On Time Petri Nets and Scheduling

Didier Lime

IRCCyN / École Centrale de Nantes

5 septembre 2007
Plan I

Introduction

Time Petri Nets with Stopwatches (and more)
  Petri Nets and Extensions
  Time Petri Nets
  Scheduling Time Petri Nets

Abstractions for Scheduling-TPNs
  The State Class Graph
  Polyhedra On Demand!

Verifying properties
  Observers
  Model-checking

Conclusion and Future Work
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;
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

Introduction

Time Petri Nets with Stopwatches (and more)

Petri Nets and Extensions
Time Petri Nets
Scheduling Time Petri Nets

Abstractions for Scheduling-TPNs

Verifying properties

Conclusion and Future Work
Petri Nets

\[
\begin{align*}
\text{if } M \geq \cdot t & \text{ then } M' = M - \cdot t + t^* \\
\end{align*}
\]
Petri Net Example: Basic Semaphore Synchronization

\[ T_{semV} \quad P_{sem} \quad T_{semP} \]

\[ P_1 \xrightarrow{\tau_1} \quad T_1 \quad P_2 \]

\[ P_3 \xrightarrow{\tau_2} \quad T_2 \quad P_4 \]
Petri Net Example: Mutual Exclusion

\[
\begin{align*}
P_1 &\quad \text{sem}P \\
&\quad \text{sc} \\
&\quad \text{sem}V \\
P_{sem} &\quad \text{sc} \\
&\quad \text{sem}P \\
&\quad \text{sem}V \\
P_3 &
\end{align*}
\]
Petri Net Example: Peterson’s Algorithm

\[ P: \]
\[ d \leftarrow \text{false} \]
\[ \text{loop} \]
\[ \quad \begin{align*}
& \quad \text{<non critical section>} \\
& \quad d \leftarrow \text{true} \\
& \quad \text{turn} \leftarrow 1 \\
& \quad \text{wait}(\neg d' \lor \text{turn} = 0) \\
& \quad \text{<critical section>} \\
& \quad d \leftarrow \text{false} \\
\end{align*} \]
\[ \text{end\_loop} \]

\[ P': \]
\[ d' \leftarrow \text{false} \]
\[ \text{loop} \]
\[ \quad \begin{align*}
& \quad \text{<non critical section>} \\
& \quad d' \leftarrow \text{true} \\
& \quad \text{turn} \leftarrow 0 \\
& \quad \text{wait}(\neg d \lor \text{turn} = 1) \\
& \quad \text{<critical section>} \\
& \quad d' \leftarrow \text{false} \\
\end{align*} \]
\[ \text{end\_loop} \]
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

\[ P_0 \xrightarrow{\bullet t} T_0 \]

\[ P_0 \xrightarrow{\Diamond t} T_0 \]

\[
\text{if } M \geq \bullet t \text{ and } M \geq \Diamond t \text{ then } M' = M - \bullet t + t^* \]

Didier Lime (IRCCyN / ECN)
Inhibitor Arcs

If $M \geq \cdot t$ and $M < \circ t$ then $M' = M - \cdot t + t^\bullet$
Petri Net Example: Fixed Priority Scheduling (non-preemptive)
Time Petri Nets

A Petri Net
**Time Petri Nets**

![Time Petri Net Diagram]

A *Time* Petri Net
Time Petri Nets

A Time Petri Net

\[
\left( M_0, \begin{array}{c} t_1 = 0, \\ t_3 = 0 \end{array} \right)
\]
Time Petri Nets

A **Time** Petri Net

\[
(M_0, \ t_1 = 0, \ t_3 = 0) \xrightarrow{0.62} (M_0, \ t_1 = 0.62, \ t_3 = 0.62)
\]
A **Time** Petri Net

\[
\begin{pmatrix}
M_0, & t_1 = 0, \\
t_3 = 0
\end{pmatrix}
\xrightarrow{0.62}
\begin{pmatrix}
M_0, & t_1 = 0.62, \\
t_3 = 0.62
\end{pmatrix}
\xrightarrow{t_1}
\begin{pmatrix}
M_1, & t_1 = 0.62, \\
t_2 = 0
\end{pmatrix}
\]
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

(a) (b) (c)
Read Arcs and Time

\[ T_1[1, 2] \]

\[ P_0 \]

\[ T_0[3, 4] \]
Read Arcs and Time

\[ T_1[1, 2] \]

\[ P_0 \]

\[ T_0[3, 4] \]

Read should not be destructive:

\[ (M_0, \begin{align*} T_0 &= 0, \\ T_1 &= 0 \end{align*}) \]
Read Arcs and Time

Read should not be destructive:

\[
\begin{align*}
(M_0, \quad T_0 = 0, \quad T_1 = 0) & \xrightarrow{1.62} (M_0, \quad T_0 = 1.62, \quad T_1 = 1.62)
\end{align*}
\]
Read Arcs and Time

Read should not be destructive:

\[
\left( M_0, \frac{T_0}{T_1} = 0, \frac{T_0}{T_1} = 0 \right) \xrightarrow{1.62} \left( M_0, \frac{T_0}{T_1} = 1.62, \frac{T_0}{T_1} = 1.62 \right) \xrightarrow{T_0} \left( M_0, \frac{T_0}{T_1} = 0, \frac{T_0}{T_1} = 0 \right)
\]
Read Arcs and Time

Read should not be destructive:

\[
\begin{align*}
(M_0, & \quad T_0 = 0, \quad T_1 = 0) \quad \overset{1.62}{\rightarrow} \quad (M_0, \quad T_0 = 1.62, \quad T_1 = 1.62) \quad \overset{T_0}{\rightarrow} \quad (M_0, \quad T_0 = 0, \quad T_1 = 0)
\end{align*}
\]
From Read Arcs To Activator Arcs
From Read Arcs To Activator Arcs

\[
\begin{align*}
& P_0 \quad P_1 \\
& T_0[3, 4] \quad T_1[1, 2] \quad T_2[0, 1] \\
& P_2
\end{align*}
\]

Memory / No Memory:

\[
\left( M_0, \quad T_0 = 0, \quad T_1 = 0 \right)
\]
From Read Arcs To Activator Arcs

Memory / No Memory:

\[ \left( M_0, \quad T_0 = 0, \quad T_1 = 0 \right) \xrightarrow{1.62} \left( M_0, \quad T_0 = 1.62, \quad T_1 = 1.62 \right) \]
From Read Arcs To Activator Arcs

Memory / No Memory:

\[
\left( M_0, \ T_0 = 0, \ T_1 = 0 \right) \xrightarrow{1.62} \left( M_0, \ T_0 = 1.62, \ T_1 = 0 \right) \xrightarrow{T_1} \left( M_1, \ T_2 = 0 \right)
\]
From Read Arcs To Activator Arcs

Memory / No Memory:

\[
\begin{align*}
(M_0, T_0 = 0, T_1 = 0) & \xrightarrow{1.62} (M_0, T_0 = 1.62, T_1 = 1.62) & \xrightarrow{T_1} (M_1, T_2 = 0) \\
& \xrightarrow{0.2} (M_1, T_2 = 0.2)
\end{align*}
\]
From Read Arcs To Activator Arcs

Memory / No Memory:

\[
\begin{align*}
(M_0, T_0 = 0, T_1 = 0) & \xrightarrow{1.62} (M_0, T_0 = 1.62, T_1 = 1.62) \xrightarrow{T_1} (M_1, T_2 = 0) \\
\xrightarrow{0.2} (M_1, T_2 = 0.2) & \xrightarrow{T_2} (M_0, T_0 = 0, T_1 = 0)
\end{align*}
\]
From Read Arcs To Activator Arcs

\[ P_0 \xrightarrow{T_0[3, 4]} P_1 \]

\[ P_2 \xrightarrow{T_2[0, 1]} T_1[1, 2] \]

Memory / No Memory:

\[
\begin{align*}
(M_0, T_0 = 0, T_1 = 0) & \xrightarrow{1.62} (M_0, T_0 = 1.62, T_1 = 1.62) \\
& \xrightarrow{T_1} (M_1, T_0 = 1.62, T_2 = 0)
\end{align*}
\]

\[
\begin{align*}
(M_1, T_2 = 0.2) & \xrightarrow{T_2} (M_0, T_0 = 0, T_1 = 0) \\
& \xrightarrow{0.31} (M_0, T_0 = 0.31, T_1 = 0.31)
\end{align*}
\]
From Read Arcs To Activator Arcs

Memory / No Memory:

\[
(M_0, T_0 = 0, T_1 = 0) \xrightarrow{1.62} (M_0, T_0 = 1.62, T_1 = 1.62) \xrightarrow{T_1} (M_1, T_0 = 1.62, T_2 = 0)
\]

\[
0.2 \cdot (M_1, T_0 = 1.62, T_2 = 0.2) \xrightarrow{T_2} (M_0, T_0 = 0, T_1 = 0) \xrightarrow{0.31} (M_0, T_0 = 0.31, T_1 = 0.31)
\]
From Read Arcs To Activator Arcs

Memory / No Memory:

\[
\begin{align*}
(M_0, \frac{T_0}{T_1} = 0, 0) &\xrightarrow{1.62} (M_0, \frac{T_0}{T_1} = 1.62, 0) &\xrightarrow{T_1} (M_1, \frac{T_0}{T_2} = 1.62, 0) \\
0.2 \times (M_1, \frac{T_0}{T_2} = 1.62, 0.2) &\xrightarrow{T_2} (M_0, \frac{T_0}{T_1} = 1.62, 0) &\xrightarrow{0.31} (M_0, \frac{T_0}{T_1} = 0.31, 0.31)
\end{align*}
\]
From Read Arcs To Activator Arcs

Memory / No Memory:

\[
(M_0, \frac{T_0}{T_1} = 0, 0) \xrightarrow{1.62} (M_0, \frac{T_0}{T_1} = 1.62, 0) \xrightarrow{T_1} (M_1, \frac{T_0}{T_2} = 1.62, 0)
\]

\[
(M_1, \frac{T_0}{T_2} = 1.62, 0.2) \xrightarrow{T_2} (M_0, \frac{T_0}{T_1} = 1.62, 0) \xrightarrow{0.31} (M_0, \frac{T_0}{T_1} = 1.93, 0.31)
\]
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)

\[
\begin{align*}
T_{in0}[1, 3] & \quad P_0 \\
T_0[1, 3] & \quad T_{in1}[0, 2] \\
& \quad P_1 \\
& \quad T_{in2}[0, 1] \\
& \quad P_2 \\
T_1[4, 10] & \quad T_2[2, 3]
\end{align*}
\]

\[T_0 = ? \quad T_1 = 0 \quad T_2 = ?\]
Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)

\[ Tin_0[1, 3] \quad Tin_1[0, 2] \quad Tin_2[0, 1] \]

\[ T_0[1, 3] \quad T_1[4, 10] \quad T_2[2, 3] \]

\[ T_0 =? \quad T_1 = 0.22 \quad T_2 =? \]
Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)

\[ T_{in_0}[1, 3] \]
\[ T_{in_1}[0, 2] \]
\[ T_{in_2}[0, 1] \]

\[ P_0 \]
\[ P_1 \]
\[ P_2 \]

\[ T_0[1, 3] \]
\[ T_1[4, 10] \]
\[ T_2[2, 3] \]

\[ T_0 = ? \]
\[ T_1 = 0.22 \]
\[ T_2 = 0 \]
Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)

\[T_{in_0}[1,3] \quad T_{in_1}[0,2] \quad T_{in_2}[0,1]\]

\[P_0 \quad P_1 \quad P_2\]

\[T_0[1,3] \quad T_1[4,10] \quad T_2[2,3]\]

\[T_0 = ? \quad T_1 = 1.6 \quad T_2 = 0\]
Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)

\[
\begin{align*}
T_{in_0}[1, 3] & \quad T_{in_1}[0, 2] & \quad T_{in_2}[0, 1] \\
\rightarrow & \quad \rightarrow & \quad \rightarrow \\
\text{\(P_0\)} & \quad \text{\(P_1\)} & \quad \text{\(P_2\)} \\
T_0[1, 3] & \quad T_1[4, 10] & \quad T_2[2, 3] \\
\rightarrow & \quad \rightarrow & \quad \rightarrow
\end{align*}
\]

\[T_0 = 0 \quad T_1 = 1.6 \quad T_2 = 0\]
Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)

\[ T_{in_0}[1, 3] \quad T_{in_1}[0, 2] \quad T_{in_2}[0, 1] \]

\[ P_0 \quad P_1 \quad P_2 \]

\[ T_0[1, 3] \quad T_1[4, 10] \quad T_2[2, 3] \]

\[ T_0 = 1.5 \quad T_1 = 1.6 \quad T_2 = 0 \]
Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)

\[ T_{in0}[1, 3] \] \( P_0 \) \( T_0[1, 3] \) 
\[ T_{in1}[0, 2] \] \( P_1 \) \( T_1[4, 10] \) 
\[ T_{in2}[0, 1] \] \( P_2 \) \( T_2[2, 3] \)

\[ T_0 = ? \quad T_1 = 1.6 \quad T_2 = 0 \]
Time Inhibitor Arcs: Fixed Priority Scheduling (Preemptive)

\[
\begin{align*}
T_{in_0}[1, 3] & \quad P_0 \\
T_0[1, 3] & \quad T_{in_1}[0, 2] \\
T_1[4, 10] & \quad T_{in_2}[0, 1] \\
T_2[2, 3] & \quad P_2
\end{align*}
\]

\[T_0 = ? \quad T_1 = 1.8 \quad T_2 = 0\]
The price of expressiveness

The reachability problem:

<table>
<thead>
<tr>
<th></th>
<th>General case</th>
<th>Bounded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Petri Nets</td>
<td>decidable</td>
<td>decidable</td>
</tr>
<tr>
<td>Petri Nets w/ Inhibitor</td>
<td>undecidable [8]</td>
<td>decidable</td>
</tr>
</tbody>
</table>
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 Scheduling (Preemptive)

\[ T_{in_0}[1,3] \rightarrow P_0, \gamma = \tau_1 \rightarrow T_0[1,3] \]
\[ T_{in_1}[0,2] \rightarrow P_1, \gamma = \tau_2 \rightarrow T_1[4,10] \]
\[ T_{in_2}[0,1] \rightarrow P_2, \gamma = \tau_3 \rightarrow T_2[2,3] \]

\[ Proc(\tau_1) = 1 \]
\[ Prio(\tau_1) = 3 \]
\[ Proc(\tau_2) = 1 \]
\[ Prio(\tau_2) = 2 \]
\[ Proc(\tau_3) = 1 \]
\[ Prio(\tau_3) = 1 \]
Example: Fixed Priority Scheduling (Preemptive)

\[ Tin_0[1, 3] \]
\[ P_0, \gamma = \tau_1 \]
\[ T_0[1, 3] \]

\[ Tin_1[0, 2] \]
\[ P_1, \gamma = \tau_2 \]
\[ T_1[4, 10] \]

\[ Tin_2[0, 1] \]
\[ P_2, \gamma = \tau_3 \]
\[ T_2[2, 3] \]

\[ Proc(\tau_1) = 1 \]
\[ Prio(\tau_1) = 3 \]

\[ Proc(\tau_2) = 1 \]
\[ Prio(\tau_2) = 2 \]

\[ Proc(\tau_3) = 1 \]
\[ Prio(\tau_3) = 1 \]

\[ Flow(T_0) = 1, Flow(T_1) = 0, Flow(T_2) = 0 \]
Example: Fixed Priority Scheduling (Preemptive)

\[ T_{in0}[1, 3] \]
\[ P_0, \gamma = \tau_1 \]
\[ T_0[1, 3] \]

\[ T_{in1}[0, 2] \]
\[ P_1, \gamma = \tau_2 \]
\[ T_1[4, 10] \]

\[ T_{in2}[0, 1] \]
\[ P_2, \gamma = \tau_3 \]
\[ T_2[2, 3] \]

\[ Proc(\tau_1) = 1 \]
\[ Prio(\tau_1) = 3 \]

\[ Proc(\tau_2) = 1 \]
\[ Prio(\tau_2) = 2 \]

\[ Proc(\tau_3) = 1 \]
\[ Prio(\tau_3) = 1 \]

\[ Flow(T_0) = ?, Flow(T_1) = 1, Flow(T_2) = 0 \]
Example: Fixed Priority Scheduling (Preemptive)

\[ \text{Proc}(\tau_1) = 1 \]
\[ \text{Proc}(\tau_2) = 1 \]
\[ \text{Proc}(\tau_3) = 1 \]

\[ \text{Prio}(\tau_1) = 3 \]
\[ \text{Prio}(\tau_2) = 2 \]
\[ \text{Prio}(\tau_3) = 1 \]

\[ \text{Flow}(T_0) = ?, \text{Flow}(T_1) = ?, \text{Flow}(T_2) = 1 \]
Example: Round-Robin

\begin{align*}
T_{in_0}[1, 3] & \quad T_{in_1}[0, 2] & \quad T_{in_2}[0, 1] \\
\text{Proc}(\tau_1) &= 1 & \text{Proc}(\tau_2) &= 1 & \text{Proc}(\tau_3) &= 1 \\
Prio(\tau_1) &= 1 & Prio(\tau_2) &= 1 & Prio(\tau_3) &= 1
\end{align*}
Example: Round-Robin

\[
\begin{align*}
T_{in_0}[1, 3] & \quad T_{in_1}[0, 2] & \quad T_{in_2}[0, 1] \\
\bullet & \quad \bullet & \quad \bullet \\
T_0[1, 3] & \quad T_1[4, 10] & \quad T_2[2, 3] \\
\end{align*}
\]

\[
\begin{align*}
Proc(\tau_1) = 1 & \quad Proc(\tau_2) = 1 & \quad Proc(\tau_3) = 1 \\
Prio(\tau_1) = 1 & \quad Prio(\tau_2) = 1 & \quad Prio(\tau_3) = 1 \\
Flow(T_0) = \frac{1}{3} & \quad Flow(T_1) = \frac{1}{3} & \quad Flow(T_2) = \frac{1}{3}
\end{align*}
\]
Example: Round-Robin

\begin{align*}
T_{in_0}[1, 3] & \quad T_{in_1}[0, 2] & \quad T_{in_2}[0, 1] \\
P_0, \gamma = \tau_1 & \quad P_1, \gamma = \tau_2 & \quad P_2, \gamma = \tau_3 \\
T_0[1, 3] & \quad T_1[4, 10] & \quad T_2[2, 3] \\
Proc(\tau_1) = 1 & \quad Proc(\tau_2) = 1 & \quad Proc(\tau_3) = 1 \\
Prio(\tau_1) = 1 & \quad Prio(\tau_2) = 1 & \quad Prio(\tau_3) = 1 \\
Flow(T_0) = ? & \quad Flow(T_1) = \frac{1}{2} & \quad Flow(T_2) = \frac{1}{2}
\end{align*}
Example: Round-Robin

\[ Tin_0[1, 3] \]
\[ P_0, \gamma = \tau_1 \]
\[ T_0[1, 3] \]

\[ Tin_1[0, 2] \]
\[ P_1, \gamma = \tau_2 \]
\[ T_1[4, 10] \]

\[ Tin_2[0, 1] \]
\[ P_2, \gamma = \tau_3 \]
\[ T_2[2, 3] \]

\[ Proc(\tau_1) = 1 \]
\[ Prio(\tau_1) = 1 \]

\[ Proc(\tau_2) = 1 \]
\[ Prio(\tau_2) = 1 \]

\[ Proc(\tau_3) = 1 \]
\[ Prio(\tau_3) = 1 \]

\[ Flow(T_0) = ?, Flow(T_1) = ?, Flow(T_2) = 1 \]
Example: Round-Robin

A **fluid** approach: minimize the number of discrete changes.
Example: Earliest Deadline First

- $p_1 \gamma = \phi$
- $t_1 [10, 10]$

- $p_2 \gamma = \tau_1$
- $t_2 [1, 3]$
- $B(\tau_2) = \{t_2\}$
- $E(\tau_2) = \{t_4\}$
- $Proc(\tau_2) = 1$
- $Deadline(\tau_2) = 8$

- $p_3 \gamma = \tau_1$
- $t_3 [3, 3]$
- $B(\tau_1) = \{t_1\}$, $E(\tau_1) = \{t_3\}$
- $Proc(\tau_1) = 1$
- $Deadline(\tau_1) = 10$

- $p_4 \gamma = \tau_2$
- $t_4 [2, 2]$
- $Proc(\tau_2) = 1$
- $Deadline(\tau_2) = 8$
Example: Earliest Deadline First

\[ p_1 \quad \gamma = \phi \]
\[ t_1 \quad [10, 10] \]
\[ p_2 \quad \gamma = \tau_1 \]
\[ t_2 \quad [1, 3] \]
\[ p_3 \quad \gamma = \tau_1 \]
\[ t_3 \quad [3, 3] \]
\[ p_4 \quad \gamma = \tau_2 \]
\[ t_4 \quad [2, 2] \]

\[ \text{Proc}(\tau_1) = 1 \]
\[ \text{Deadline}(\tau_1) = 10 \]
\[ B(\tau_1) = \{t_1\}, E(\tau_1) = \{t_3\} \]
\[ \text{Proc}(\tau_2) = 1 \]
\[ \text{Deadline}(\tau_2) = 8 \]
\[ B(\tau_2) = \{t_2\}, E(\tau_2) = \{t_4\} \]

\[ \text{Flow}(t_1) = 1 \quad \text{Flow}(t_2) = 1 \quad \text{Flow}(t_3) = \text{?} \quad \text{Flow}(t_4) = \text{?} \]
Example: Earliest Deadline First

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

if $t_2$ was fired after 2
$Flow(t_1) = 1$
$Flow(t_2) = \text{?}$
$Flow(t_3) = 1$
$Flow(t_4) = 0$
Example: Earliest Deadline First

\[
\begin{align*}
\text{Proc}(\tau_1) &= 1 \\
\text{Deadline}(\tau_1) &= 10 \\
B(\tau_1) &= \{ t_1 \} \\
E(\tau_1) &= \{ t_3 \} \\
\text{Proc}(\tau_2) &= 1 \\
\text{Deadline}(\tau_2) &= 8 \\
B(\tau_2) &= \{ t_2 \} \\
E(\tau_2) &= \{ t_4 \} \\
\end{align*}
\]

if \( t_2 \) was fired before 2
\[
\begin{align*}
\text{Flow}(t_1) &= 1 \\
\text{Flow}(t_2) &= ? \\
\text{Flow}(t_3) &= 0 \\
\text{Flow}(t_4) &= 1
\end{align*}
\]

if \( t_2 \) was fired after 2
\[
\begin{align*}
\text{Flow}(t_1) &= 1 \\
\text{Flow}(t_2) &= ? \\
\text{Flow}(t_3) &= 1 \\
\text{Flow}(t_4) &= 0
\end{align*}
\]
Example: Earliest Deadline First

**Petri Net Diagram**

- **Places**:
  - $p_1 \ \gamma = \phi$
  - $p_2 \ \gamma = \tau_1$
  - $p_3 \ \gamma = \tau_1$
  - $p_4 \ \gamma = \tau_2$

- **Transitions**:
  - $t_1 [10, 10]$
  - $t_2 [1, 3]$
  - $t_3 [3, 3]$
  - $t_4 [2, 2]$

- **External Marks**:
  - $D_{\tau_1} [10, 10]$
  - $D_{\tau_2} [8, 8]$

- **Deadlines**:
  - $\text{Deadline}(\tau_2) = 8$
  - $\text{Deadline}(\tau_1) = 10$

- **Processes**:
  - $\text{Proc}(\tau_2) = 1$
  - $\text{Proc}(\tau_1) = 1$

- **Bounds**:
  - $B(\tau_2) = \{ t_2 \}$
  - $B(\tau_1) = \{ t_1 \}$
  - $E(\tau_2) = \{ t_4 \}$
  - $E(\tau_1) = \{ t_3 \}$

- **Flow**
  - If $D_{\tau_2} \leq D_{\tau_1}$
    - $\text{Flow}(t_1) = 1$
    - $\text{Flow}(t_2) = ?$
    - $\text{Flow}(t_3) = 0$
    - $\text{Flow}(t_4) = 1$
  - If $D_{\tau_1} < D_{\tau_2}$
    - $\text{Flow}(t_1) = 1$
    - $\text{Flow}(t_2) = ?$
    - $\text{Flow}(t_3) = 1$
    - $\text{Flow}(t_4) = 0$
Plan 1

Introduction

Time Petri Nets with Stopwatches (and more)

Abstractions for Scheduling-TPNs
  The State Class Graph
  Polyhedra On Demand!

Verifying properties

Conclusion and Future Work
Abstractions

- Infinite state-space $\Rightarrow$ 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 = ∅
  Waiting = \{ C_0 \}
  while Waiting ≠ ∅
    C = pop(Waiting)
    Passed = Passed ∪ C
    for t firable from C
      C' = AbstractSuccessor(C, t)
      if C' ∉ Passed
        Waiting = Waiting ∪ C'
      end if
    end for
  end while
end
State Class

\[ C = \begin{cases} \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix}, \end{cases} \]

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

\[
\begin{cases} 
-d_{0i} \leq \theta_i - 0 \leq d_{i0}, \\
\theta_i - \theta_j \leq d_{ij}
\end{cases}
\]
State Class

\[ C = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} \right\} \]

SwTPNs: General polyhedron: \( A\bar{\Theta} \leq B \)
Over-approximation

\[ C = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, \right\} \]

Over-approximation using the \textit{smallest englobing} zone
Over-approximation

\[ C = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} \right\} \]

Over-approximation using the \textit{smallest englobing} zone
Computing the state class graph (normal)

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

- \( M' = M - t_f + t_f^\bullet \)
- \( D' \) is computed by:
  - for all enabled transitions \( t_i \) s.t. \( \text{Flow}(t_i) \neq 0 \), constrain by \( \theta_f \leq \theta_i \)
  - for all enabled transitions \( t_i \) s.t. \( \text{Flow}(t_i) \neq 0 \), \( \theta'_i = \theta_i - \theta_f \)
  - 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. $\text{Flow}(t_i) \neq 0$, constrain by $\theta_f \leq \theta_i$
  - for all enabled transitions $t_i$ s.t. $\text{Flow}(t_i) \neq 0$, $\theta'_i = \theta_i - \frac{\text{Flow}(t_i)}{\text{Flow}(t_f)} \theta_f$
  - 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. $\text{Flow}(t_i) \neq 0$, constrain by $\theta_f \leq \theta_i$
  - for all enabled transitions $t_i$ s.t. $\text{Flow}(t_i) \neq 0$, $\theta'_i = \theta_i - \theta_f$
  - 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 $D_\tau > D_{\tau'}$
Abstractions for Scheduling-TPNs

The State Class Graph

Example

\[ \{P_1, P_2, P_4\} \rightarrow \{P_1, P_2, P_3, P_4\} \rightarrow \{P_1, P_3, P_4\} \rightarrow \{P_1\} \rightarrow \emptyset \]

\[ 4 \leq \theta_1 \leq 5 \]
\[ \theta_2 = 1 \]
\[ 2 \leq \theta_4 \leq 4 \]

\[ 3 \leq \theta_1 \leq 4 \]
\[ 1 \leq \theta_3 \leq 2 \]
\[ 1 \leq \theta_4 \leq 3 \]

\[ 3 \leq \theta_1 \]
\[ 0 \leq \theta_3 \leq 1 \]
\[ \theta_1 + \theta_3 \leq 5 \]
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)

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

\( D' \) 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. \( \text{Flow}(t_i) \neq 0 \) and \( \text{Flow}(t_k) \neq 0 \);
3. \( \text{Flow}(t_j) = 0 \);
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)

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

\[ D' \text{ is not a zone iff there are at least three enabled transitions } t_i, t_j, t_k \text{ in } D \text{ such that:} \]

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

**Complexity:** \( O(n^3) \)
Plan I

Introduction

Time Petri Nets with Stopwatches (and more)

Abstractions for Scheduling-TPNs

Verifying properties
  Observers
  Model-checking

Conclusion and Future Work
Observing durations

\[ p_1 \quad p_3 \]
\[ t_1 [0, 4] \quad t_3 [5, 6] \]
\[ p_2 \quad p_4 \]
\[ t_2 [3, 4] \quad t_4 [0, 0] \]
\[ p_{obs} \]
\[ t_{obs} [5, 5] \]
Verifying properties
Observers

Observing durations

There can be $\geq 5$ t.u. between the firings of $t_1$ and $t_2$ iff $t_{obs}$ is fired.
Observing durations

\[ p_1 \xrightarrow{t_1} [0, 4] \xrightarrow{p_2} t_2 [3, 4] \xrightarrow{p_4} t_{obs} [5, 5] \]

\[ p_3 \xrightarrow{t_3} [5, 6] \xrightarrow{p_4} \]

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 $\text{AG}(M(p_2) \geq 1) \Rightarrow \text{EF}[0, 5](M(p_2) = 0))$ (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 in a 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 $\text{AG}(M(p_2) \geq 1) \Rightarrow \text{EF}[0, 5](M(p_2) = 0))$ (bounded response).
- This is implemented in a ROMEO!
  (http://romeo.rts-software.org)
- **Warning**: It cannot express properties on stopwatches (response times but not cumulated execution durations).
Plan I

Introduction

Time Petri Nets with Stopwatches (and more)

Abstractions for Scheduling-TPNs

Verifying properties

Conclusion and Future Work
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.
Conclusion and Future Work

B. Berthomieu and M. Diaz.
Modeling and verification of time dependent systems using time Petri nets.

Reachability problems and abstract state spaces for time petri nets with stopwatches.

Time state space analysis of real-time preemptive systems.

G.B. Dantzig.
Linear programming and extensions.
*IEICE Transactions on Information and Systems*, 1963.
Conclusion and Future Work

State space computation and analysis of time Petri nets.

Guillaume Gardey.
*Réseaux de Petri temporels.*

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.
Conclusion and Future Work

N.D. Jones, L.H. Landweber, and Y.E. Lien.
Complexity of some problems in Petri nets.

D. Lime and O.H. Roux.
Expressiveness and analysis of scheduling extended time Petri nets.

O.H. Roux and D. Lime.
Time Petri nets with inhibitor hyperarcs. Formal semantics and state space computation.