_i) \neq 0$, constrain by $\theta_f \leq \theta_i$
  - for all enabled transitions t_i s.t. $Flow(t_i) \neq 0$, $\theta'_i = \theta_i - \theta_f$
  - eliminate variables for **disabled** transitions (e.g. using Fourier-Motzkin method [4])
  - add new variables for **newly** enabled transitions t_i:

$$\alpha(t_i) \leq \theta_i \leq \beta(t_i)$$

  - **partition** D with $D_\tau \leq D_{\tau'}$ and $D_\tau > D_{\tau'}$

Example


P_1
P_2
t_1 $[4, 5]$
t_2 $[1, 1]$
P_4
P_3
t_4 $[2, 4]$
t_3 $[1, 2]$


$\{P_1, P_2, P_4\}$ C_0
$\begin{cases} 4 \leq \theta_1 \leq 5 \\ \theta_2 = 1 \\ 2 \leq \theta_4 \leq 4 \end{cases}$
T_2
$\{P_2, P_3, P_4\}$ C_1
$\begin{cases} 3 \leq \theta_1 \leq 4 \\ 1 \leq \theta_3 \leq 2 \\ 1 \leq \theta_4 \leq 3 \end{cases}$
T_3
T_4
$\{P_1, P_4\}$ C_2
$\cdots$ $\{P_1, P_3\}$ C_3
$\begin{cases} 3 \leq \theta_1 \\ 0 \leq \theta_3 \leq 1 \\ \boldsymbol{\theta_1 + \theta_3 \leq 5} \end{cases}$
T_4
T_3
$\{P_1\}$ C_4 $\cdots$
$\{P_1\}$ C_5 $\cdots$
T_1
T_1
$\emptyset$ C_6 $\cdots$

The Way of the Middle (1/2)

For IHTPNs:

- The polyhedron in the **initial** state class is **always a zone**.
- The successor of a non-zone polyhedron might be a zone
- The successor of a zone might not be a zone

The Way of the Middle (1/2)

For IHTPNs:

- The polyhedron in the **initial** state class is **always a zone**.
- The successor of a non-zone polyhedron might be a zone
- The successor of a zone might not be a zone

We want to start to compute with zones and fall back to general polyhedra when needed (and return to zones asap).

The Way of the Middle (1/2)

For IHTPNs:

- The polyhedron in the **initial** state class is **always a zone**.
- The successor of a non-zone polyhedron might be a zone (**easy to check**: $O(n^2)$)
- The successor of a zone might not be a zone (**not so easy to check**)

We want to start to compute with zones and fall back to general polyhedra when needed (and return to zones asap).

Abstractions: The Way of the Middle (2/2)

$D = [d_{ij}]_{i,j \in [0..n]}$ and $(M', D') = AbstractSuccessor((M, D), t_f)$.

D' is **not a zone** iff there are at least three enabled transitions t_i, t_j, t_k in D such that:

1. t_i, t_j, t_k are not disabled when firing t_f;
2. $Flow(t_i) \neq 0$ and $Flow(t_k) \neq 0$;
3. $Flow(t_j) = 0$;
4. $d_{j0} + d_{ki} > d_{k0} + d_{ji}$ or $d_{0j} - d_{ik} < d_{0k} - d_{ij}$

Abstractions: The Way of the Middle (2/2)

$D = [d_{ij}]_{i,j \in [0..n]}$ and $(M', D') = AbstractSuccessor((M, D), t_f)$.

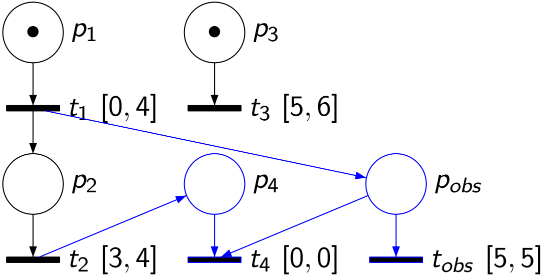
D' is **not a zone** iff there are at least three enabled transitions t_i, t_j, t_k in D such that:

1. t_i, t_j, t_k are not disabled when firing t_f;
2. $Flow(t_i) \neq 0$ and $Flow(t_k) \neq 0$;
3. $Flow(t_j) = 0$;
4. $d_{j0} + d_{ki} > d_{k0} + d_{ji}$ or $d_{0j} - d_{ik} < d_{0k} - d_{ij}$

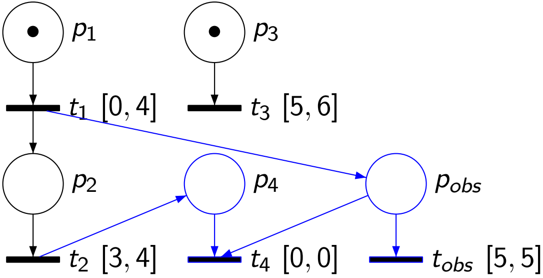
Complexity: $O(n^3)$

Plan I

Observing durations

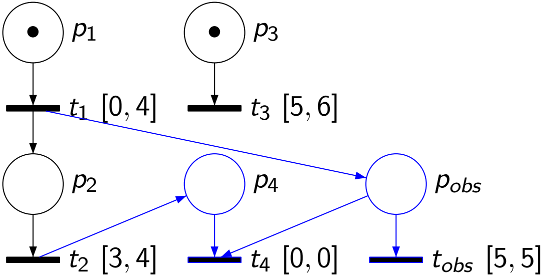


Observing durations



There can be ≥ 5 t.u. between the firings of t_1 and t_2 iff t_{obs} is fired.

Observing durations



$\boxed{\text{the max sojourn time of the token is } \max_{(M,D)\in \text{Classes}}\{5 - \min(D_{|t_{obs}})\}.}$

TCTL Model-checking

- Techniques for the verification of **Timed Computation Tree Logic** (TCTL) exist for TPNs ([6, 7]).

TCTL Model-checking

- Techniques for the verification of **Timed Computation Tree Logic** (TCTL) exist for TPNs ([6, 7]).
- They can be (easily) extended to work with SwTPNs.

TCTL Model-checking

- Techniques for the verification of **Timed Computation Tree Logic** (TCTL) exist for TPNs ([6, 7]).
- They can be (easily) extended to work with SwTPNs.
- One can check for instance $AG(M(p_2) \geq 1) \Rightarrow EF[0,5](M(p_2) = 0))$ (bounded response).

TCTL Model-checking

- Techniques for the verification of **Timed Computation Tree Logic** (TCTL) exist for TPNs ([6, 7]).
- They can be (easily) extended to work with SwTPNs.
- One can check for instance $AG(M(p_2) \geq 1) \Rightarrow EF[0,5](M(p_2) = 0))$ (bounded response).
- This is implemented ina Romeo! (`http://romeo.rts-software.org`)

TCTL Model-checking

- Techniques for the verification of **Timed Computation Tree Logic** (TCTL) exist for TPNs ([6, 7]).
- They can be (easily) extended to work with SwTPNs.
- One can check for instance $AG(M(p_2) \geq 1) \Rightarrow EF[0,5](M(p_2) = 0))$ (bounded response).
- This is implemented ina ROMEO! (`http://romeo.rts-software.org`)
- **Warning**: It cannot express properties on stopwatches (response times but not cumulated execution durations).

Plan I

Conclusion

- Time Petri Nets with Stopwatches (and more) are a very nice formalism for **real-time systems** modelling, including (preemptive) scheduling;

Conclusion

- Time Petri Nets with Stopwatches (and more) are a very nice formalism for **real-time systems** modelling, including (preemptive) scheduling;
- Theoretical properties are not good but they are **usable in practice**!

Conclusion

- Time Petri Nets with Stopwatches (and more) are a very nice formalism for **real-time systems** modelling, including (preemptive) scheduling;
- Theoretical properties are not good but they are **usable in practice**!
- Some **tools** exist including ROMEO (IRCCyN, Nantes) and TINA (LAAS, Toulouse).

Future Work

Future work includes:

- Interaction with **higher-level** models and specifications languages;

Future Work

Future work includes:

- Interaction with **higher-level** models and specifications languages;
- **Discrete time** semantics;

Future Work

Future work includes:

- Interaction with **higher-level** models and specifications languages;
- **Discrete time** semantics;
- **Parametric** extensions;

Future Work

Future work includes:

- Interaction with **higher-level** models and specifications languages;
- **Discrete time** semantics;
- **Parametric** extensions;
- **Control** problems.

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.

B. Berthomieu, D. Lime, O.H. Roux, and F. Vernadat.
Reachability problems and abstract state spaces for time petri nets with stopwatches.
Journal of Discrete Event Dynamic Systems (jDEDS), 17(2):133–158, 2007.

G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario.
Time state space analysis of real-time preemptive systems.
IEEE transactions on software engineering, 30(2):97–111, February 2004.

G.B. Dantzig.
Linear programming and extensions.
IEICE Transactions on Information and Systems, 1963.

G. Gardey, O.H. Roux, and O.F. Roux.
State space computation and analysis of time Petri nets.
Theory and Practice of Logic Programming (TPLP). Special Issue on Specification Analysis and Verification of Reactive Systems, 2005.
to appear.

Guillaume Gardey.
Rseaux de Petri temporels.
PhD thesis, Université de Nantes, Nantes, France, 2005.

Rachid Hadjidj and Hanifa Boucheneb.
On-the-fly tctl model checking for time petri nets using state class graphs.
In *Sixth International Conference on Application of Concurrency to System Design (ACSD'06)*, pages 111–122, 2006.

R. Janicki and M. Koutny.
Semantics of inhibitor nets.
Information and Computation, 123(1):1–15, 1995.

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.

D. Lime and O.H. Roux.
Expressiveness and analysis of scheduling extended time Petri nets.
In *5th IFAC International Conference on Fieldbus Systems and their Applications, (FET 2003)*, Aveiro, Portugal, July 2003. Elsevier Science.

O.H. Roux and D. Lime.
Time Petri nets with inhibitor hyperarcs. Formal semantics and state space computation.
In Jordi Cortadella and Wolfgang Reisig, editors, *The 25th International Conference on Application and Theory of Petri Nets, (ICATPN 2004)*, volume 3099 of *Lecture Notes in Computer Science*, pages 371–390, Bologna, Italy, June 2004. Springer-Verlag.