Preserving Partial Order Runs in Parametric Time Petri Nets

Étienne André¹, Thomas Chatain², César Rodríguez¹

¹LIPN, Université Paris 13, France
²LSV, ENS Cachan & CNRS, INRIA Saclay, France

ACSD, Bruxelles, 26 June 2015
Parametric Timed Systems

Timed models:
- Time bounds for firing delays
- Time/timed Petri nets, timed automata, ...

Challenges:
- Exact delay unknown, but known to be in [60, 70]
- Robustness: safe if we check with delay 50 and implement 50.01?
- Improve latencies: until what value can 10 be decreased?

Parameter synthesis:
- Consider that timing constants are unknown constants: parameters
- Find valuations such that something good happens
Example

Reference valuation: \( \langle p_1, p_2 \rangle := \langle 6, 7 \rangle \)

Questions:

▶ Can we increase \( p_1, p_2 \) while preserving the reference behaviour?

▶ Safe if we implement \( \langle p_1, p_2 \rangle = \langle 6.99, 2.99 \rangle \)?

▶ Can we increase \( p_2 \) at the cost of decreasing \( p_1 \)?
Example

Reference valuation: \( \langle p_1, p_2 \rangle := \langle 7, 3 \rangle \)
Example

Reference valuation: \( \langle p_1, p_2 \rangle := \langle 7, 3 \rangle \)

Questions:

- Can we increase \( p_1, p_2 \) while preserving the reference behaviour?
- Safe if we implement \( \langle p_1, p_2 \rangle = \langle 6.99, 2.99 \rangle \)?
- Can we increase \( p_2 \) at the cost of decreasing \( p_1 \)?
Synthesis Problems

Large class of *synthesis problems*:

---

**Reachability Synthesis ("EF-synthesis")** [Alur, Henzinger, Vardi 1993]

*Given*

- system model $M$ over parameters $P := \{p_1, \ldots, p_n\}$, and
- a set $U$ of states,

find parameter valuations $\nu : P \rightarrow \mathbb{R}$ such that

- $U$ is unreachable in $M[\nu]$.

* where $M[\nu]$ is like $M$ with parameter $p \in P$ substituted by $\nu(p)$. 
Synthesis Problems

Large class of synthesis problems:

Inverse Method [André, Chatain, Encrenaz, Fribourg 2009]

Given

- system model $M$ over parameters $P := \{p_1, \ldots, p_n\}$, and
- a reference parameter valuation $v_0$,

find parameter valuations $v: P \rightarrow \mathbb{R}$ such that

* the sequential, untimed behaviour of $M[v]$ equals $M[v_0]$.

* where $M[v]$ is like $M$ with parameter $p \in P$ substituted by $v(p)$. 
Synthesis Problems

Large class of synthesis problems:

This paper

Given

- system model $M$ over parameters $P := \{p_1, \ldots, p_n\}$, and
- a reference parameter valuation $v_0$,

find parameter valuations $v : P \rightarrow \mathbb{R}$ such that
- the sequential partially-ordered, untimed behaviour of $M[v]$ equals $M[v_0]$.

* where $M[v]$ is like $M$ with parameter $p \in P$ substituted by $v(p)$. 
Motivation – Limitations of the Inverse Method

Problem: Specific time delay forces unnecessary sequentialization of gates.

Solution: Preserve partial order rather than sequential executions!
Motivation – Limitations of the Inverse Method

Problem: Specific time delay forces unnecessary sequentialization of gates.
Solution: Preserve partial-order rather than sequential executions!
Motivation – Limitations of the Inverse Method

Problem: Specific time delay forces unnecessary sequentialization of gates.

Solution: Preserve partially-order rather than sequential executions!
Motivation – Limitations of the Inverse Method

Problem: Specific time delay forces unnecessary sequentialization of gates.

Solution: Preserve partially-order rather than sequential executions!
Motivation – Limitations of the Inverse Method

Problem: Specific time delay forces unnecessary sequentialization of gates.

Solution: Preserve partially-order rather than sequential executions!

\[ \begin{align*}
N_1 & \quad [p_1, p_2] \\
N_2 & \quad [p_3, p_4] \\
I_1 & \quad 1 \quad 0 \\
I_2 & \quad 0 \quad 1 \\
And & \quad 0 \quad 1 \\
Q & \quad 0 \\
\end{align*} \]
Motivation – Limitations of the Inverse Method

Problem: Specific time delay forces unnecessary sequentialization of Not gates
Solution: Preserve partially-order rather than sequential executions!
Outline

Introduction

Time Petri nets

Process Semantics

Synthesis Procedure

Conclusion
Safe Time Petri Nets (TPN): Definition

- Specification of real-time distributed systems
- Strong time semantics (urgency)
- We consider safe time Petri nets

\[(P, T, pre, post, efd, lfd, M_0)\]

- \(\bullet t := pre(t) \subseteq P\)
- \(t^* := post(t) \subseteq P\)
- earliest firing delay: \(efd : T \rightarrow \mathbb{N}\)
- latest firing delay: \(lfd : T \rightarrow \mathbb{N} \cup \{\infty\}\)
- initial marking: \(M_0 \subseteq P\)

![Petri Net Diagram]
Safe Time Petri Nets: Semantics (Runs)

**Definition**

\[ \text{runs}(N) := \text{set of timed executions in } (T \times \mathbb{R})^* \]

\[ \text{untimed-runs}(N) := \text{strip-time}(\text{runs}(N)) \]
Safe Time Petri Nets: Semantics (Runs)

Definition

\[
\text{runs}(N) := \text{set of timed executions in } (T \times \mathbb{R})^* \\
\text{untimed-runs}(N) := \text{strip-time}(\text{runs}(N))
\]
Safe Time Petri Nets: Semantics (Runs)

Current time: 2
Possible run: \((a, 2)\)

Definition

\[
\text{runs}(N) := \text{set of timed executions in } (T \times \mathbb{R})^* \\
\text{untimed-runs}(N) := \text{strip-time}(\text{runs}(N))
\]
Safe Time Petri Nets: Semantics (Runs)

Definition

\[
\text{runs}(N) := \text{set of timed executions in } (T \times \mathbb{R})^* \\
\text{untimed-runs}(N) := \text{strip-time}(\text{runs}(N))
\]
Safe Time Petri Nets: Semantics (Runs)

Current time: 3
Possible run: (a, 2), (c, 3)

Definition

\[
\text{runs}(N) := \text{set of timed executions in } (T \times \mathbb{R})^* \\
\text{untimed-runs}(N) := \text{strip-time}(\text{runs}(N))
\]
Safe Time Petri Nets: Semantics (Runs)

Current time: 6
Possible run: \((a, 2), (c, 3), (b, 6)\)

**Definition**

\[
\text{runs}(N) := \text{set of timed executions in } (T \times \mathbb{R})^* \\
\text{untimed-runs}(N) := \text{strip-time(\text{runs}(N))}
\]
Safe Time Petri Nets: Semantics (Runs)

Current time: 6
Possible run: \((a, 2), (c, 3), (b, 6)\)
Untimed run: \(a, c, b\)

**Definition**

\[
\text{runs}(N) := \text{set of timed executions in } (T \times \mathbb{R})^* \\
\text{untimed-runs}(N) := \text{strip-time}(\text{runs}(N))
\]
Processes (Partially-ordered Runs)

- Events ordered by causality

\[ (a,2), (c, 3), (b, 6) \]
Processes (Partially-ordered Runs)

- Events ordered by causality

- \( (a, 2) \)
- \( (c, 3) \)
- \( (b, 6) \)

\[ \begin{align*}
\text{Processes (Partially-ordered Runs)} \\
\text{Introduction} & \quad \text{Time Petri nets} & \quad \text{Process Semantics} & \quad \text{Synthesis Procedure} & \quad \text{Conclusion} \\
\end{align*} \]
Processes (Partially-ordered Runs)

- Events ordered by causality

\[ [0, \infty), [0, 5), [3, 4), [2, 4) \]

- Definition: processes \((N) := \text{partial-order-of}(\text{runs}(N))\)

- Untimed processes: \(\text{untimed-processes}(N) := \text{strip-time}(\text{processes}(N))\)
Processes (Partially-ordered Runs)

Events ordered by causality
Processes (Partially-ordered Runs)

Events ordered by causality
Processes (Partially-ordered Runs)

Events ordered by causality

Definition

\[
\text{processes}(N) := \text{partial-order-of}(\text{runs}(N))
\]

\[
\text{untimed-processes}(N) := \text{strip-time}(\text{processes}(N))
\]
Processes (Partially-ordered Runs)

Events ordered by causality

Definition

\[
\text{processes}(N) := \text{partial-order-of(} \text{runs}(N) \text{)}
\]

\[
\text{untimed-processes}(N) := \text{strip-time(} \text{processes}(N) \text{)}
\]
Parametric TPNs and Parameter Valuations

Let $P := \{p_1, \ldots, p_n\}$ be a set of parameters:

- A **parametric TPN** over $P$ is a TPN $\mathcal{N}$ with delay bounds over $\mathbb{N} \cup \{\infty\} \cup P$
- A **parameter valuation** is a function $\nu : P \rightarrow \mathbb{R}$.
- Notation $\mathcal{N}[\nu]$ denotes parameter substitution:

![Diagram](image-url)

with $\nu(p_1) = 5$ and $\nu(p_2) = 3$
Inverse Method with Partial Orders (IMPO)

The problem

Given

- a parametric TPN \(\mathcal{N}\) over parameters \(P\), and
- a reference parameter valuation \(v_0\),

find parameter valuations \(v: P \rightarrow \mathbb{R}\) (i.e., a constraint over \(P\)) such that

\[
\text{untimed-processes}(\mathcal{N}[v]) \subseteq \text{untimed-processes}(\mathcal{N}[v_0]).
\]

Our solution in a nutshell

1. Enumerate all untimed processes of (the support of) \(\mathcal{N}\)
2. To each of them, associate a linear constraint \(K\) over \(P\)
3. Select those which \(\mathcal{N}[v_0]\) cannot produce
4. Return linear constraint over \(P\) forbidding exactly those processes
**Step 1: Enumerate all Untimed Processes**

A. Compute the unfolding of the untimed support of $\mathcal{N}$

B. Extract the set of maximal processes

**Example**

Our circuit:

<table>
<thead>
<tr>
<th>Input</th>
<th>Process 1</th>
<th>Process 2</th>
<th>And</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>$N_1$</td>
<td>$[p_1, p_2]$</td>
<td>0</td>
<td>$[p_5, p_6]$</td>
</tr>
<tr>
<td>$I_2$</td>
<td>$N_2$</td>
<td>$[p_3, p_4]$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Reference valuation:

- $[p_1, p_2] := [6, 7]$
- $[p_3, p_4] := [2, 3]$
- $[p_5, p_6] := [1, 2]$

Its unfoldings: next slide . . .
Step 1: Example: the Unfolding

Maximal processes:

\[ \mathcal{R}_1 = \{e_1, e_4, e_2, e_5\} \]
\[ \mathcal{R}_2 = \{e_1, e_4, e_2, e_8, e'_5, e_9\} \]
Step 2: Linear Constraint for each Process

For each untimed process \( \mathcal{R} \), we build a linear constraint \( K_{\mathcal{R}} \) over \( P \):

- \( K_{\mathcal{R}} \): parameter valuations \( \nu \) such that \( \mathcal{R} \) is a process of \( N[\nu] \)

A. Build constraint \( K'_{\mathcal{R}} \) over some new date variables \( Z \) and parameters \( P \)

B. Eliminate the date variables \( Z \) from \( K'_{\mathcal{R}} \)
Step 2.A: Symbolic Characterization of Processes  [AL00]

Two timed runs may have the same untimed process:

(a,2), (c,3), (b,6)

(a,1), (c,3), (b,6)

\[\text{[T. Aura, J. Lilius 2000]}\]
Step 2.A: Symbolic Characterization of Processes  [AL00]

- …so we represent symbolically dates of processes sharing same partial-order
- 1 variable $z_i$ per event date
Step 2.A: Symbolic Characterization of Processes [AL00]

- ...so we represent symbolically dates of processes sharing same partial-order
- 1 variable $z_i$ per event date

1. For every event $e$:
   $efd(e) \leq z_e - doe(e) \leq lfd(e)$

```
   p1
   □ (a, z_1)
   p3
   □ (b, z_3)
   p1

   p1
   □ (a, z_1)
   p3
   □ (b, z_3)
   p1

   p2
   □ (c, z_2)
   p4
   □ (c, z_2)
   p4
```

$p_2$
Step 2.A: Symbolic Characterization of Processes [AL00]

- ...so we represent symbolically dates of processes sharing same partial-order
- 1 variable $z_i$ per event date

1. For every event $e$:
   
   $$\text{efd}(e) \leq z_e - \text{doe}(e) \leq \text{lf}(e)$$

- $0 \leq z_1 \leq \infty$
- $3 \leq z_2 \leq 4$
- $0 \leq z_3 - \max\{z_1, z_2\} \leq 5$
Step 2.A: Symbolic Characterization of Processes  [AL00]

- so we represent symbolically dates of processes sharing same partial-order
- 1 variable $z_i$ per event date

1. For every event $e$:
   $$ efd(e) \leq z_e - doe(e) \leq lfd(e) $$

2. For every transition $t$ enabled in the final state:
   $$ z_{\text{end}} \leq doe(t) + lfd(t) $$

![Diagram of a Petri net with time constraints]
Step 2.A: Symbolic Characterization of Processes [AL00]

- ...so we represent symbolically dates of processes sharing same partial-order
- 1 variable \( z_i \) per event date

1. For every event \( e \):
   \[
efd(e) \leq z_e - doe(e) \leq lfd(e)
   \]

2. For every transition \( t \) enabled in the final state:
   \[
z_{\text{end}} \leq doe(t) + lfd(t)
   \]

\[
\begin{align*}
p_1 & \rightarrow (a, z_1) \\
p_3 & \rightarrow (b, z_3) \\
p_1 & \rightarrow (c, z_2)
\end{align*}
\]

- \( 0 \leq z_1 \leq \infty \)
- \( 3 \leq z_2 \leq 4 \)
- \( 0 \leq z_3 - \max \{ z_1, z_2 \} \leq 5 \)
- \( \max \{ z_1, z_2, z_3 \} \leq z_3 + \infty \)
- \( \max \{ z_1, z_2, z_3 \} \leq z_3 + 4 \)
Step 2.A: Symbolic Characterization of Processes \[\text{[AL00]}\]

- So we represent symbolically dates of processes sharing same partial-order
- 1 variable $z_i$ per event date

1. For every event $e$:
   \[ efd(e) \leq z_e - doe(e) \leq lfd(e) \]

2. For every transition $t$ enabled in the final state:
   \[ z_{\text{end}} \leq doe(t) + lfd(t) \]

3. No transition eventually enabled overtakes its latest firing delay

\begin{align*}
0 & \leq z_1 \leq \infty \\
3 & \leq z_2 \leq 4 \\
0 & \leq z_3 - \max \{z_1, z_2\} \leq 5 \\
\max \{z_1, z_2, z_3\} & \leq z_3 + \infty \\
\max \{z_1, z_2, z_3\} & \leq z_3 + 4
\end{align*}
Step 2.A: Symbolic Characterization of Processes  [AL00]

- so we represent symbolically dates of processes sharing same partial-order
- 1 variable $z_i$ per event date

1. For every event $e$:
   $$efd(e) \leq z_e - doe(e) \leq lfd(e)$$

2. For every transition $t$ enabled in the final state:
   $$z_{\text{end}} \leq doe(t) + lfd(t)$$

3. No transition eventually enabled overtakes its latest firing delay

- $0 \leq z_1 \leq \infty$
- $3 \leq z_2 \leq 4$
- $0 \leq z_3 - \max\{z_1, z_2\} \leq 5$
- $\max\{z_1, z_2, z_3\} \leq z_3 + \infty$
- $\max\{z_1, z_2, z_3\} \leq z_3 + 4$
Step 2.A: Symbolic Characterization of Processes [AL00]

- ...so we represent symbolically dates of processes sharing same partial-order
- 1 variable \( z_i \) per event date

1. For every event \( e \):
   \[
   efd(e) \leq z_e - doe(e) \leq lfd(e)
   \]

2. For every transition \( t \) enabled in the final state:
   \[
   z_{end} \leq doe(t) + lfd(t)
   \]

3. No transition eventually enabled overtakes its latest firing delay

\[0 \leq z_1 \leq \infty\]
\[3 \leq z_2 \leq 4\]
\[0 \leq z_3 - \max\{z_1, z_2\} \leq 5\]
\[\max\{z_1, z_2, z_3\} \leq z_3 + \infty\]
\[\max\{z_1, z_2, z_3\} \leq z_3 + 4\]
\[z_3 - z_1 \leq 4\]
Step 2.A: Processes of Parametric TPNs

For an untimed process $\mathcal{R}$ of a non-parametric TPN $\mathcal{N}$ we

- generate linear constraint $K'_{\mathcal{R}}$

over the variables:

- date variables $Z := \{z_1, \ldots, z_m\}$
Step 2.A: Processes of Parametric TPNs

For an untimed process $\mathcal{R}$ of a parametric TPN $\mathcal{N}$ we

- generate linear constraint $K'_\mathcal{R}$

over the variables:

- date variables $Z := \{z_1, \ldots, z_m\}$,
- system parameters $P := \{p_1, \ldots, p_n\}$. 
Step 2.A: Processes of Parametric TPNs

For an untimed process $\mathcal{R}$ of a parametric TPN $\mathcal{N}$ we

- generate linear constraint $K'_\mathcal{R}$

over the variables:

- date variables $Z := \{z_1, \ldots, z_m\}$,
- system parameters $P := \{p_1, \ldots, p_n\}$.

\begin{itemize}
    \item $0 \leq z_1 \leq \infty$
    \item $3 \leq z_2 \leq p_2$
    \item $0 \leq z_3 - \max\{z_1, z_2\} \leq p_1$
    \item $\max\{z_1, z_2, z_3\} \leq z_3 + \infty$
    \item $\max\{z_1, z_2, z_3\} \leq z_3 + p_2$
    \item $z_3 - z_1 \leq 4$
\end{itemize}
Parameter Synthesis Procedure

For a PTPN $\mathcal{N}$ over parameters $P$ (where all untimed processes are finite):

1. Enumerate all untimed processes of the support of $\mathcal{N}$
   A. Compute untimed unfolding of $\mathcal{N}$
   B. Extract set $\mathcal{M}$ of maximal processes

2. For every process $\mathcal{R} \in \mathcal{M}$,
   A. Build constraint $K'_\mathcal{R}$ over the date variables $Z$ and parameters $P$
   B. Obtain $K_\mathcal{R}$ from $K'_\mathcal{R}$ by eliminating date variables $Z$

3. Let $\mathcal{M}'$ be the set of processes $\mathcal{R}$ such that $\nu_0 \not\models K_\mathcal{R}$

4. Output constraint over $P$ forbidding $\mathcal{M}'$:

$$K_0 \land \bigwedge_{\mathcal{R} \in \mathcal{M}'} \neg K_\mathcal{R}$$
Steps 2, 3, 4: Example

Maximal processes:

\[ R_1 = \{ e_1, e_4, e_2, e_5 \} \]
\[ R_2 = \{ e_1, e_4, e_2, e_8, e'_5, e_9 \} \]
Steps 2, 3, 4: Example

\[ \mathcal{R}_1 = \{e_1, e_4, e_2, e_5\} \]

- Constraint \( K'_{\mathcal{R}_1} \):
  
  \[
  0 \leq z_1 \leq 1 \\
  0 \leq z_2 \leq 1 \\
  p_1 \leq z_4 - z_1 \leq p_2 \\
  p_3 \leq z_5 - z_2 \leq p_4 \\
  z_5 \leq z_4 + p_6
  \]

- After elimination of \( z_i \) variables:
  
  \[ p_3 \leq p_2 + p_6 + 1 \]

- Satisfied by \( v_0 \): ✗ (skipped)

\[ \mathcal{R}_2 = \{e_1, e_4, e_2, e_8, e'_5, e_9\} \]

- Constraint \( K'_{\mathcal{R}_2} \):
  
  \[
  0 \leq z_1 \leq 1 \\
  0 \leq z_2 \leq 1 \\
  p_1 \leq z_4 - z_1 \leq p_2 \\
  p_5 \leq z_8 - z_4 \leq p_6 \\
  p_3 \leq y'_5 - \max\{z_8, z_2\} \leq p_4 \\
  p_5 \leq z_9 - y'_5 \leq p_6 \\
  z_8 \leq z_2 + p_4
  \]

- After elimination of \( z_i \) variables:
  
  \[ p_1 + p_5 \leq p_4 + 1 \]

- Not satisfied by \( v_0 \): ✔ (taken)

Output: \( K_0 \wedge (N_1^- + A^- > N_2^+ + 1) \)

Preserving sequential executions requires stronger constraint \( N_1^- > N_2^+ + 1\)
Concluding Remarks

- Parameter synthesis preserving partial-order reference behaviour
- Enhanced (i.e. weaker) constraints than the Inverse Method
- Restriction: acyclic systems

Perspectives:

- Extension to non-acyclic systems
  - Cutoff events
  - Block-based restriction
- Non-convex constraints: strengthening or efficient implementation
- Implementation
Concluding Remarks

- Parameter synthesis preserving partial-order reference behaviour
- Enhanced (i.e. weaker) constraints than the Inverse Method
- Restriction: acyclic systems

Perspectives:

- Extension to non-acyclic systems
  - Cutoff events
  - Block-based restriction
- Non-convex constraints: strengthening or efficient implementation
- Implementation

Thank you!