Cost Problems for Parametric Time Petri Nets

Hanifa Boucheneb[1] **Didier Lime**[2] Olivier H. Roux[2] Charlotte Seidner[2]

[1]École Polytechnique de Montréal, Québec, Canada

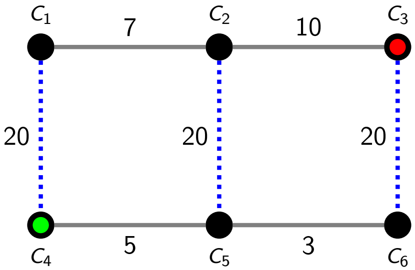
[2]Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France

SynCoP'22
2nd of April 2022, Munich, Germany

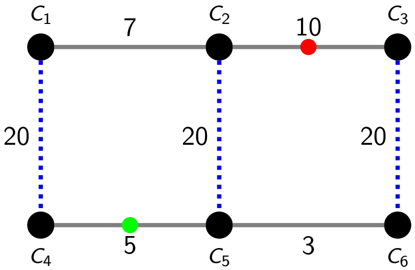
Plan

Plan

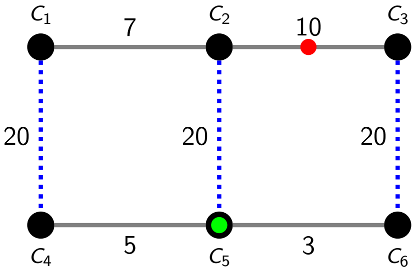
Meeting across the river



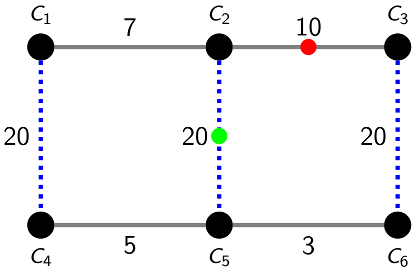
Meeting across the river



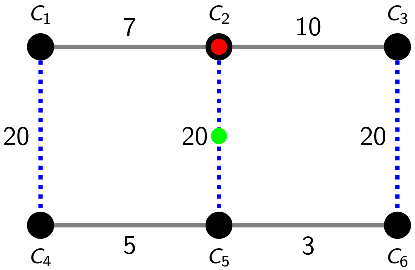
Meeting across the river



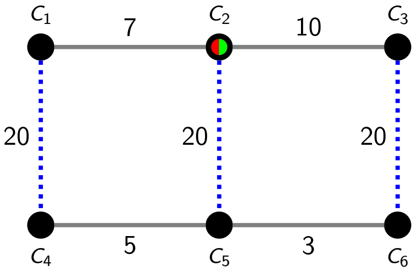
Meeting across the river



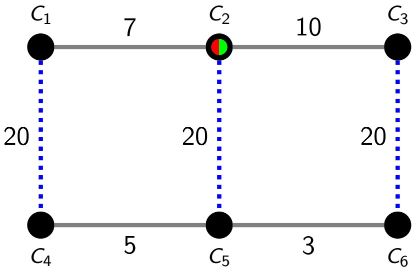
Meeting across the river



Meeting across the river

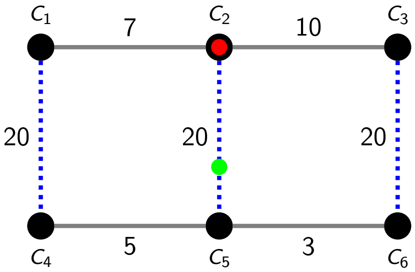


Meeting across the river



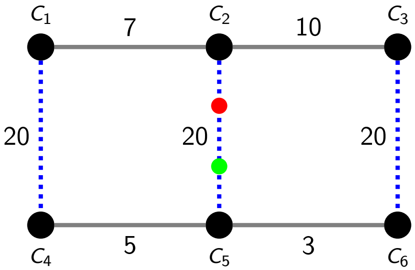
But that's not how it ended...

Meeting across the river

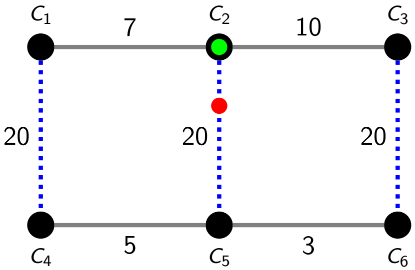


They won't wait!

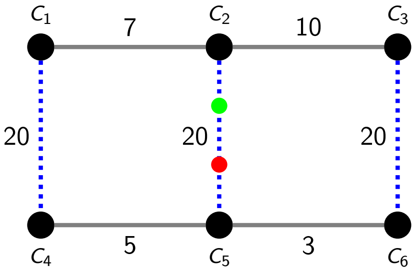
Meeting across the river



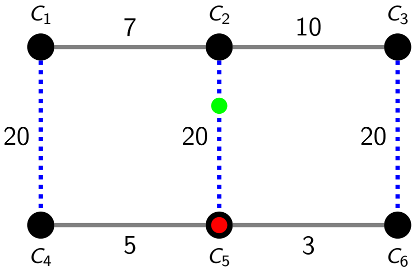
Meeting across the river



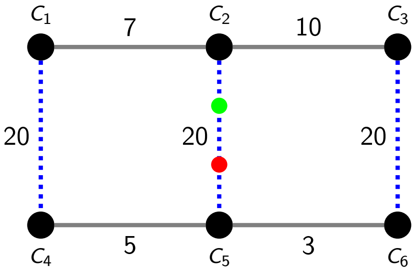
Meeting across the river



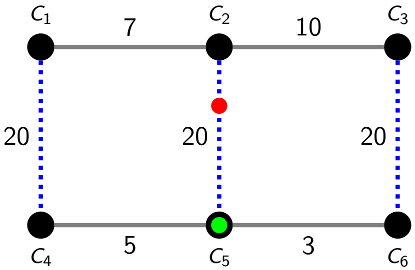
Meeting across the river



Meeting across the river



Meeting across the river



Is all hope lost?

Plan

Introduction

Time Petri Nets and State Classes

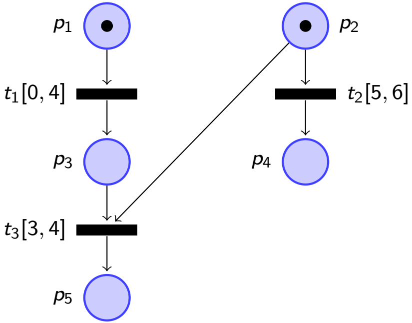
Costs in Time Petri Nets

Termination of the Infcost Algorithm

Parametric Cost Time Petri Nets

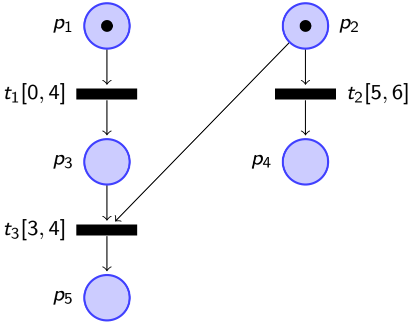
Conclusion

Time Petri Nets



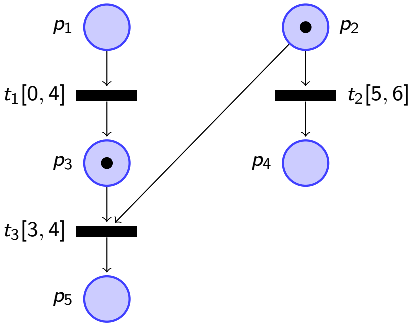
$t_1 \in [0, 4]$
$t_2 \in [5, 6]$

Time Petri Nets



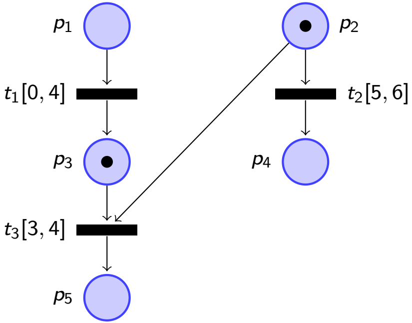
$$\begin{matrix} t_1 \in [0,4] \\ t_2 \in [5,6] \end{matrix} \xrightarrow{1.4} \begin{matrix} t_1 \in [0,2.6] \\ t_2 \in [3.6,4.6] \end{matrix}$$

Time Petri Nets



$$\begin{array}{l} t_1 \in [0,4] \\ t_2 \in [5,6] \end{array} \xrightarrow{1.4} \begin{array}{l} t_1 \in [0,2.6] \\ t_2 \in [3.6,4.6] \end{array} \xrightarrow{t_1} \begin{array}{l} t_2 \in [3.6,4.6] \\ t_3 \in [3,4] \end{array}$$

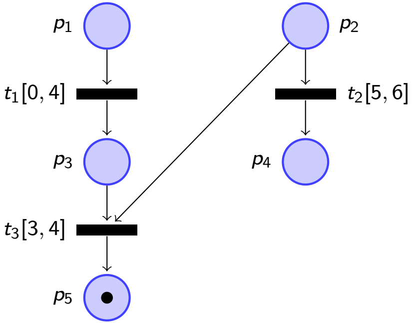
Time Petri Nets



$$\begin{array}{l} t_1 \in [0,4] \\ t_2 \in [5,6] \end{array} \xrightarrow{1.4} \begin{array}{l} t_1 \in [0,2.6] \\ t_2 \in [3.6,4.6] \end{array} \xrightarrow{t_1} \begin{array}{l} t_2 \in [3.6,4.6] \\ t_3 \in [3,4] \end{array}$$

$$\xrightarrow{3.6} \begin{array}{l} t_2 \in [0,1] \\ t_3 \in [0,0.4] \end{array}$$

Time Petri Nets



$$\begin{array}{l} t_1 \in [0,4] \\ t_2 \in [5,6] \end{array} \xrightarrow{1.4} \begin{array}{l} t_1 \in [0,2.6] \\ t_2 \in [3.6,4.6] \end{array} \xrightarrow{t_1} \begin{array}{l} t_2 \in [3.6,4.6] \\ t_3 \in [3,4] \end{array}$$

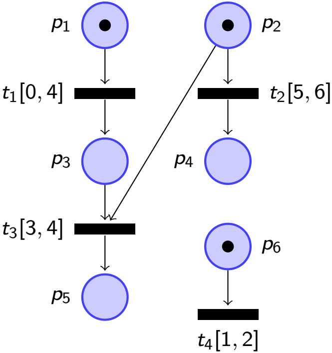
$$\xrightarrow{3.6} \begin{array}{l} t_2 \in [0,1] \\ t_3 \in [0,0.4] \end{array} \xrightarrow{t_3} \perp$$

State Classes[1]

- A **state class** C_σ is the (collapsed) set of states obtained by the transition sequence σ;
- Those states all share the same marking;
- The union of all points in the intervals in the valuations on the transitions can be represented by a **convex polyhedron** (encoded by a Difference Bound Matrix, DBM);
- A state class is thus a pair $C = (m, D)$, where m is a marking, and D a DBM.

[1] Berthomieu and Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE trans. on soft. eng.*, 17(3):259–273, 1991.

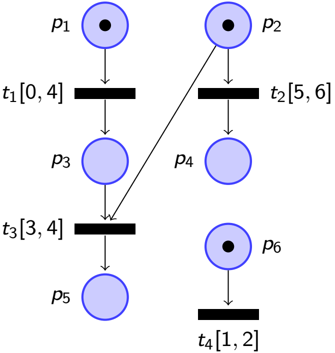
State Class Computation



Initially:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$

State Class Computation



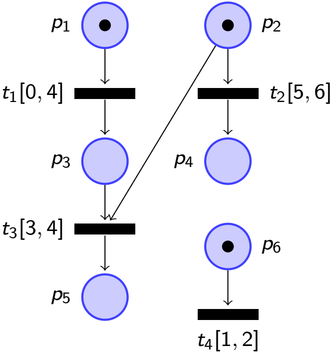
Initially:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$

Fire t_1:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$
$$\boldsymbol{\theta_1 \leq \theta_2}$$
$$\boldsymbol{\theta_1 \leq \theta_4}$$

State Class Computation



Initially:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$

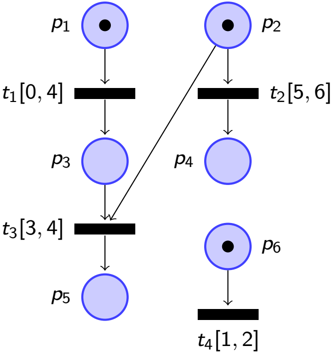
Fire t_1:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$
$$\theta_1 \leq \theta_2$$
$$\theta_1 \leq \theta_4$$

Change origin:

$$\theta_1 \in [0,4]$$
$$\theta_2' + \theta_1 \in [5,6]$$
$$\theta_4' + \theta_1 \in [1,2]$$
$$\theta_1 \leq \theta_2' + \theta_1$$
$$\theta_1 \leq \theta_4' + \theta_1$$

State Class Computation



Initially:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$

Fire t_1:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$
$$\theta_1 \leq \theta_2$$
$$\theta_1 \leq \theta_4$$

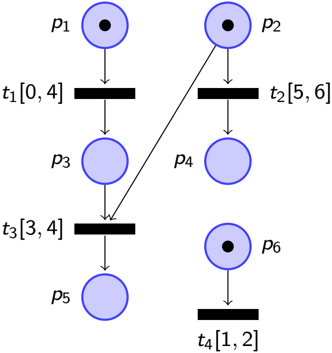
Change origin:

$$\theta_1 \in [0,4]$$
$$\theta_2' + \theta_1 \in [5,6]$$
$$\theta_4' + \theta_1 \in [1,2]$$
$$\theta_1 \leq \theta_2' + \theta_1$$
$$\theta_1 \leq \theta_4' + \theta_1$$

Eliminate disabled:

$$\theta_2' \in [3,6]$$
$$\theta_4' \in [0,2]$$
$$\theta_2' - \theta_4' \in [3,5]$$

State Class Computation



Initially:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$

Fire t_1:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$
$$\boldsymbol{\theta_1 \leq \theta_2}$$
$$\boldsymbol{\theta_1 \leq \theta_4}$$

Change origin:

$$\theta_1 \in [0,4]$$
$$\boldsymbol{\theta_2' + \theta_1} \in [5,6]$$
$$\boldsymbol{\theta_4' + \theta_1} \in [1,2]$$
$$\theta_1 \leq \boldsymbol{\theta_2' + \theta_1}$$
$$\theta_1 \leq \boldsymbol{\theta_4' + \theta_1}$$

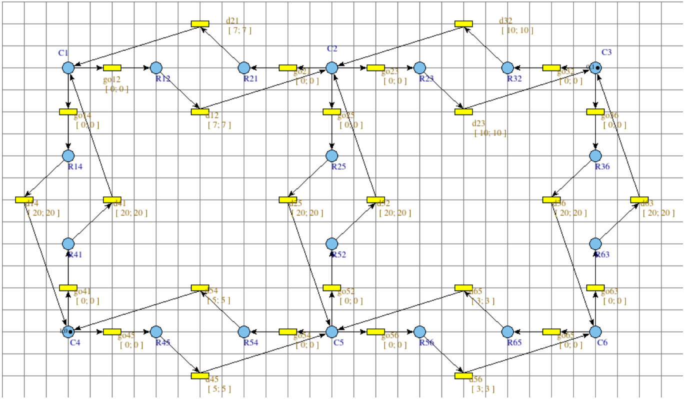
Eliminate disabled:

$$\theta_2' \in [3,6]$$
$$\theta_4' \in [0,2]$$
$$\theta_2' - \theta_4' \in [3,5]$$

Add newly enabled:

$$\theta_2' \in [3,6]$$
$$\boldsymbol{\theta_3 \in [3,4]}$$
$$\theta_4' \in [0,2]$$
$$\theta_2' - \theta_4' \in [3,5]$$

Meeting across the river




```
Checking property EF (C1[0] + C1[1] == 2 or C2[0] + C2[1] == 2 or C3[0] + C3[1] == 2 or C4[0] + C4[1] == 2  or C5[0] + C5[1] == 2  or C6[0] + C6[1] ==
2) on TPN: /home/did/Desktop/romeo-3.8.5/meeting-simple2-noparam.xml
Waiting for response (kill the romeo-cli process to interrupt)...
true

Trace: go41__0, go32__1, d32__1, go25__1, d41__0, go14__0, d25__1, go54__1, d54__1, go41__1, d14__0, go41__0, d41__1, go14__1, d41__0, go12__0, d12__0,
go25__0, d14__1, go41__1, d25__0, go54__0, d54__0, go41__0, d41__1, go14__1, d41__0, go14__0, d14__1, go45__1, d45__1, go54__1, d54__1, go41__1, d14__0,
go41__0, d41__1, go14__1, d41__0, go12__0, d12__0, go25__0, d14__1, go41__1, d25__0, go54__0, d54__0, go41__0, d41__1, go14__1, d41__0, go14__0, d14__1,
go45__1, d45__1, go54__1, d54__1, go41__1, d14__0, go41__0, d41__1, go14__1, d41__0, go12__0, d12__0, go25__0, d14__1, go41__1, d25__0, go54__0, d41__1,
go14__1, d54__0, go41__0, d14__1, go41__1, d41__0, go14__0, d41__1, go12__1, d14__0, go41__0, d12__1, go25__1, d41__0, go12__0, d25__1, go54__1, d12__0,
go21__0, d54__1, go41__1, d21__0, go14__0, d41__1, go14__1, d14__0, go41__0, d14__1, go45__1, d41__0, go14__0, d45__1, go54__1, d54__1, go41__1, d14__0,
go41__0, d41__1, go14__1, d41__0, go12__0, d14__1, d12__0, go25__0, go41__1, d25__0, d41__1, go14__1, go54__0, d54__0, go41__0, d14__1, go41__1, d41__0,
go14__0, d41__1, go12__1, d14__0, go41__0, d12__1, go25__1, d41__0, go12__0, d25__1, go52__1, d12__0, go25__0, d52__1, go25__1, d25__0, go54__0, d54__0,
go41__0, d25__1, go52__1, d41__0, go12__0, d12__0, go25__0, d52__1, go25__1, d25__0, go54__0, d25__1, go54__1, d54__0, go41__0, d54__1, go41__1, d41__0,
go14__0, d41__1, go14__1, d14__0, go45__0, d14__1, go41__1, d45__0, go52__0, d41__1, go14__1, d52__0, go25__0, d14__1, go45__1, d25__0, go52__0, d45__1,
go54__1, d54__1, go41__1, d52__0, go25__0, d41__1, go14__1, d25__0, go56__0, d56__0, go63__0, d14__1, go41__1, d63__0, go32__0, d41__1, go14__1, d32__0,
go25__0, d14__1, go41__1, d25__0, go54__0, d54__0, go41__0, d41__1, go14__1, d41__0, go14__0, d14__1, go45__1, d45__1, go54__1, d54__1, d14__0
```

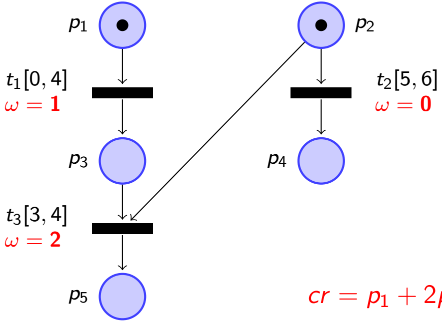
Meeting optimally across the river

What is an **optimal** (common) strategy to meet?

- Pay 1 for each move;
- Pay 1 for each time unit they wait
  - either while being idle in a city
  - or globally until they meet

Plan

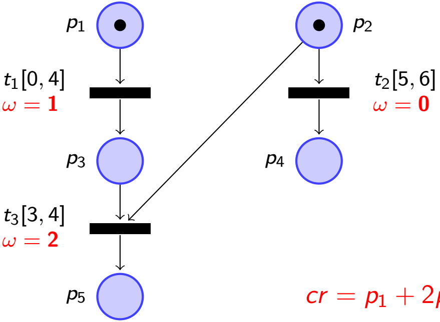
Cost Time Petri Nets



$cr = p_1 + 2p_2$

$t_1 \in [0,4]$
$t_2 \in [5,6]$
$\text{cost} = \mathbf{0}$

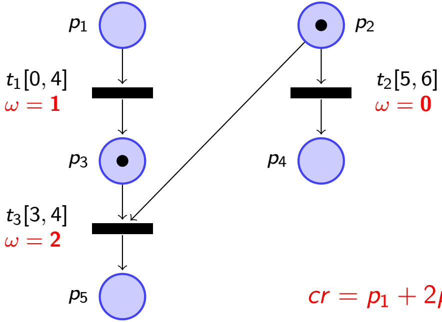
Cost Time Petri Nets



$cr = p_1 + 2p_2$

$$\begin{array}{l} t_1 \in [0,4] \\ t_2 \in [5,6] \\ \text{cost} = \mathbf{0} \end{array} \xrightarrow{1.4} \begin{array}{l} t_1 \in [0,2.6] \\ t_2 \in [3.6,4.6] \\ \text{cost} = (\mathbf{1}+\mathbf{2}) * \mathbf{1.4} = \mathbf{4.2} \end{array}$$

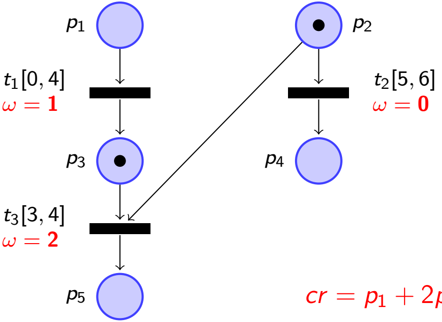
Cost Time Petri Nets



$cr = p_1 + 2p_2$

$$\begin{array}{l} t_1 \in [0,4] \\ t_2 \in [5,6] \\ \text{cost} = \mathbf{0} \end{array} \xrightarrow{1.4} \begin{array}{l} t_1 \in [0,2.6] \\ t_2 \in [3.6,4.6] \\ \text{cost} = (\mathbf{1}+\mathbf{2}) * \mathbf{1.4} = \mathbf{4.2} \end{array} \xrightarrow{t_1} \begin{array}{l} t_2 \in [3.6,4.6] \\ t_3 \in [3,4] \\ \text{cost} = \mathbf{4.2} + \mathbf{1} = \mathbf{5.2} \end{array}$$

Cost Time Petri Nets

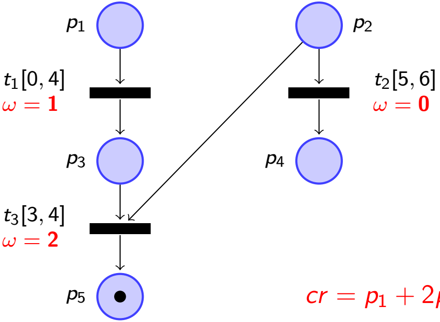


$cr = p_1 + 2p_2$

$$\begin{array}{l} t_1 \in [0,4] \\ t_2 \in [5,6] \\ \text{cost} = \mathbf{0} \end{array} \xrightarrow{1.4} \begin{array}{l} t_1 \in [0,2.6] \\ t_2 \in [3.6,4.6] \\ \text{cost} = (\mathbf{1}+\mathbf{2}) * \mathbf{1.4} = \mathbf{4.2} \end{array} \xrightarrow{t_1} \begin{array}{l} t_2 \in [3.6,4.6] \\ t_3 \in [3,4] \\ \text{cost} = \mathbf{4.2} + \mathbf{1} = \mathbf{5.2} \end{array}$$

$$\xrightarrow{3.6} \begin{array}{l} t_2 \in [0,1] \\ t_3 \in [0,0.4] \\ \text{cost} = \mathbf{5.2} + \mathbf{2} * \mathbf{3.6} = \mathbf{12.4} \end{array}$$

Cost Time Petri Nets



$cr = p_1 + 2p_2$

$$\begin{array}{l} t_1 \in [0,4] \\ t_2 \in [5,6] \\ \text{cost} = \mathbf{0} \end{array} \xrightarrow{1.4} \begin{array}{l} t_1 \in [0,2.6] \\ t_2 \in [3.6,4.6] \\ \text{cost} = (\mathbf{1}+\mathbf{2}) * \mathbf{1.4} = \mathbf{4.2} \end{array} \xrightarrow{t_1} \begin{array}{l} t_2 \in [3.6,4.6] \\ t_3 \in [3,4] \\ \text{cost} = \mathbf{4.2} + \mathbf{1} = \mathbf{5.2} \end{array}$$

$$\xrightarrow{3.6} \begin{array}{l} t_2 \in [0,1] \\ t_3 \in [0,0.4] \\ \text{cost} = \mathbf{5.2} + \mathbf{2} * \mathbf{3.6} = \mathbf{12.4} \end{array} \xrightarrow{t_3} \begin{array}{l} \perp \\ \text{cost} = \mathbf{12.4} + \mathbf{2} = \mathbf{14.4} \end{array}$$

The min/inf-cost Reachability Problem

Inf-cost reachability

Given a Cost-TPN $\mathcal{N}$ and a set of markings Goal, decide if Goal is reachable and if so compute:

$$\inf_{\rho \text{ s.t. } \mathsf{last}(\rho) \in \mathsf{Goal}} \mathsf{cost}(\rho)$$

Symbolic Algorithm for Inf-cost Reachability [2,3]

$\textsc{Cost} \leftarrow \infty$
$\textsc{Passed} \leftarrow \emptyset$
$\textsc{Waiting} \leftarrow \{(m_0, D_0)\}$
while $\textsc{Waiting} \neq \emptyset$ **do**
    select $C_\sigma = (m, D)$ from $\textsc{Waiting}$
    if $m \in$ Goal **and** $\mathrm{cost}(\mathbf{C}_\sigma) < \textsc{Cost}$ **then**
        $\textsc{Cost} \leftarrow \mathrm{cost}(C_\sigma)$
    end if
    if for all $C' \in \textsc{Passed}$, $C_\sigma \not\preceq C'$ **then**
        add C_σ to $\textsc{Passed}$
        for all $t \in \mathrm{firable}(C_\sigma)$, add $C_{\sigma.t}$ to $\textsc{Waiting}$
    end if
end while
return $\textsc{Cost}$

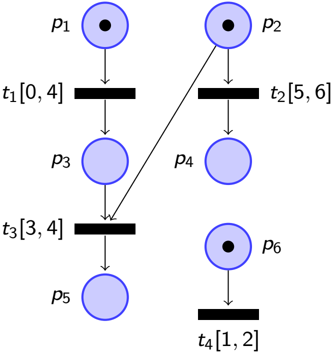
[2] Larsen et al. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *CAV'01*, 2001.

[3] Rasmussen et al. On using priced timed automata to achieve optimal scheduling. *FMSD*, 29(1):97–114, 2006.

Cost State Classes

- From a transition sequence σ, we want to compute cost(σ) the inf-cost of all runs built on σ
- We extend state class firing domains with a new variable c:
  $\Rightarrow$ Cost state classes: $C_\sigma = (m, D)$
- When firing t_i, c changes by $\omega(t_i) + \theta_i * cr(m)$
- An extended firing domain D is not a DBM anymore but still a **convex polyhedron**;
- $\mathrm{cost}(\sigma) = \mathrm{cost}(C_\sigma) = \inf_{(\vec{\theta},c)\in D} c$ computable using **linear programming**.

Cost State Classes: Example

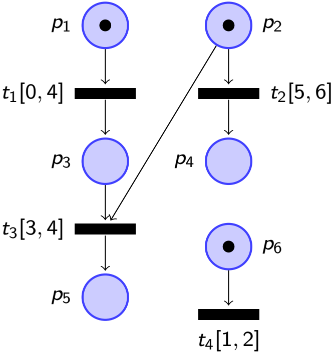


$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

Initially:

$\theta_1 \in [0,4]$
$\theta_2 \in [5,6]$
$\theta_4 \in [1,2]$
$\mathbf{c \geq 0}$

Cost State Classes: Example



$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

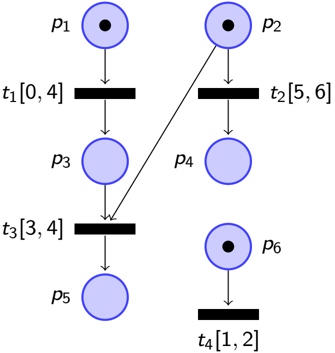
Initially:

$$\begin{aligned} &\theta_1 \in [0,4] \\ &\theta_2 \in [5,6] \\ &\theta_4 \in [1,2] \\ &\mathbf{c \geq 0} \end{aligned}$$

Fire t_1:

$$\begin{aligned} &\theta_1 \in [0,4] \\ &\theta_2 \in [5,6] \\ &\theta_4 \in [1,2] \\ &c \geq 0 \\ &\theta_1 \leq \theta_2 \\ &\theta_1 \leq \theta_4 \end{aligned}$$

Cost State Classes: Example



$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

Initially:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$
$$\mathbf{c \geq 0}$$

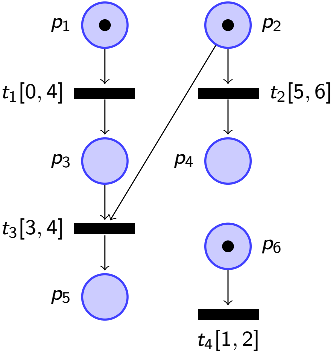
Fire t_1:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$
$$c \geq 0$$
$$\theta_1 \leq \theta_2$$
$$\theta_1 \leq \theta_4$$

Change origin:

$$\theta_1 \in [0,4]$$
$$\theta_2' + \theta_1 \in [5,6]$$
$$\theta_4' + \theta_1 \in [1,2]$$
$$\mathbf{c' - \omega(t_1) - cr(m_0) * \theta_1} \geq 0$$
$$\theta_1 \leq \theta_2' + \theta_1$$
$$\theta_1 \leq \theta_4' + \theta_1$$

Cost State Classes: Example



$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

Initially:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$
$$\mathbf{c \geq 0}$$

Fire t_1:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [1,2]$$
$$c \geq 0$$
$$\theta_1 \leq \theta_2$$
$$\theta_1 \leq \theta_4$$

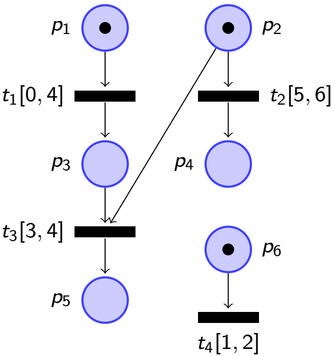
Change origin:

$$\theta_1 \in [0,4]$$
$$\theta_2' + \theta_1 \in [5,6]$$
$$\theta_4' + \theta_1 \in [1,2]$$
$$\mathbf{c' - \omega(t_1) - cr(m_0) * \theta_1} \geq 0$$
$$\theta_1 \leq \theta_2' + \theta_1$$
$$\theta_1 \leq \theta_4' + \theta_1$$

Eliminate disabled:

$$\theta_2' \in [3,6]$$
$$\theta_4' \in [0,2]$$
$$\theta_2' - \theta_4' \in [3,5]$$
$$\mathbf{c \geq 16 - 3\theta_2'}$$
$$\mathbf{c \geq 4 - 3\theta_4'}$$
$$\mathbf{c \geq 1}$$

Cost State Classes: Example



$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

Initially:

$$\begin{aligned} &\theta_1 \in [0,4] \\ &\theta_2 \in [5,6] \\ &\theta_4 \in [1,2] \\ &\mathbf{c \geq 0} \end{aligned}$$

Fire t_1:

$$\begin{aligned} &\theta_1 \in [0,4] \\ &\theta_2 \in [5,6] \\ &\theta_4 \in [1,2] \\ &c \geq 0 \\ &\theta_1 \leq \theta_2 \\ &\theta_1 \leq \theta_4 \end{aligned}$$

Change origin:

$$\begin{aligned} &\theta_1 \in [0,4] \\ &\theta'_2 + \theta_1 \in [5,6] \\ &\theta'_4 + \theta_1 \in [1,2] \\ &\mathbf{c' - \omega(t_1) - cr(m_0)} * \boldsymbol{\theta_1} \geq 0 \\ &\theta_1 \leq \theta'_2 + \theta_1 \\ &\theta_1 \leq \theta'_4 + \theta_1 \end{aligned}$$

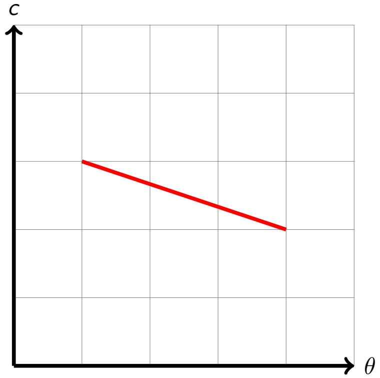
Eliminate disabled:

$$\begin{aligned} &\theta'_2 \in [3,6] \\ &\theta'_4 \in [0,2] \\ &\theta'_2 - \theta'_4 \in [3,5] \\ &\mathbf{c \geq 16 - 3}\boldsymbol{\theta'_2} \\ &\mathbf{c \geq 4 - 3}\boldsymbol{\theta'_4} \\ &\mathbf{c \geq 1} \end{aligned}$$

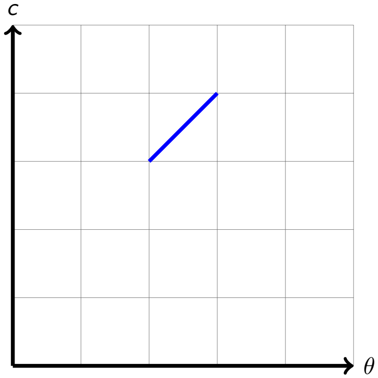
Add newly enabled:

$$\begin{aligned} &\theta'_2 \in [3,6] \\ &\theta_3 \in [3,4] \\ &\theta'_4 \in [0,2] \\ &\theta'_2 - \theta'_4 \in [3,5] \\ &c \geq 16 - 3t'_2 \\ &c \geq 4 - 3t'_4 \\ &c \geq 1 \end{aligned}$$

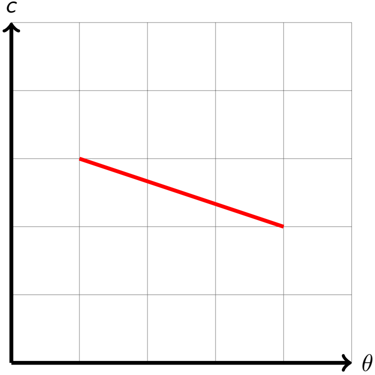
Cost State Class Subsumption: $\preccurlyeq$



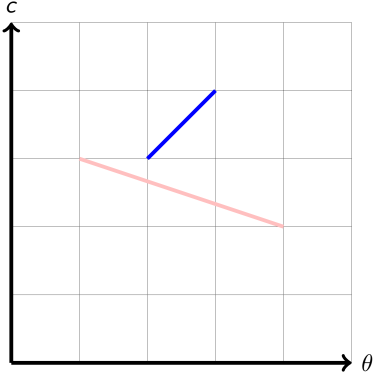
$\succcurlyeq$



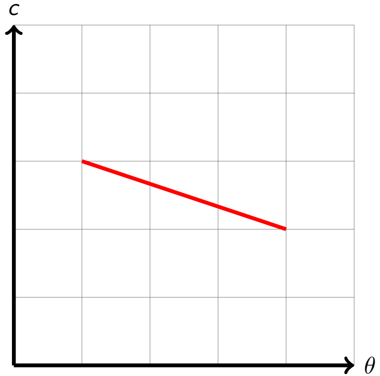
Cost State Class Subsumption: $\preccurlyeq$



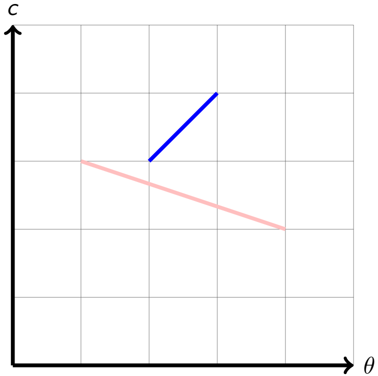
$\succcurlyeq$



Cost State Class Subsumption: $\preccurlyeq$

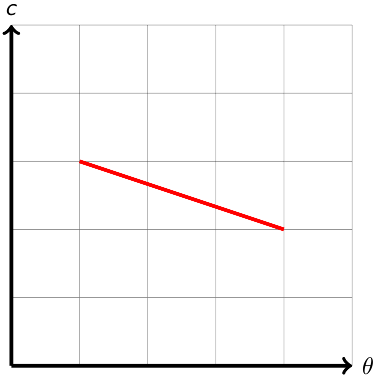


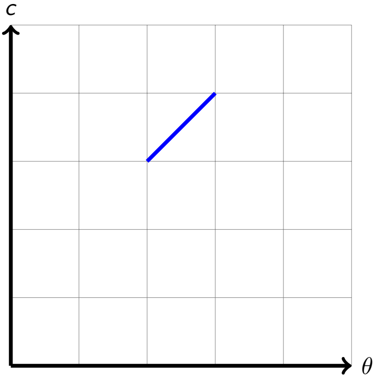
$\succcurlyeq$



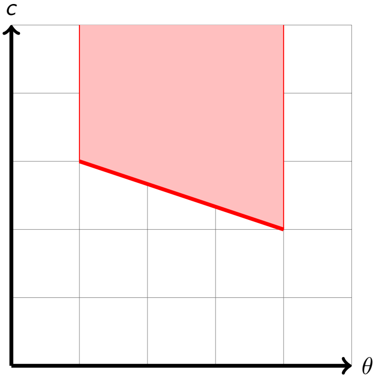
Can be checked with linear programming

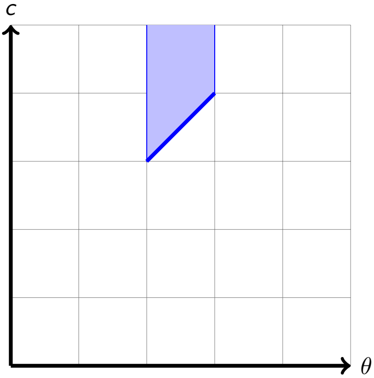
Relaxing Cost State Classes



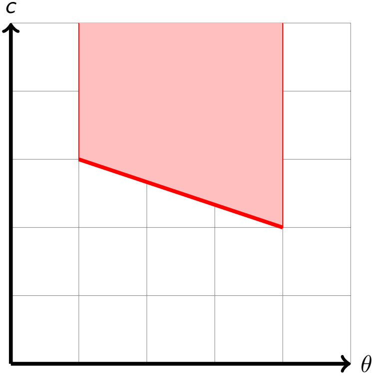


Relaxing Cost State Classes

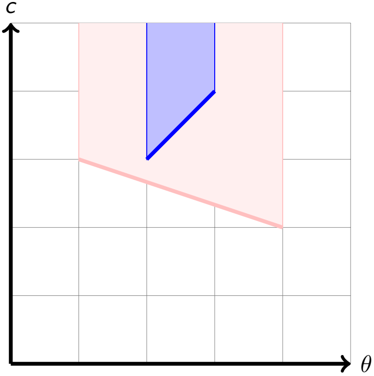




Relaxing Cost State Classes



$\supseteq$



Simple (Relaxed) Cost State Classes

- The computation of cost state classes relies on general convex polyhedra;
- Can we also do it only with DBMs (like for Timed Automata)?
- We need to closely examine the **variable elimination** in the successor computation;

Simple (Relaxed) Cost State Classes

- The computation of cost state classes relies on general convex polyhedra;
- Can we also do it only with DBMs (like for Timed Automata)?
- We need to closely examine the **variable elimination** in the successor computation;
- It can be done by the **Fourier-Motzkin** algorithm: to eliminate θ_1 just write that all lower bounds of θ_1 are less than all upper bounds of θ_1:

$$\begin{array}{lcl}
\theta_1 \in [0,4] & & 0 \leq 4 \\
\theta'_2 + \theta_1 \in [5,6] & & 0 \leq 6 - \theta'_2 \\
\theta'_4 + \theta_1 \in [1,2] & \xrightarrow{\text{eliminate } \theta_1} & \ldots \\
c' - 1 - 3 * \theta_1 \geq 0 & & 3 * 0 \leq c' - 1 \\
0 \leq \theta'_2 & & 3 * (5 - \theta'_2) \leq c' - 1 \\
0 \leq \theta'_4 & & 3 * (1 - \theta'_4) \leq c' - 1
\end{array}$$

Simple (Relaxed) Cost State Classes

- We can **split the successor** according to which term among 0, $5 - \theta'_2$, and $1 - \theta'_4$ is the greatest: this always gives DBMs with exactly one lower bound inequality on cost ($\rightarrow$ **simple** cost state classes):

$$\begin{array}{lcl}
\begin{array}{l} \dots \\ 0 \geq 5 - \theta'_2 \\ 0 \geq 1 - \theta'_4 \\ c' \geq 1 \end{array} & \text{or} & \begin{array}{l} \dots \\ 5 - \theta'_2 \geq 0 \\ 5 - \theta'_2 \geq 1 - \theta'_4 \\ c' \geq 16 - 3\theta'_2 \end{array} & \text{or} & \begin{array}{l} \dots \\ 1 - \theta'_4 \geq 0 \\ 1 - \theta'_4 \geq 5 - \theta'_2 \\ c' \geq 4 - 3\theta'_4 \end{array}
\end{array}$$

[4] Boucheneb and Mullins. Analyse des réseaux temporels : Calcul des classes en $O(n^2)$ et des temps de chemin en $O(mn)$. *TSI.*, 22(4):435–459, 2003.

[5] Bourdil et al. Symmetry reduction for time Petri net state classes. *Science of Computer Programming*, 132:209–225, 2016.

Simple (Relaxed) Cost State Classes

- We can **split the successor** according to which term among 0, $5 - \theta'_2$, and $1 - \theta'_4$ is the greatest: this always gives DBMs with exactly one lower bound inequality on cost ($\rightarrow$ **simple** cost state classes):

$$\begin{array}{lclcl} \ldots & & \ldots & & \ldots \\ 0 \geq 5 - \theta'_2 & \text{or} & 5 - \theta'_2 \geq 0 & \text{or} & 1 - \theta'_4 \geq 0 \\ 0 \geq 1 - \theta'_4 & & 5 - \theta'_2 \geq 1 - \theta'_4 & & 1 - \theta'_4 \geq 5 - \theta'_2 \\ c' \geq 1 & & c' \geq 16 - 3\theta'_2 & & c' \geq 4 - 3\theta'_4 \end{array}$$

- The inequations on non-cost variables can be directly computed as usual [4] [5]

[4] Boucheneb and Mullins. Analyse des réseaux temporels : Calcul des classes en $O(n^2)$ et des temps de chemin en $O(mn)$. *TSI.*, 22(4):435–459, 2003.

[5] Bourdil et al. Symmetry reduction for time Petri net state classes. *Science of Computer Programming*, 132:209–225, 2016.

Simple (Relaxed) Cost State Classes

- We can **split the successor** according to which term among 0, $5 - \theta'_2$, and $1 - \theta'_4$ is the greatest: this always gives DBMs with exactly one lower bound inequality on cost ($\rightarrow$ **simple** cost state classes):

$$\begin{array}{lclcl} \dots & & \dots & & \dots \\ 0 \geq 5 - \theta'_2 & \text{or} & 5 - \theta'_2 \geq 0 & \text{or} & 1 - \theta'_4 \geq 0 \\ 0 \geq 1 - \theta'_4 & & 5 - \theta'_2 \geq 1 - \theta'_4 & & 1 - \theta'_4 \geq 5 - \theta'_2 \\ c' \geq 1 & & c' \geq 16 - 3\theta'_2 & & c' \geq 4 - 3\theta'_4 \end{array}$$

- The inequations on non-cost variables can be directly computed as usual [4] [5]
- Eliminating disabled variables other than the fired transition can be done similarly.

[4] Boucheneb and Mullins. Analyse des réseaux temporels : Calcul des classes en $O(n^2)$ et des temps de chemin en $O(mn)$. *TSI.*, 22(4):435–459, 2003.

[5] Bourdil et al. Symmetry reduction for time Petri net state classes. *Science of Computer Programming*, 132:209–225, 2016.

Deciding Subsumption for Simple Cost State Classes

- To decide if $(m, D, c \geq \ell(\vec{\theta})) \preccurlyeq (m', D', c \geq \ell'(\vec{\theta}))$, we can check if [6]:
    1. $m = m'$;
    2. $D \subseteq D'$ (**DBM** inclusion);
    3. $\inf_{\vec{\theta} \in D} (\ell - \ell')(\vec{\theta}) \geq 0$
- Minimization over a DBM can be done efficiently as an instance of the **mincost flow graph problem**[6].

[6] Rasmussen et al. On using priced timed automata to achieve optimal scheduling. *FMSD*, 29(1):97–114, 2006.

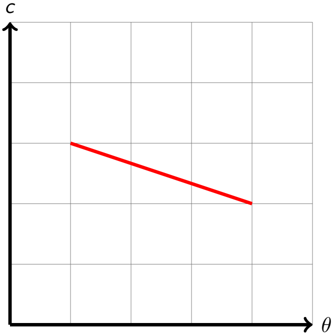
Plan

Termination for Simple Cost State Classes

We can use the symbolic algorithm with simple cost state classes: if (m', D') is a successor of (m, D) and D can be decomposed as $\{D'_1, \ldots, D'_n\}$ as before then each of the (m', D'_i) is a successor of (m, D)

Lemma

$\succcurlyeq$ *is a* **well quasi-order** *on simple cost state classes.*

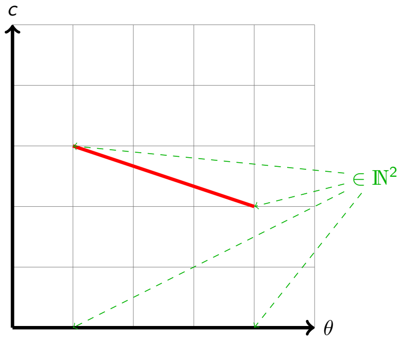


Termination for Simple Cost State Classes

We can use the symbolic algorithm with simple cost state classes: if (m', D') is a successor of (m, D) and D can be decomposed as $\{D'_1, \ldots, D'_n\}$ as before then each of the (m', D'_i) is a successor of (m, D)

Lemma

$\succcurlyeq$ *is a* **well quasi-order** *on simple cost state classes.*

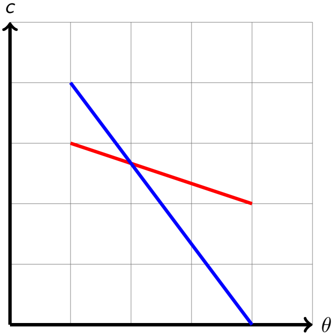


Termination for Simple Cost State Classes

We can use the symbolic algorithm with simple cost state classes: if (m', D') is a successor of (m, D) and D can be decomposed as $\{D'_1, \ldots, D'_n\}$ as before then each of the (m', D'_i) is a successor of (m, D)

Lemma

$\succcurlyeq$ *is a* **well quasi-order** *on simple cost state classes.*

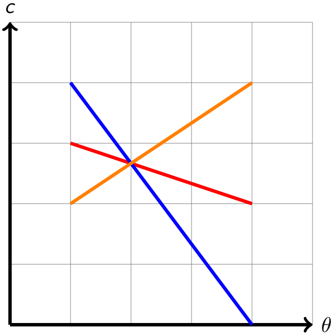


Termination for Simple Cost State Classes

We can use the symbolic algorithm with simple cost state classes: if (m', D') is a successor of (m, D) and D can be decomposed as $\{D'_1, \ldots, D'_n\}$ as before then each of the (m', D'_i) is a successor of (m, D)

Lemma

$\succcurlyeq$ *is a* **well quasi-order** *on simple cost state classes.*

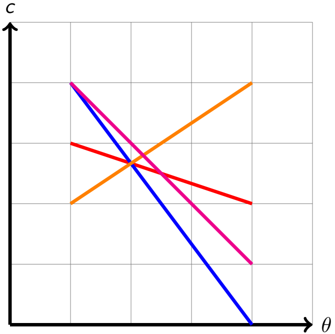


Termination for Simple Cost State Classes

We can use the symbolic algorithm with simple cost state classes: if (m', D') is a successor of (m, D) and D can be decomposed as $\{D'_1, \ldots, D'_n\}$ as before then each of the (m', D'_i) is a successor of (m, D)

Lemma

$\succcurlyeq$ *is a* **well quasi-order** *on simple cost state classes.*

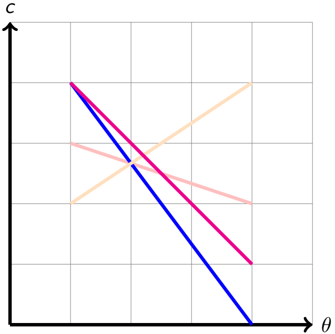


Termination for Simple Cost State Classes

We can use the symbolic algorithm with simple cost state classes: if (m', D') is a successor of (m, D) and D can be decomposed as $\{D'_1, \dots, D'_n\}$ as before then each of the (m', D'_i) is a successor of (m, D)

Lemma

$\succcurlyeq$ *is a* **well quasi-order** *on simple cost state classes.*



Termination for Normal Cost State Classes

Lemma

$\succcurlyeq$ *is a well quasi-order on cost state classes.*

1. $\succcurlyeq$ is actually a **better quasi-order** on simple cost state classes;

[7] Abdulla and Nylén. Better is better than well: On efficient verification of infinite-state systems. In *LICS*, 2000.

Termination for Normal Cost State Classes

Lemma

$\succcurlyeq$ *is a well quasi-order on cost state classes.*

1. $\succcurlyeq$ is actually a **better quasi-order** on simple cost state classes;
2. Therefore $\sqsupseteq$ such that $X \sqsupseteq Y$ iff $\forall y \in Y, \exists x \in X$ such that $x \succcurlyeq y$ is a better quasi-order[7];

[7] Abdulla and Nylén. Better is better than well: On efficient verification of infinite-state systems. In *LICS*, 2000.

Termination for Normal Cost State Classes

Lemma

$\succcurlyeq$ *is a well quasi-order on cost state classes.*

1. $\succcurlyeq$ is actually a **better quasi-order** on simple cost state classes;
2. Therefore $\sqsupseteq$ such that $X \sqsupseteq Y$ iff $\forall y \in Y, \exists x \in X$ such that $x \succcurlyeq y$ is a better quasi-order[7];
3. A relaxed cost state class is a union of simple cost state classes;

[7] Abdulla and Nylén. Better is better than well: On efficient verification of infinite-state systems. In *LICS*, 2000.

Termination for Normal Cost State Classes

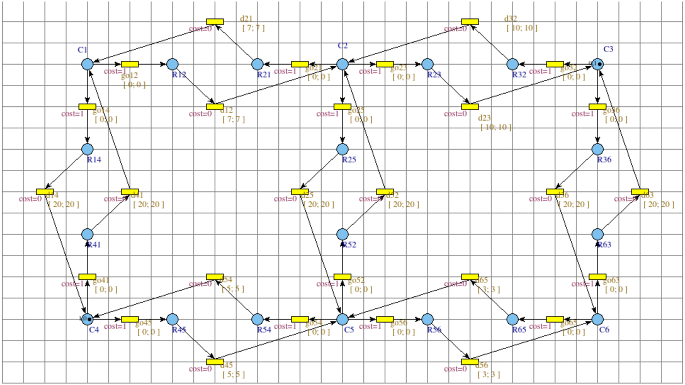
Lemma

$\succcurlyeq$ *is a well quasi-order on cost state classes.*

1. $\succcurlyeq$ is actually a **better quasi-order** on simple cost state classes;
2. Therefore $\sqsupseteq$ such that $X \sqsupseteq Y$ iff $\forall y \in Y, \exists x \in X$ such that $x \succcurlyeq y$ is a better quasi-order[7];
3. A relaxed cost state class is a union of simple cost state classes;
4. $\sqsupseteq$ implies $\supseteq$, implies $\succcurlyeq$ for relaxed classes.

[7] Abdulla and Nylén. Better is better than well: On efficient verification of infinite-state systems. In *LICS*, 2000.

Meeting optimally across the river



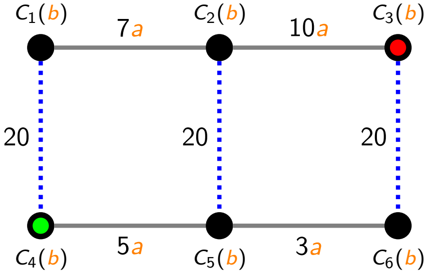
Waiting time

```
Checking property mincost (C1[0] + C1[1] == 2 or C2[0] + C2[1] == 2 or C3[0] + C3[1] == 2 or C4[0] + C4[1] == 2  or C5[0] + C5[1] == 2  or C6[0] + C6[1]
== 2) on TPN: /home/did/Desktop/romeo-3.8.5/meeting-simple2-noparam-cost.xml
Waiting for response (kill the romeo-cli process to interrupt)...
=7

Trace: go36__1, go45__0, d45__0, go56__0, d56__0, go65__0, d65__0, go56__0, d56__0, go65__0, d65__0, go56__0, d56__0, d36__1
```

Global time

```
Checking property mincost (C1[0] + C1[1] == 2 or C2[0] + C2[1] == 2 or C3[0] + C3[1] == 2 or C4[0] + C4[1] == 2  or C5[0] + C5[1] == 2  or C6[0] + C6[1]
== 2) on TPN: /home/did/Desktop/romeo-3.8.5/meeting-simple2-noparam-cost.xml
Waiting for response (kill the romeo-cli process to interrupt)...
=27

Trace: go36__1, go45__0, d45__0, go56__0, d56__0, go65__0, d65__0, go56__0, d56__0, go65__0, d65__0, go56__0, d56__0, d36__1
```

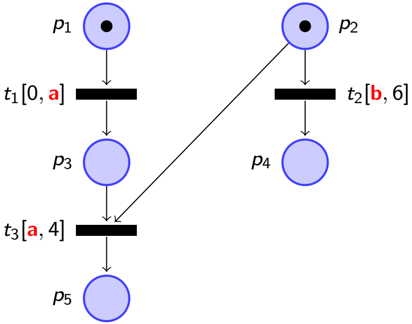
Can we do better? Meeting optimally and parametrically



- Let b be the maximum time they accept to wait in a city;
- They can accept to go more slowly (except on the river) by a common factor a;

Plan

Parametric Time Petri Nets



Parametric Bounded-Cost Problems

- **Existential problem**: given $c_{\max}$ and a marking m, does there exist a value of the parameters such that m is reachable with cost less or equal to $c_{\max}$.
- **Bounded-cost Synthesis problem**: find all such parameter values.
- **Infcost problem**: What is the infimum cost we can achieve over all parameter values?
- **Infcost synthesis problem**: find all parameter values for which the infimum cost (over all parameter values) can be achieved.

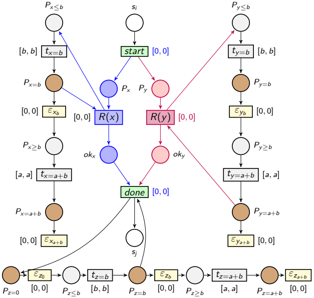
Undecidability of the existential problem

Theorem

The existential problem is undecidable for bounded parametric time Petri nets.

Encode the halting problem of two-counter machines in the existential problem for **time bounded** reachability.

- Start from an encoding for reachability in Parametric Timed Automata[8]
- Adapt to time Petri nets;
- Make all instructions execute in b time units instead of 1.

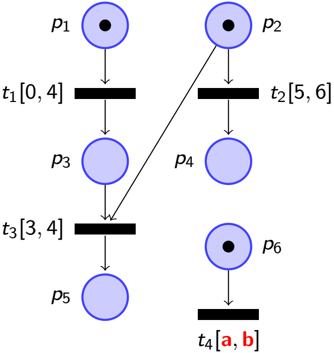


8 André et al. Decision problems for parametric timed automata. In ICFEM, 2016.

Parametric Cost State Classes

- We can compute state classes as before;
- The polyhedra obtained are **parametric DBMs** plus **cost inequalities**;
- Instantiating parameters with integer (or rational) values gives again a **cost state class**;
- Class subsumption is extended naturally: subsumed if subsumed for **all** parameter valuations.

Parametric Cost State Classes: Example

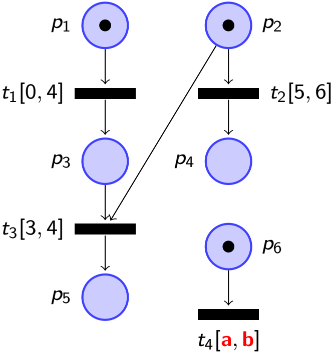


$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

Initially:

$\theta_1 \in [0,4]$
$\theta_2 \in [5,6]$
$\theta_4 \in [\mathbf{a},\mathbf{b}]$
$c \geq 0$

Parametric Cost State Classes: Example



$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

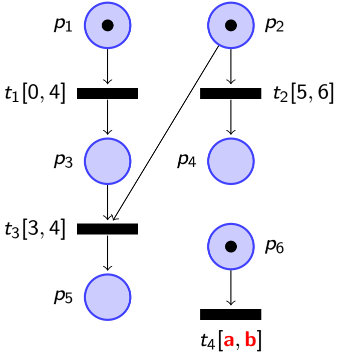
Initially:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [\mathbf{a},\mathbf{b}]$$
$$c \geq 0$$

Fire t_1:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [a,b]$$
$$c \geq 0$$
$$\theta_1 \leq \theta_2$$
$$\theta_1 \leq \theta_4$$

Parametric Cost State Classes: Example



$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

Initially:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [\mathbf{a},\mathbf{b}]$$
$$c \geq 0$$

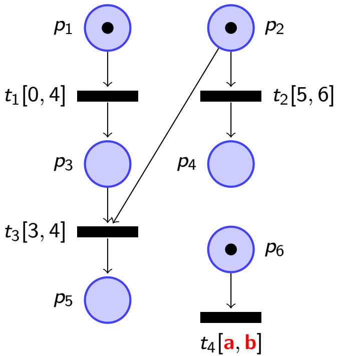
Fire t_1:

$$\theta_1 \in [0,4]$$
$$\theta_2 \in [5,6]$$
$$\theta_4 \in [a,b]$$
$$c \geq 0$$
$$\theta_1 \leq \theta_2$$
$$\theta_1 \leq \theta_4$$

Change origin:

$$\theta_1 \in [0,4]$$
$$\theta_2' + \theta_1 \in [5,6]$$
$$\theta_4' + \theta_1 \in [a,b]$$
$$c' - \omega(t_1) - cr(m_0) * \theta_1 \geq 0$$
$$\theta_1 \leq \theta_2' + \theta_1$$
$$\theta_1 \leq \theta_4' + \theta_1$$

Parametric Cost State Classes: Example



$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

Initially:

$\theta_1 \in [0,4]$
$\theta_2 \in [5,6]$
$\theta_4 \in [\mathbf{a},\mathbf{b}]$
$c \geq 0$

Fire t_1:

$\theta_1 \in [0,4]$
$\theta_2 \in [5,6]$
$\theta_4 \in [a,b]$
$c \geq 0$
$\theta_1 \leq \theta_2$
$\theta_1 \leq \theta_4$

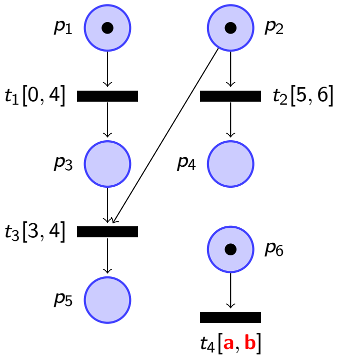
Change origin:

$\theta_1 \in [0,4]$
$\theta_2' + \theta_1 \in [5,6]$
$\theta_4' + \theta_1 \in [a,b]$
$c' - \omega(t_1) - cr(m_0) * \theta_1 \geq 0$
$\theta_1 \leq \theta_2' + \theta_1$
$\theta_1 \leq \theta_4' + \theta_1$

Eliminate disabled:

$\theta_2' \in [1,6]$
$\theta_4' \in [\mathbf{a-4},\mathbf{b}]$
$\theta_4' \geq 0$
$\theta_2' - \theta_4' \in [\mathbf{5-b},\mathbf{6-a}]$
$\mathbf{0 \leq a \leq b}$
$c \geq 16 - 3\theta_2'$
$c \geq \mathbf{3(a - \theta_4') + 1}$
$c \geq 1$

Parametric Cost State Classes: Example



$cr = p_1 + 2p_2$
$\omega(t_1) = 1$

Initially:

$\theta_1 \in [0,4]$
$\theta_2 \in [5,6]$
$\theta_4 \in [\mathbf{a},\mathbf{b}]$
$c \geq 0$

Fire t_1:

$\theta_1 \in [0,4]$
$\theta_2 \in [5,6]$
$\theta_4 \in [a,b]$
$c \geq 0$
$\theta_1 \leq \theta_2$
$\theta_1 \leq \theta_4$

Change origin:

$\theta_1 \in [0,4]$
$\theta'_2 + \theta_1 \in [5,6]$
$\theta'_4 + \theta_1 \in [a,b]$
$c' - \omega(t_1) - cr(m_0) * \theta_1 \geq 0$
$\theta_1 \leq \theta'_2 + \theta_1$
$\theta_1 \leq \theta'_4 + \theta_1$

Eliminate disabled:

$\theta'_2 \in [1,6]$
$\theta'_4 \in [\mathbf{a-4},\mathbf{b}]$
$\theta'_4 \geq 0$
$\theta'_2 - \theta'_4 \in [\mathbf{5-b},\mathbf{6-a}]$
$\mathbf{0 \leq a \leq b}$
$c \geq 16 - 3\theta'_2$
$c \geq \mathbf{3(a - \theta'_4) + 1}$
$c \geq 1$

Add newly enabled:

...
$\theta_3 \in [3,4]$
...

Symbolic Semi-algorithm for Bounded-Cost Reachability

PolyRes $\leftarrow \emptyset$
$\textsc{Passed} \leftarrow \emptyset$
$\textsc{Waiting} \leftarrow \{(m_0, D_0)\}$
while $\textsc{Waiting} \neq \emptyset$ **do**
    select $C_\sigma = (m, D)$ from $\textsc{Waiting}$
    if $m \in$ Goal **then**
        PolyRes $\leftarrow$ PolyRes $\cup \left(\mathbf{D} \cap (\mathbf{c} \leq \mathbf{c}_{\mathsf{max}})\right)_{|\mathbb{P}}$
    end if
    if for all $C' \in \textsc{Passed}, C_\sigma \not\preceq C'$ **then**
        add C_σ to $\textsc{Passed}$
        for all $t \in$ firable(C_σ), add $C_{\sigma.t}$ to $\textsc{Waiting}$
    end if
end while
return PolyRes

Symbolic Semi-algorithm for Infcost Reachability

$\textsc{Cost} \leftarrow \infty$
PolyRes $\leftarrow \emptyset$
$\textsc{Passed} \leftarrow \emptyset$
$\textsc{Waiting} \leftarrow \{(m_0, D_0)\}$
while $\textsc{Waiting} \neq \emptyset$ **do**
  select $C_\sigma = (m, D)$ from $\textsc{Waiting}$
  if $m \in$ Goal **then**
    if cost$(C_\sigma) < \textsc{Cost}$ **then**
      $\textsc{Cost} \leftarrow$ cost(C_σ)
      PolyRes $\leftarrow \big(\mathbf{D} \cap (\mathbf{c} = \textsc{Cost})\big)_{|\mathbb{P}}$
    else if cost$(C_\sigma) = \textsc{Cost}$ **then**
      PolyRes $\leftarrow$ PolyRes $\cup \big(\mathbf{D} \cap (\mathbf{c} = \textsc{Cost})\big)_{|\mathbb{P}}$
    end if
  end if
  if for all $C' \in \textsc{Passed}$, $C_\sigma \not\preccurlyeq C'$ **then**
    add C_σ to $\textsc{Passed}$
    for all $t \in$ firable(C_σ), add $C_{\sigma.t}$ to $\textsc{Waiting}$
  end if
end while
return $(\textsc{Cost},$ PolyRes$)$

Symbolic Parameter Synthesis Algorithms

- When they terminate, the previous algorithms are **sound** and **complete**:

Lemma

For all classes $C_\sigma = (m, D)$, $(\vec{\theta}, c, v) \in D$ if and only if there exists a run ρ in $v(\mathcal{N})$, and $I : \mathrm{en}(m) \to \mathcal{I}(\mathbb{Q}_{\geq 0})$, such that $\mathrm{sequence}(\rho) = \sigma$, $(m, I, c) = \mathrm{last}(\rho)$, and $\vec{\theta} \in I$.

Lemma

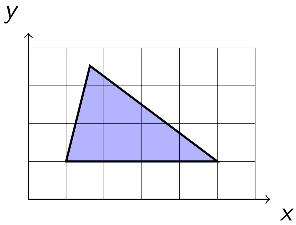
Let C_{σ_1} and C_{σ_2} be two state classes such that $C_{\sigma_1} \preccurlyeq C_{\sigma_2}$.
If a transition sequence σ is firable from C_{σ_1}, it is also firable from C_{σ_2} and $\mathrm{cost}(C_{\sigma_1.\sigma}) \geq \mathrm{cost}(C_{\sigma_2.\sigma})$.

- Termination is **not** guaranteed;

Integer hull

We use the **integer hull** trick[9] to:

1. make them compute **integer** parameter valuations;
2. ensure termination when parameters are **bounded**.

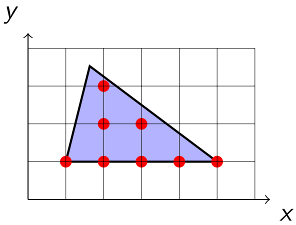


[9] Jovanović et al. Integer Parameter Synthesis for Real-Time Systems. Int *IEEE trans. on soft. eng.*, 41(5):445–461, 2015.

Integer hull

We use the **integer hull** trick[9] to:

1. make them compute **integer** parameter valuations;
2. ensure termination when parameters are **bounded**.

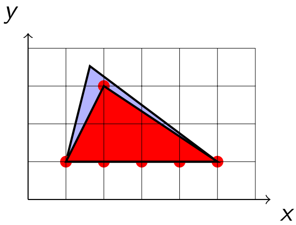


[9] Jovanović et al. Integer Parameter Synthesis for Real-Time Systems. Int *IEEE trans. on soft. eng.*, 41(5):445–461, 2015.

Integer hull

We use the **integer hull** trick[9] to:

1. make them compute **integer** parameter valuations;
2. ensure termination when parameters are **bounded**.



[9] Jovanović et al. Integer Parameter Synthesis for Real-Time Systems. Int *IEEE trans. on soft. eng.*, 41(5):445–461, 2015.

Integer Parameter Synthesis for Bounded-Cost Reachability

PolyRes $\leftarrow \emptyset$
$\textsc{Passed} \leftarrow \emptyset$
$\textsc{Waiting} \leftarrow \{(m_0, D_0)\}$
while $\textsc{Waiting} \neq \emptyset$ **do**
  select $C_\sigma = (m, D)$ from $\textsc{Waiting}$
  if $m \in$ Goal **then**
    PolyRes $\leftarrow$ PolyRes $\cup \left(\mathsf{IH}(D) \cap (c \leq c_{\max})\right)_{|\mathbb{P}}$
  end if
  if for all $C' \in \textsc{Passed}, \mathsf{IH}(C_\sigma) \not\preccurlyeq \mathsf{IH}(C')$ **then**
    add C_σ to $\textsc{Passed}$
    for all $t \in$ firable($\mathsf{IH}(C_\sigma)$), add $C_{\sigma.t}$ to $\textsc{Waiting}$
  end if
end while
return PolyRes

Integer Parameter Synthesis for Infcost Reachability

$\textsc{Cost} \leftarrow \infty$
PolyRes $\leftarrow \emptyset$
$\textsc{Passed} \leftarrow \emptyset$
$\textsc{Waiting} \leftarrow \{(m_0, D_0)\}$
while $\textsc{Waiting} \neq \emptyset$ **do**
    select $C_\sigma = (m, D)$ from $\textsc{Waiting}$
    if $m \in$ Goal **then**
        if $\mathrm{cost}(\mathrm{IH}(C_\sigma)) < \textsc{Cost}$ **then**
            $\textsc{Cost} \leftarrow \mathrm{cost}(\mathrm{IH}(C_\sigma))$
            PolyRes $\leftarrow \big(\mathrm{IH}(D) \cap (c = \textsc{Cost})\big)_{|\mathbb{P}}$
        else if $\mathrm{cost}(\mathrm{IH}(C_\sigma)) = \textsc{Cost}$ **then**
            PolyRes $\leftarrow$ PolyRes $\cup \big(\mathrm{IH}(D) \cap (c = \textsc{Cost})\big)_{|\mathbb{P}}$
        end if
    end if
    if for all $C' \in \textsc{Passed}, \mathrm{IH}(C_\sigma) \not\preccurlyeq \mathrm{IH}(C')$ **then**
        add C_σ to $\textsc{Passed}$
        for all $t \in \mathrm{firable}(\mathrm{IH}(C_\sigma))$, add $C_{\sigma.t}$ to $\textsc{Waiting}$
    end if
end while
return $(\textsc{Cost}, \mathrm{PolyRes})$

Integer Parameter Synthesis

- When they terminate, the previous algorithms are **sound** and **complete** for **integer** parameter valuations;

Lemma

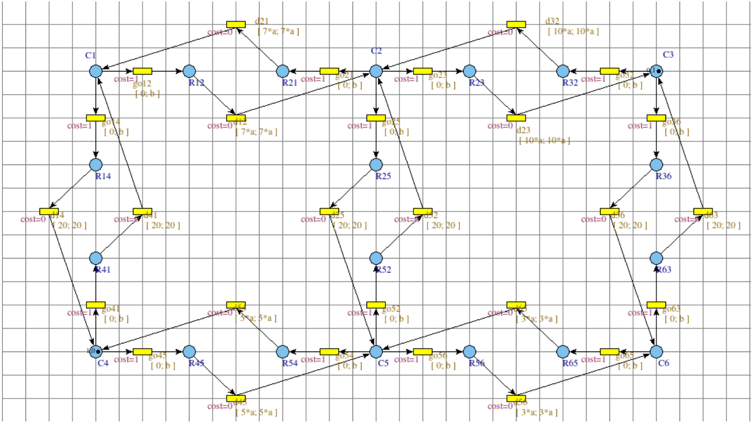
If v is an ***integer*** *parameter valuation, then for all classes $C_\sigma = (m, D)$, $(\vec{\theta}, c, v) \in \mathsf{IH}(D)$ if and only if there exists a run ρ in $v(\mathcal{N})$, and $I : \mathsf{en}(m) \to \mathcal{I}(\mathbb{Q}_{\geq 0})$, such that $\mathsf{sequence}(\rho) = \sigma$, $(m, I, c) = \mathsf{last}(\rho)$, and $\vec{\theta} \in I$.*

Lemma

Let C_{σ_1} and C_{σ_2} be two state classes such that $\mathsf{IH}(C_{\sigma_1}) \preccurlyeq \mathsf{IH}(C_{\sigma_2})$.
If a transition sequence σ is $\mathbb{N}^{\mathbb{P}}$-firable from C_{σ_1} it is also $\mathbb{N}^{\mathbb{P}}$-firable from C_{σ_2} and $\mathsf{cost}_{\mathbb{N}}(C_{\sigma_1.\sigma}) \geq \mathsf{cost}_{\mathbb{N}}(C_{\sigma_2.\sigma})$.

- Termination is **still not** guaranteed, except when parameters are **bounded**;
- When parameters are bounded, $\succcurlyeq$ is again a well-quasiorder;
- Integer hull can also be computed as part of the successor class computation.

Meeting parametrically across the river



Meeting parametrically across the river

Infcost

```
Checking property mincost (C1[0] + C1[1] == 2 or C2[0] + C2[1] == 2 or C3[0] + C3[1] == 2 or C4[0] + C4[1] == 2  or C5[0] + C5[1] == 2  or C6[0] + C6[1] == 2) on TPN: /home/did/Desktop/romeo-3.8.6/meeting-simple2.xml
Waiting for response (kill the romeo-cli process to interrupt)...
[warning] Real-valued parameters: no (theoretical) termination guarantee
in CTS file: /home/did/.romeo/temp/ctsfile.cts
=23
a in [1, 20/17]
b in [0, 10]
17*a + 3*b >= 20
or
a in [1, 5/2]
b in [0, 10]
8*a + 3*b >= 20

Traces:
-> go41__0, go32__1, d32__1, go21__1, d41__0, d21__1
-> go45__0, go36__1, d45__0, go56__0, d56__0, d36__1
-> go32__1, go41__0, d32__1, go21__1, d41__0, d21__1
-> go32__1, go41__0, d32__1, go21__1, d21__1, d41__0
-> go41__0, go32__1, d32__1, go21__1, d21__1, d41__0
-> go36__1, go45__0, d45__0, go56__0, d56__0, d36__1
-> go45__0, go36__1, d45__0, go56__0, d36__1, d56__0
-> go36__1, go45__0, d45__0, go56__0, d36__1, d56__0
```

Bounded cost $\leq$ 25

```
Checking property EF ((C1[0] + C1[1] == 2 or C2[0] + C2[1] == 2 or C3[0] + C3[1] == 2 or C4[0] + C4[1] == 2  or C5[0] + C5[1] == 2  or C6[0] + C6[1] == 2) and cost <= 25) on TPN: /home/did/Desktop/romeo-3.8.6/meeting-simple2.xml
Waiting for response (kill the romeo-cli process to interrupt)...
[warning] Real-valued parameters: no (theoretical) termination guarantee
in CTS file: /home/did/.romeo/temp/ctsfile.cts
a in [1, 10/7]
b in [0, 12/3[
14*a + 5*b >= 20
8*a + 3*b < 20
or
a in [1, 10/9]
b in [0, 3/3[
18*a + 5*b >= 20
17*a + 3*b < 20
or
a in [1, 140/113[
b in [0, 6/5[
14*a + 5*b < 20
-17*a + 2*b >= -20
17*a + 3*b >= 20
or
a in [1, 11/4]
b in [0, 10]
8*a + 3*b >= 20
-4*a + b >= -10
```

Integer hull and Performance

- Computing the integer hull is **expensive**:
- In all previous examples real parameters terminate, and faster;

Integer hull and Performance

- Computing the integer hull is **expensive**:
- In all previous examples real parameters terminate, and faster;
- The integer hull can **cut a lot of paths** off:
- By setting discrete costs to 0, with $a \leq 10, b = 0$ and a bounded cost of 40, IH terminates in 3s, Real in 55s.

Plan

Conclusion and Perspective

- Summary:
  - Using state classes we can solve the optimal cost reachability problem for bounded TPNs;
  - With state classes we need **no extrapolation** to ensure termination;
  - We can directly compute costs using **polyhedra** or using **DBMs**, through state class splitting;
  - The polyhedra approach can be extended for **parameter synthesis** in bounded-cost and inf-cost reachability;
  - The **integer hull** trick allows for terminating **symbolic** algorithms for bounded integer parameters;
  - The techniques are **implemented** in the freely available Roméo tool.

Conclusion and Perspective

- Summary:
  - Using state classes we can solve the optimal cost reachability problem for bounded TPNs;
  - With state classes we need **no extrapolation** to ensure termination;
  - We can directly compute costs using **polyhedra** or using **DBMs**, through state class splitting;
  - The polyhedra approach can be extended for **parameter synthesis** in bounded-cost and inf-cost reachability;
  - The **integer hull** trick allows for terminating **symbolic** algorithms for bounded integer parameters;
  - The techniques are **implemented** in the freely available Roméo tool.
- Future work:
  - Optimal cost as a function of parameters;
  - Parameter synthesis in **parametric cost** timed models;
  - Integer hull for undecidable non-parametric cost problems (control, upper bound “hard” constraints).