On the optimal reachability problem in weighted timed automata and games

Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France

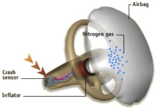
Based on former works with Thomas Brihaye, Kim G. Larsen, Nicolas Markey, etc...
And on recent work with Samy Jaziri and Nicolas Markey

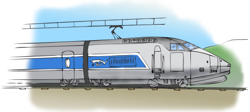
Time-dependent systems

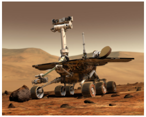
- We are interested in timed systems

Time-dependent systems

- We are interested in timed systems

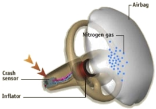


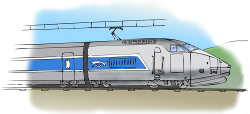


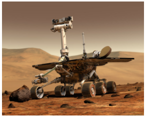


Time-dependent systems

- We are interested in timed systems







- ... and in their analysis and control

An example: The task graph scheduling problem

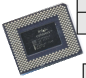
Compute $D \times (C \times (A + B)) + (A + B) + (C \times D)$ using two processors:

P_1 (fast):

time	
+	2 picoseconds
×	3 picoseconds

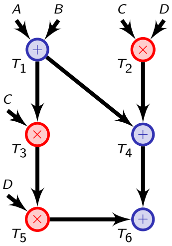
energy	
idle	10 Watt
in use	90 Watts

P_2 (slow):



time	
+	5 picoseconds
×	7 picoseconds

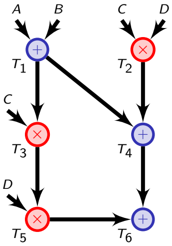
energy	
idle	20 Watts
in use	30 Watts



[BFLM10] Bouyer, Fahrenberg, Larsen, Markey. Quantitative Analysis of Real-Time Systems using Priced Timed Automata.

An example: The task graph scheduling problem

Compute $D\times(C\times(A+B))+(A+B)+(C\times D)$ using two processors:

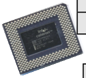


P_1 (fast):

	time
$+$	2 picoseconds
$\times$	3 picoseconds

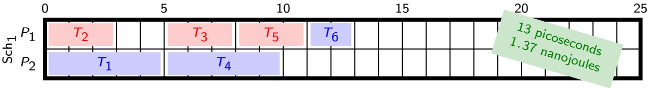
energy	
idle	10 Watt
in use	90 Watts

P_2 (slow):



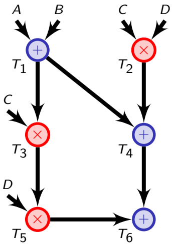
	time
$+$	5 picoseconds
$\times$	7 picoseconds

energy	
idle	20 Watts
in use	30 Watts



An example: The task graph scheduling problem

Compute $D\times(C\times(A+B))+(A+B)+(C\times D)$ using two processors:

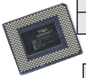


P_1 (fast):

time	
$+$	2 picoseconds
$\times$	3 picoseconds

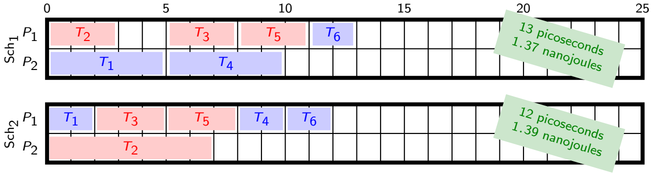
energy	
idle	10 Watt
in use	90 Watts

P_2 (slow):



time	
$+$	5 picoseconds
$\times$	7 picoseconds

energy	
idle	20 Watts
in use	30 Watts



[BFLM10] Bouyer, Fahrenberg, Larsen, Markey. Quantitative Analysis of Real-Time Systems using Priced Timed Automata.

An example: The task graph scheduling problem

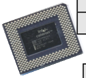
Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

P_1 (fast):

	time
$+$	2 picoseconds
$\times$	3 picoseconds

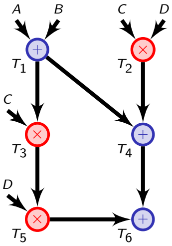
energy	
idle	10 Watt
in use	90 Watts

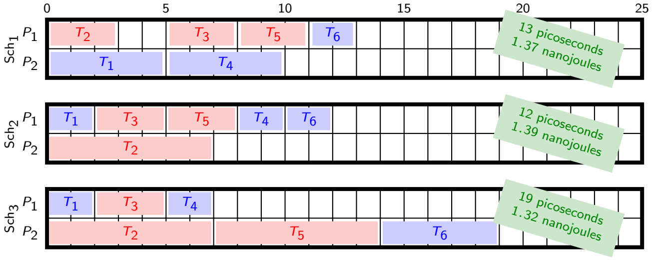
P_2 (slow):



	time
$+$	5 picoseconds
$\times$	7 picoseconds

energy	
idle	20 Watts
in use	30 Watts



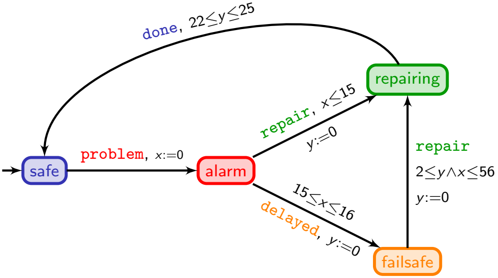


[BFLM10] Bouyer, Fahrenberg, Larsen, Markey. Quantitative Analysis of Real-Time Systems using Priced Timed Automata.

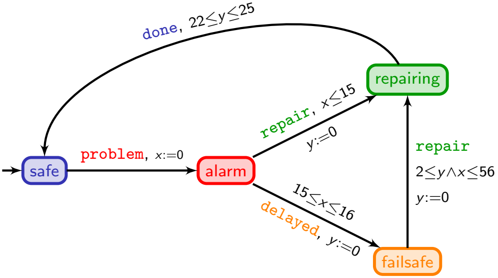
Outline

1. Timed automata
2. Weighted timed automata
3. Timed games
4. Weighted timed games
5. Conclusion

The model of timed automata



The model of timed automata



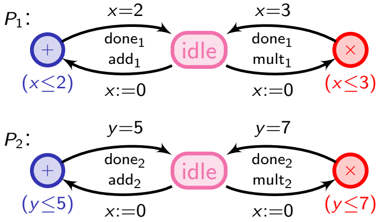
	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	⋯
y	0		23		23		38.6		0	

	failsafe	$\xrightarrow{2.3}$	failsafe	$\xrightarrow{\texttt{repair}}$	repairing	$\xrightarrow{22.1}$	repairing	$\xrightarrow{\texttt{done}}$	safe
⋯	15.6		17.9		17.9		40		40
	0		2.3		0		22.1		22.1

Modelling the task graph scheduling problem

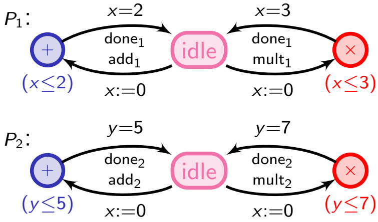
Modelling the task graph scheduling problem

- Processors

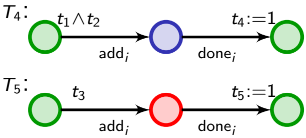


Modelling the task graph scheduling problem

- Processors

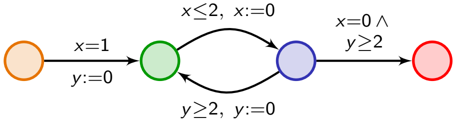


- Tasks

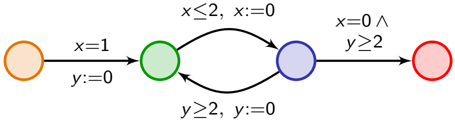


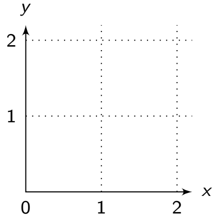
A schedule is a path in the product automaton

Analyzing timed automata

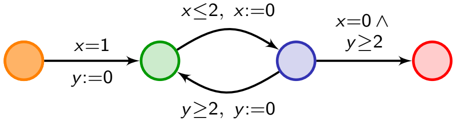


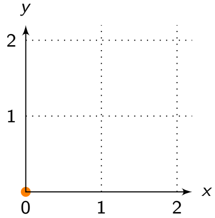
Analyzing timed automata



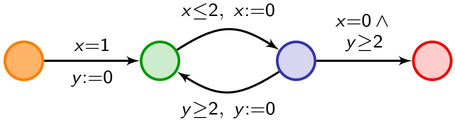


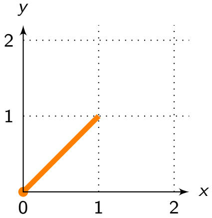
Analyzing timed automata



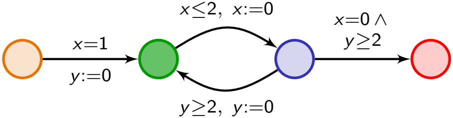


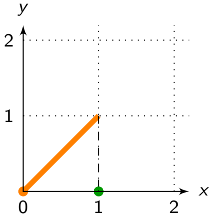
Analyzing timed automata



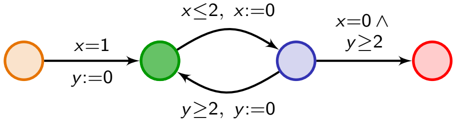


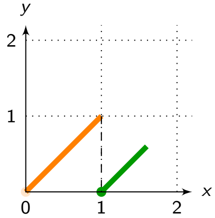
Analyzing timed automata



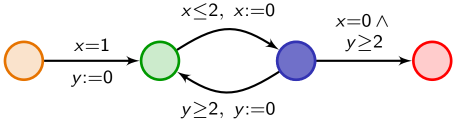


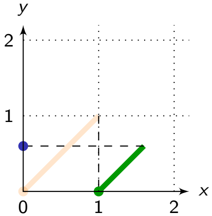
Analyzing timed automata



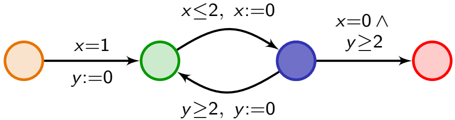


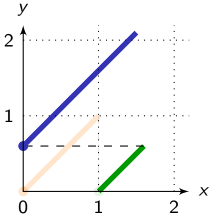
Analyzing timed automata



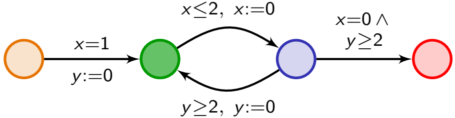


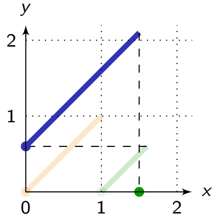
Analyzing timed automata



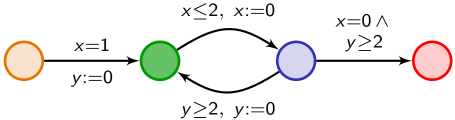


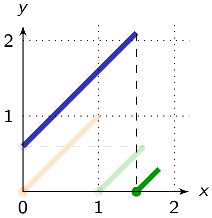
Analyzing timed automata



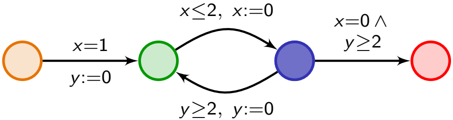


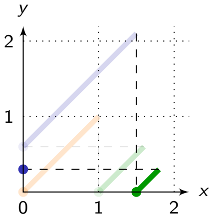
Analyzing timed automata



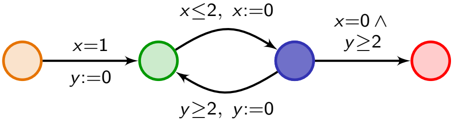


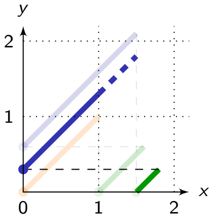
Analyzing timed automata



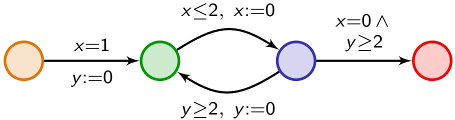


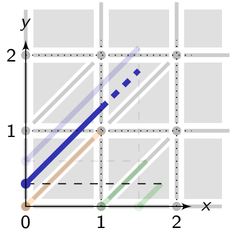
Analyzing timed automata



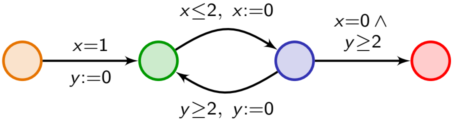


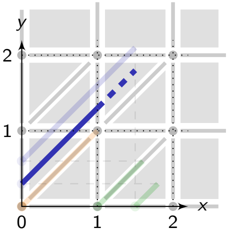
Analyzing timed automata





Analyzing timed automata



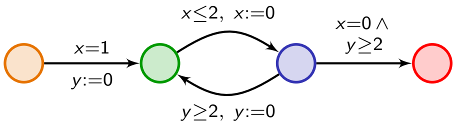


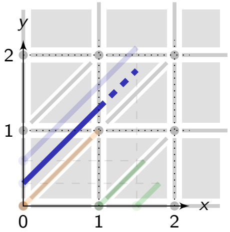
Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction

Analyzing timed automata



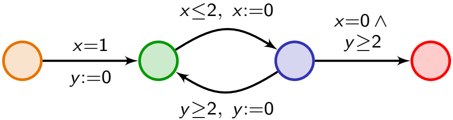


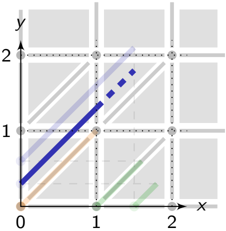
Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction
- Efficient symbolic technics based on zones, implemented in tools

Analyzing timed automata





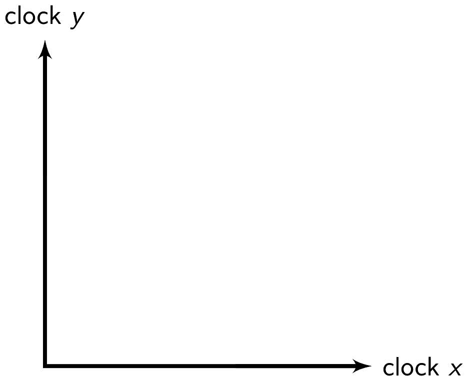
Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

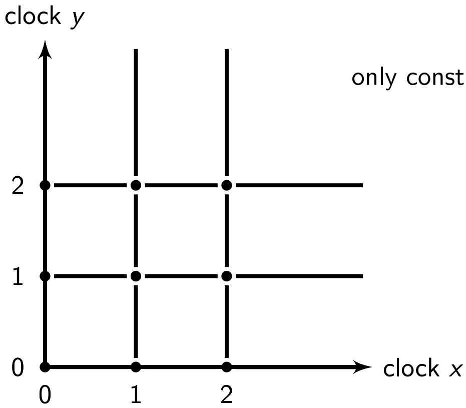
- Technical tool: region abstraction
- Efficient symbolic technics based on zones, implemented in tools

Skip regions

Technical tool: Region abstraction



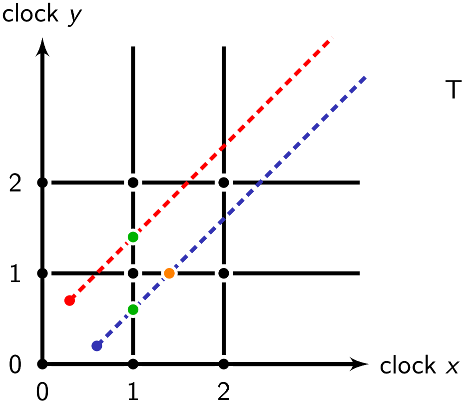
Technical tool: Region abstraction



only constraints: $x \sim c$ with $c \in \{0, 1, 2\}$
$y \sim c$ with $c \in \{0, 1, 2\}$

- "compatibility" between regions and constraints

Technical tool: Region abstraction

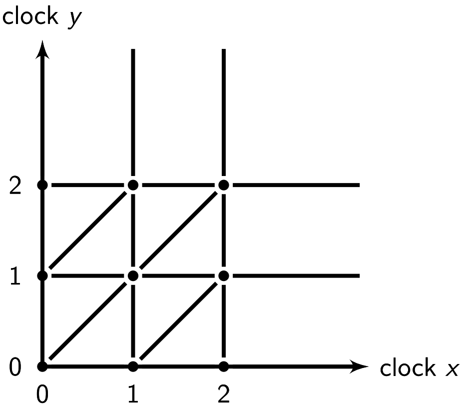


The path 

- can be fired from ●
- cannot be fired from ●

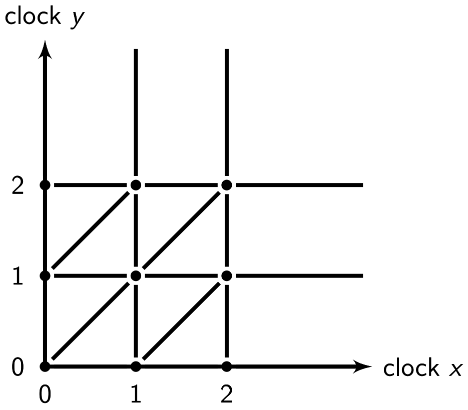
- “compatibility” between regions and constraints
- “compatibility” between regions and time elapsing

Technical tool: Region abstraction



- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

Technical tool: Region abstraction



- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

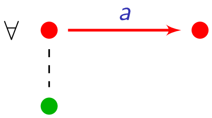
$\rightsquigarrow$ This is a finite time-abstract bisimulation!

Time-abstract bisimulation

This is a relation between • and • such that:

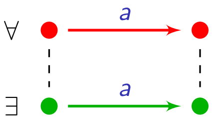
Time-abstract bisimulation

This is a relation between • and • such that:



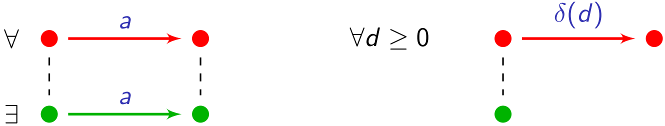
Time-abstract bisimulation

This is a relation between • and • such that:



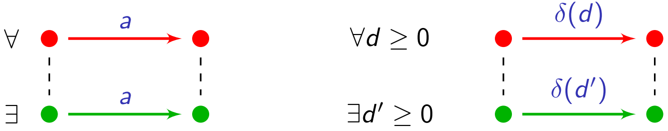
Time-abstract bisimulation

This is a relation between • and • such that:



Time-abstract bisimulation

This is a relation between • and • such that:



Time-abstract bisimulation

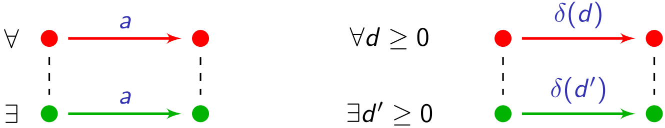
This is a relation between $\bullet$ and $\bullet$ such that:

$$\begin{tikzcd} \forall & \bullet \arrow[r, "a"] \arrow[d, dash, dashed] & \bullet \arrow[d, dash, dashed] \\ \exists & \bullet \arrow[r, "a"] & \bullet \end{tikzcd}
\qquad
\begin{tikzcd} \forall d \geq 0 & \bullet \arrow[r, "\delta(d)"] \arrow[d, dash, dashed] & \bullet \arrow[d, dash, dashed] \\ \exists d' \geq 0 & \bullet \arrow[r, "\delta(d')"] & \bullet \end{tikzcd}$$

... and vice-versa (swap $\bullet$ and $\bullet$).

Time-abstract bisimulation

This is a relation between • and • such that:



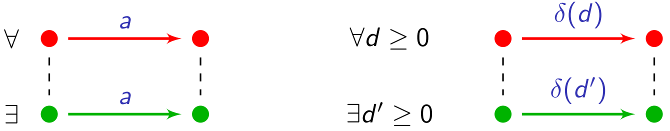
... and vice-versa (swap • and •).

Consequence

$$\forall \quad (\ell_1, v_1) \xrightarrow{d_1, a_1} (\ell_2, v_2) \xrightarrow{d_2, a_2} (\ell_3, v_3) \xrightarrow{d_3, a_3} \quad \cdots$$

Time-abstract bisimulation

This is a relation between • and • such that:



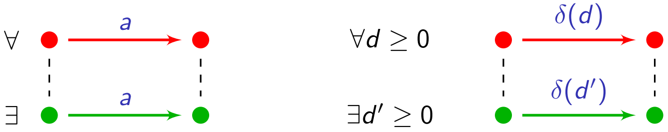
... and vice-versa (swap • and •).

Consequence

$$
\begin{array}{ccccccccc}
\forall & (\ell_1, v_1) & \xrightarrow{d_1, a_1} & (\ell_2, v_2) & \xrightarrow{d_2, a_2} & (\ell_3, v_3) & \xrightarrow{d_3, a_3} & \cdots & \\
& \downarrow & & \downarrow & & \downarrow & & & \\
& (\ell_1, R_1) & \xrightarrow{a_1} & (\ell_2, R_2) & \xrightarrow{a_2} & (\ell_3, R_3) & \xrightarrow{a_3} & \cdots & \text{with } v_i \in R_i
\end{array}
$$

Time-abstract bisimulation

This is a relation between • and • such that:



... and vice-versa (swap • and •).

Consequence

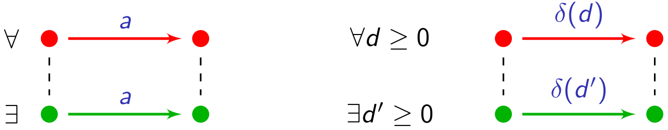
$$\begin{tikzcd}[column sep=small] \forall & (\ell_1, v_1) \arrow[r, "d_1, a_1"] \arrow[d] & (\ell_2, v_2) \arrow[r, "d_2, a_2"] \arrow[d] & (\ell_3, v_3) \arrow[r, "d_3, a_3"] \arrow[d] & \cdots \\ & (\ell_1, R_1) \arrow[r, "a_1"] & (\ell_2, R_2) \arrow[r, "a_2"] & (\ell_3, R_3) \arrow[r, "a_3"] & \cdots \end{tikzcd}$$

with $v_i \in R_i$

$\forall v'_1 \in R_1$

Time-abstract bisimulation

This is a relation between • and • such that:

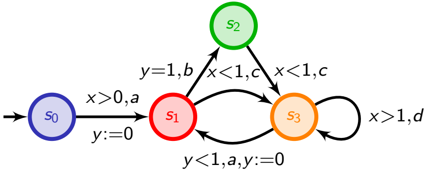


... and vice-versa (swap • and •).

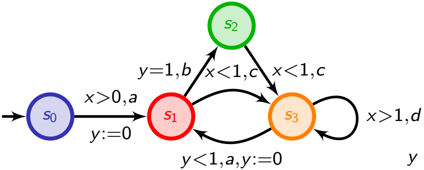
Consequence

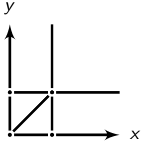
$$
\begin{tikzcd}
\forall & (\ell_1, v_1) \arrow[r, "d_1, a_1"] \arrow[d] & (\ell_2, v_2) \arrow[r, "d_2, a_2"] \arrow[d] & (\ell_3, v_3) \arrow[r, "d_3, a_3"] \arrow[d] & \cdots & \\
 & (\ell_1, R_1) \arrow[r, "a_1"] \arrow[d] & (\ell_2, R_2) \arrow[r, "a_2"] \arrow[d] & (\ell_3, R_3) \arrow[r, "a_3"] \arrow[d] & \cdots & \text{with } v_i \in R_i \\
\forall v_1' \in R_1 \ \exists & (\ell_1, v_1') \arrow[r, "d_1', a_1"] & (\ell_2, v_2') \arrow[r, "d_2', a_2"] & (\ell_3, v_3') \arrow[r, "d_3', a_3"] & \cdots & \text{with } v_i' \in R_i
\end{tikzcd}
$$

Technical tool: Region abstraction – An example [AD94]

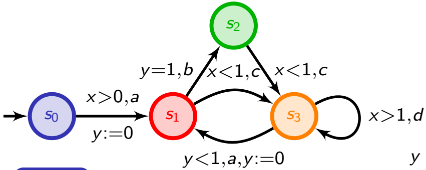


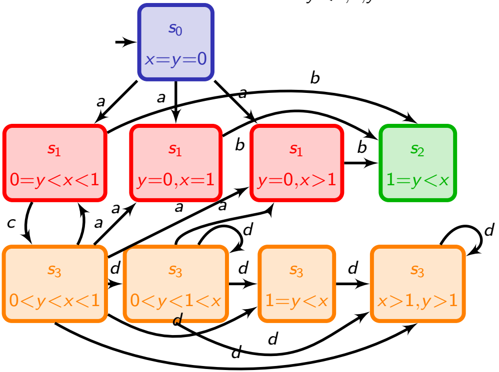
Technical tool: Region abstraction – An example [AD94]

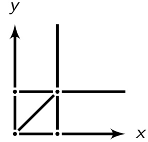




Technical tool: Region abstraction – An example [AD94]







Outline

Modelling resources in timed systems

- System resources might be relevant and even crucial information

Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...
  - price to pay,
  - bandwidth,

Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...
  - price to pay,
  - bandwidth,

$\rightsquigarrow$ timed automata are not powerful enough!

Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...
  - price to pay,
  - bandwidth,

$\leadsto$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

a discrete control (the mode of the system)
$+$ continuous evolution of the variables within a mode

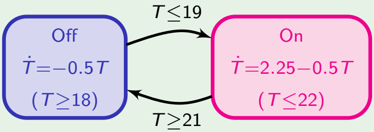
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...
  - price to pay,
  - bandwidth,

$\rightsquigarrow$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

The thermostat example



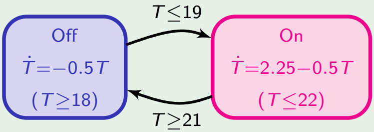
Modelling resources in timed systems

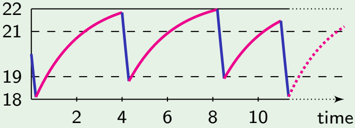
- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...
  - price to pay,
  - bandwidth,

$\rightsquigarrow$ timed automata are not powerful enough!

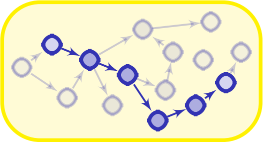
- A possible solution: use hybrid automata

The thermostat example

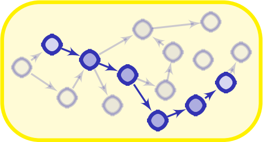




Ok...

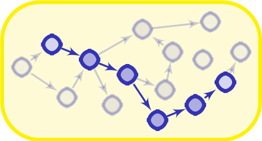


Ok...

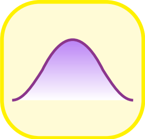


Easy...

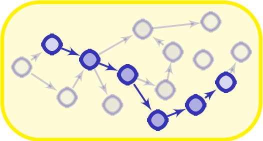
Ok...



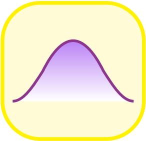
Easy...



Ok...

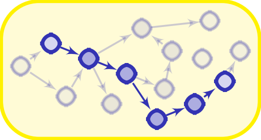


Easy...

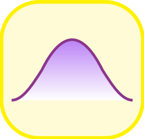


Easy...

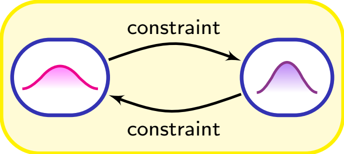
Ok... but?



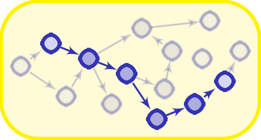
Easy...



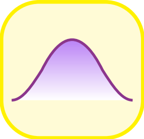
Easy...



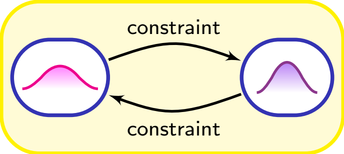
Ok... but?



Easy...



Easy...



Hard!

Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...
  - price to pay,
  - bandwidth,

  $\rightsquigarrow$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

Theorem [HKPV95]

The reachability problem is undecidable in hybrid automata. Even for the simplest, the so-called stopwatch automata (clocks can be stopped).

[HKPV95] Henzinger, Kopke, Puri, Varaiya. What's decidable wbout hybrid automata? *(SToC'95)*.

Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...
  - price to pay,
  - bandwidth,

  $\rightsquigarrow$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

Theorem [HKPV95]

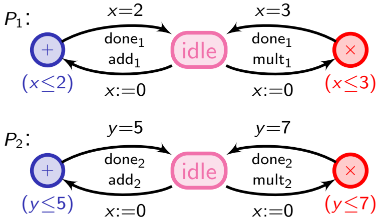
The reachability problem is undecidable in hybrid automata. Even for the simplest, the so-called stopwatch automata (clocks can be stopped).

- An alternative: weighted/priced timed automata [ALP01,BFH+01]
  $\rightsquigarrow$ hybrid variables do not constrain the system
  hybrid variables are observer variables

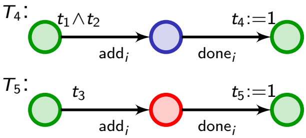
[HKPV95] Henzinger, Kopke, Puri, Varaiya. What's decidable wbout hybrid automata? *(SToC'95)*.
[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Modelling the task graph scheduling problem

- Processors

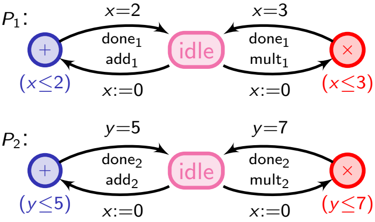


- Tasks

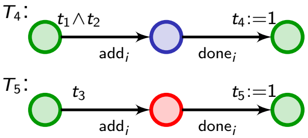


Modelling the task graph scheduling problem

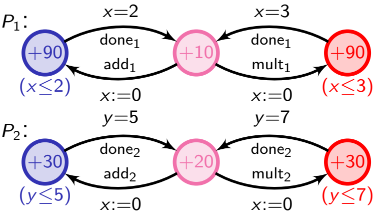
- Processors



- Tasks

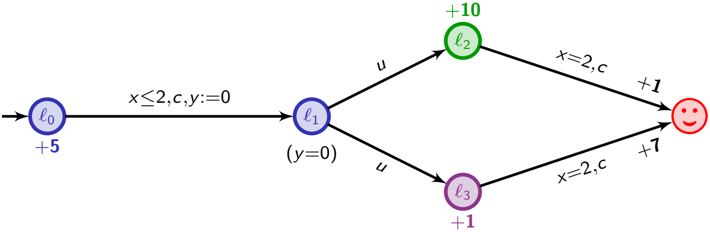


- Modelling energy



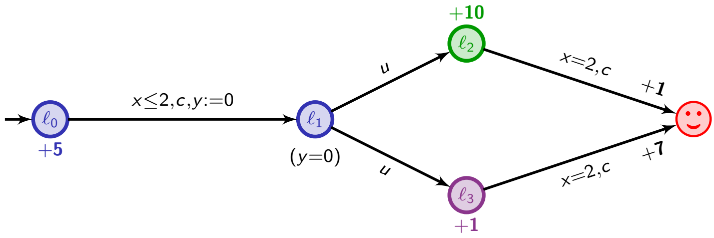
A good schedule is a path in the product automaton with a low cost

Weighted/priced timed automata [ALP01,BFH+01]



[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

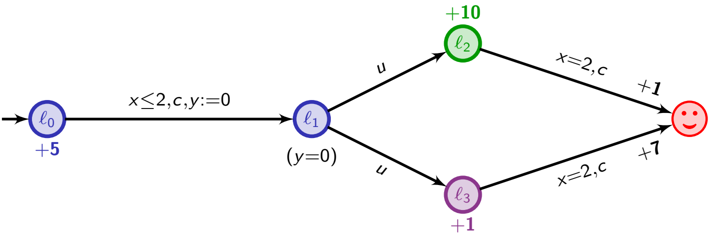
Weighted/priced timed automata [ALP01,BFH+01]



$$\begin{array}{ccccccccccccc}
 & \ell_0 & \xrightarrow{1.3} & \ell_0 & \xrightarrow{c} & \ell_1 & \xrightarrow{u} & \ell_3 & \xrightarrow{0.7} & \ell_3 & \xrightarrow{c} & ☺ \\
x & 0 & & 1.3 & & 1.3 & & 1.3 & & 2 & & \\
y & 0 & & 1.3 & & 0 & & 0 & & 0.7 & &
\end{array}$$

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

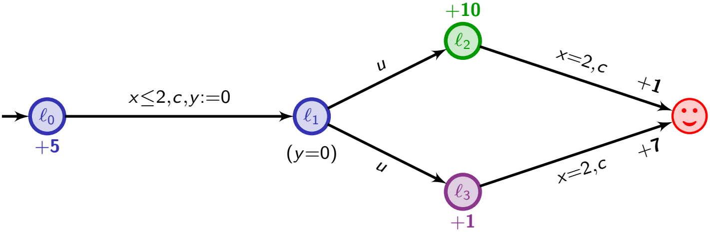


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

cost :

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

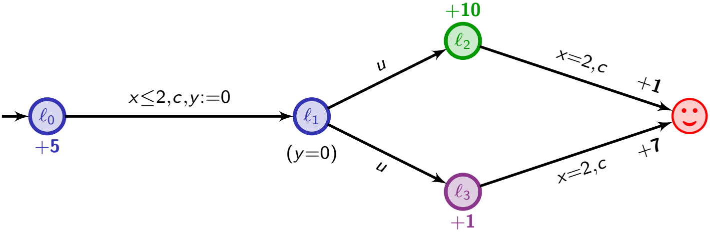


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

cost : 6.5

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

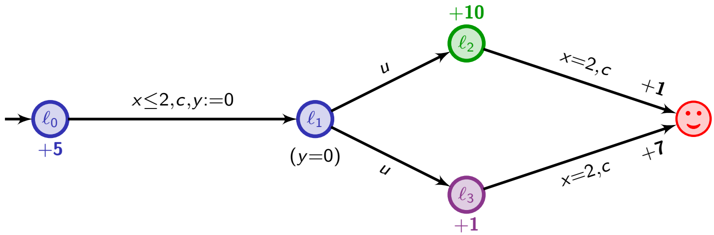


$$
\begin{array}{lccccccccccc}
 & \ell_0 & \xrightarrow{1.3} & \ell_0 & \xrightarrow{c} & \ell_1 & \xrightarrow{u} & \ell_3 & \xrightarrow{0.7} & \ell_3 & \xrightarrow{c} & ☺ \\
x & 0 & & 1.3 & & 1.3 & & 1.3 & & 2 & & \\
y & 0 & & 1.3 & & 0 & & 0 & & 0.7 & &
\end{array}
$$

cost : 6.5 + 0

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

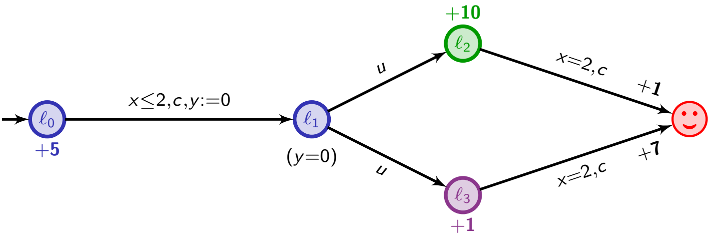


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

cost : 6.5 + 0 + 0

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

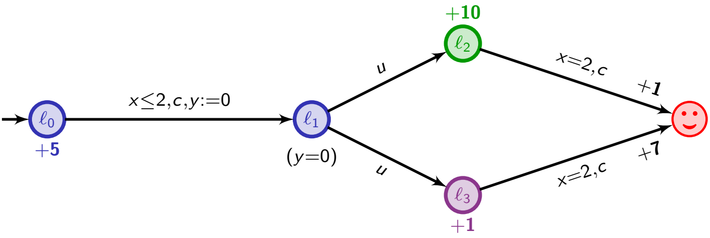


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

cost : 6.5 + 0 + 0 + 0.7

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

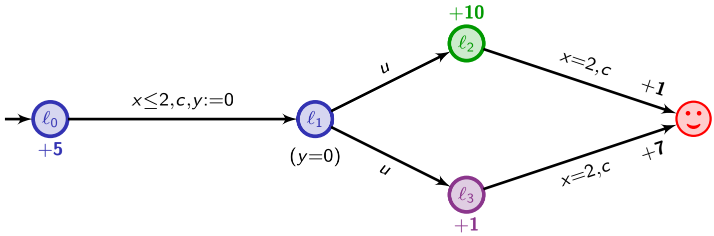


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

cost : 6.5 + 0 + 0 + 0.7 + 7

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

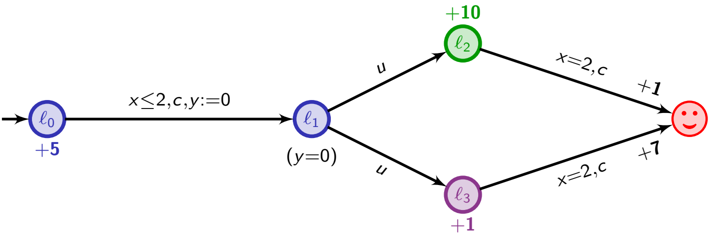
Weighted/priced timed automata [ALP01,BFH+01]



	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺	
x	0		1.3		1.3		1.3		2			
y	0		1.3		0		0		0.7			
cost :		6.5	+	0	+	0	+	0.7	+	7	=	14.2

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

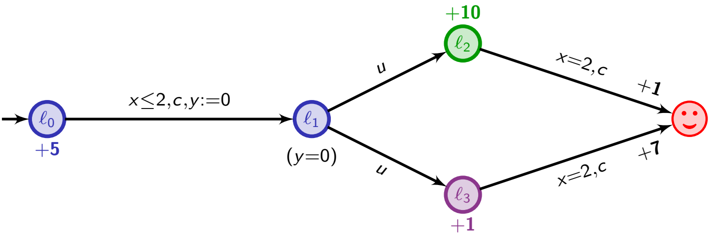
Weighted/priced timed automata [ALP01,BFH+01]



Question: what is the optimal cost for reaching 🙂?

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

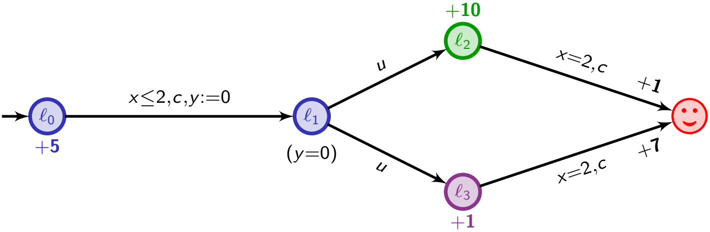


Question: what is the optimal cost for reaching 🙂 ?

$$5t + 10(2 - t) + 1$$

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

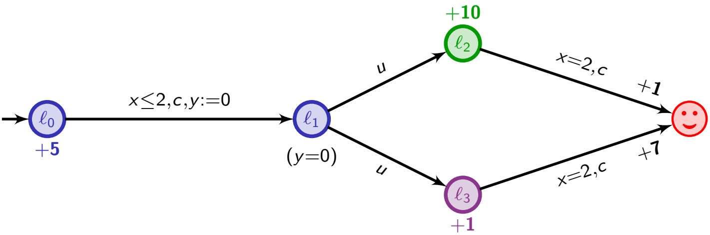


Question: what is the optimal cost for reaching ☺ ?

$$5t + 10(2-t) + 1\ ,\ 5t + (2-t) + 7$$

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

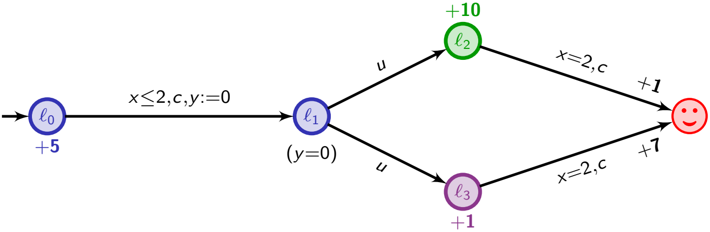


Question: what is the optimal cost for reaching ☺?

$$\min\left(5t + 10(2-t) + 1 \;,\; 5t + (2-t) + 7 \right)$$

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]

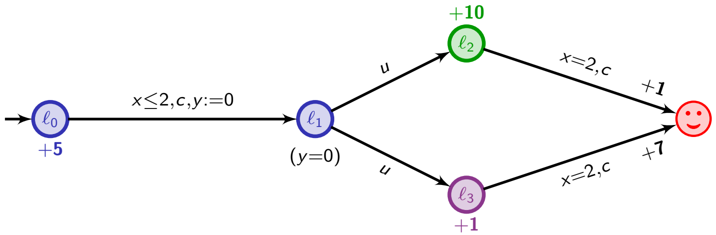


Question: what is the optimal cost for reaching 🙂?

$$\inf_{0 \leq t \leq 2} \min \left(5t + 10(2-t) + 1 \;,\; 5t + (2-t) + 7 \right) = 9$$

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata [ALP01,BFH+01]



Question: what is the optimal cost for reaching ☺ ?

$$\inf_{0\leq t\leq 2} \min\left(\, 5t+10(2-t)+1 \,,\, 5t+(2-t)+7 \,\right) = 9$$

$\rightsquigarrow$ *strategy:* leave immediately ℓ_0, go to ℓ_3, and wait there 2 t.u.

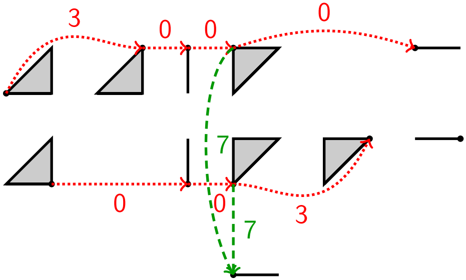
[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Optimal-cost reachability

Theorem [ALP01,BFH+01,BBBR07]

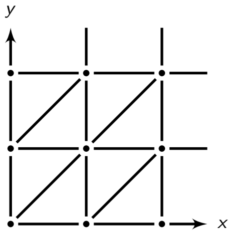
In weighted timed automata, the optimal cost is an integer and can be computed in PSPACE.

- Technical tool: a refinement of the regions, the corner-point abstraction

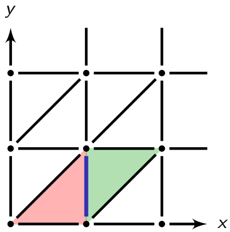


[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.
[BBBR07] Bouyer, Brihaye, Bruyère, Raskin. On the optimal reachability problem *(Formal Methods in System Design)*.

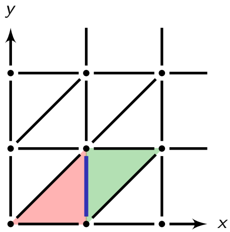
Technical tool: the corner-point abstraction



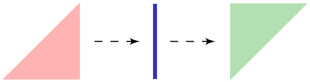
Technical tool: the corner-point abstraction



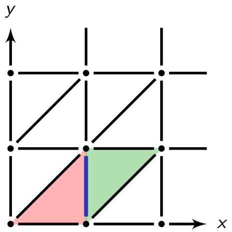
Technical tool: the corner-point abstraction



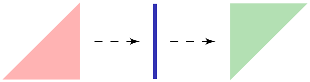
Abstract time successors:



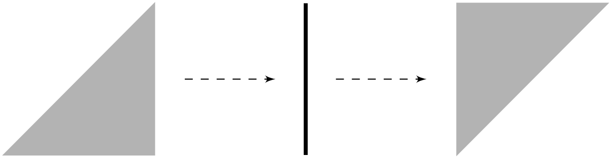
Technical tool: the corner-point abstraction



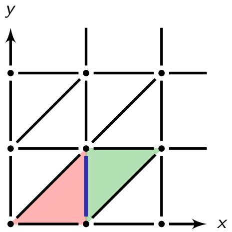
Abstract time successors:



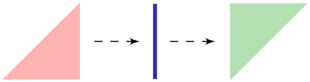
Concrete time successors:



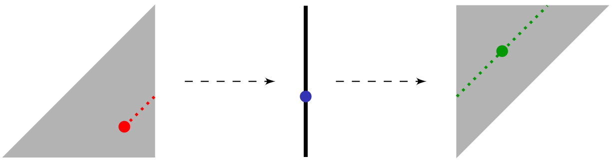
Technical tool: the corner-point abstraction



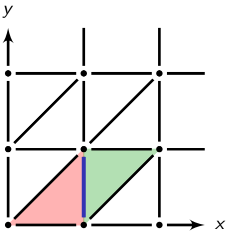
Abstract time successors:



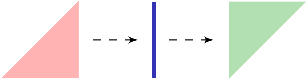
Concrete time successors:



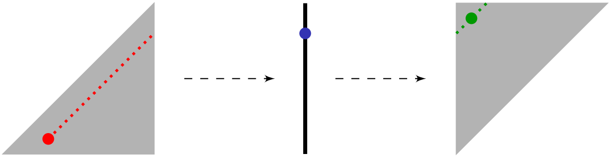
Technical tool: the corner-point abstraction



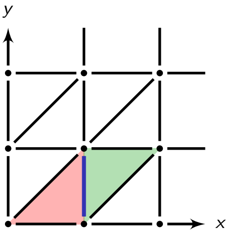
Abstract time successors:



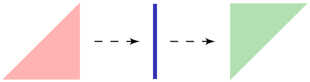
Concrete time successors:



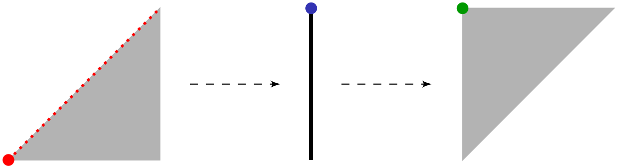
Technical tool: the corner-point abstraction



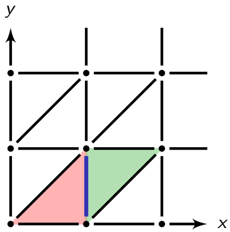
Abstract time successors:



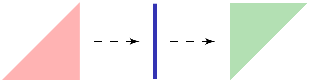
Concrete time successors:



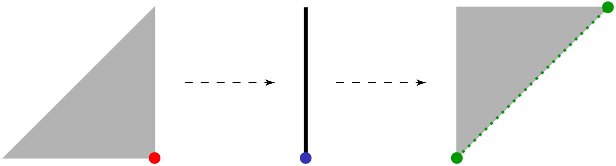
Technical tool: the corner-point abstraction



Abstract time successors:



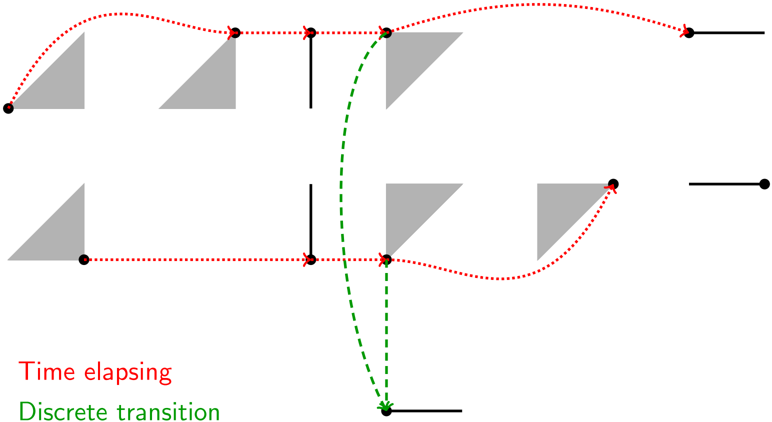
Concrete time successors:



Technical tool: the corner-point abstraction

Idea

only record information about the extremal points



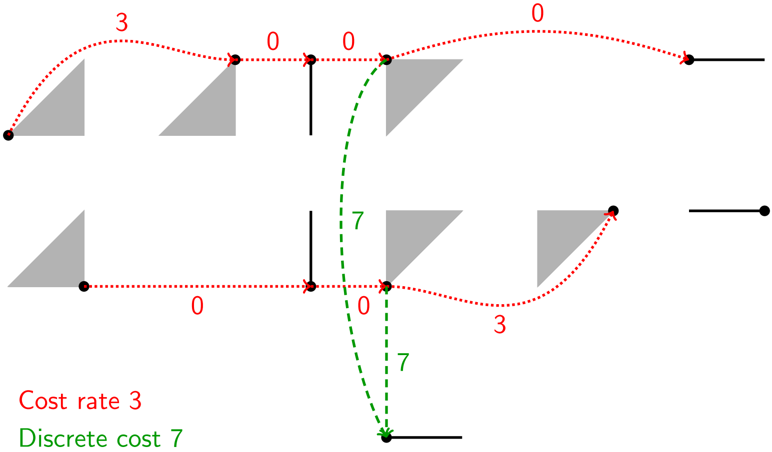
Time elapsing

Discrete transition

Technical tool: the corner-point abstraction

Idea

only record information about the extremal points



From timed to discrete behaviours

Optimal reachability as a linear programming problem

From timed to discrete behaviours

Optimal reachability as a linear programming problem

$$\circ \xrightarrow{t_1} \circ \xrightarrow{t_2} \circ \xrightarrow{t_3} \circ \xrightarrow{t_4} \circ \xrightarrow{t_5} \circ \quad \cdots$$

From timed to discrete behaviours

Optimal reachability as a linear programming problem

$$\circ \xrightarrow{t_1} \circ \xrightarrow[x\leq 2]{t_2} \circ \xrightarrow{t_3} \circ \xrightarrow{t_4} \circ \xrightarrow{t_5} \circ \quad \cdots \quad \left\{ \begin{array}{l} t_1+t_2\leq 2 \\ \\ \end{array} \right.$$

From timed to discrete behaviours

Optimal reachability as a linear programming problem

$$\circ \xrightarrow[y:=0]{t_1} \circ \xrightarrow[x\leq 2]{t_2} \circ \xrightarrow{t_3} \circ \xrightarrow[y\geq 5]{t_4} \circ \xrightarrow{t_5} \circ \quad \cdots \quad \begin{cases} t_1+t_2\leq 2 \\ t_2+t_3+t_4\geq 5 \end{cases}$$

From timed to discrete behaviours

Optimal reachability as a linear programming problem

$$\begin{array}{cccccc} T_1 & T_2 & T_3 & T_4 & T_5 & \\ \circ \xrightarrow[y:=0]{t_1} & \circ \xrightarrow[x\le 2]{t_2} & \circ \xrightarrow{t_3} & \circ \xrightarrow[y\ge 5]{t_4} & \circ \xrightarrow{t_5} & \circ \end{array} \quad \cdots \quad \begin{cases} t_1+t_2\le 2 & T_2\le 2 \\ t_2+t_3+t_4\ge 5 & T_4-T_1\ge 5 \end{cases}$$

From timed to discrete behaviours

Optimal reachability as a linear programming problem

$$\begin{array}{cccccc} & T_1 & T_2 & T_3 & T_4 & T_5 \end{array}$$

$$\circ \xrightarrow[y:=0]{t_1} \circ \xrightarrow[x\leq 2]{t_2} \circ \xrightarrow{t_3} \circ \xrightarrow[y\geq 5]{t_4} \circ \xrightarrow{t_5} \circ \quad \cdots \quad \begin{cases} t_1+t_2\leq 2 & T_2\leq 2 \\ t_2+t_3+t_4\geq 5 & T_4-T_1\geq 5 \end{cases}$$

Lemma

Let Z be a bounded zone and f be a function

$$f : (T_1, ..., T_n) \mapsto \sum_{i=1}^{n} c_i T_i + c$$

well-defined on $\overline{Z}$. Then $inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

From timed to discrete behaviours

Optimal reachability as a linear programming problem

$$\begin{array}{cccccc} & T_1 & T_2 & T_3 & T_4 & T_5 \end{array}$$

$$\circ \xrightarrow[y:=0]{t_1} \circ \xrightarrow[x\leq 2]{t_2} \circ \xrightarrow{t_3} \circ \xrightarrow[y\geq 5]{t_4} \circ \xrightarrow{t_5} \circ \quad \cdots \quad \begin{cases} t_1+t_2\leq 2 & T_2\leq 2 \\ t_2+t_3+t_4\geq 5 & T_4-T_1\geq 5 \end{cases}$$

Lemma

Let Z be a bounded zone and f be a function

$$f : (T_1, \ldots, T_n) \mapsto \sum_{i=1}^{n} c_i T_i + c$$

well-defined on $\overline{Z}$. Then $inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

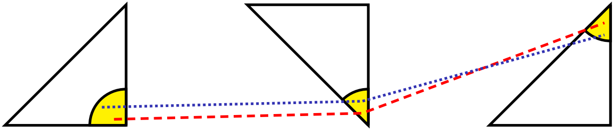
$\rightsquigarrow$ for every finite path π in $\mathcal{A}$, there exists a path Π in $\mathcal{A}_{\text{cp}}$ such that

$$\text{cost}(\Pi) \leq \text{cost}(\pi)$$

[Π is a "corner-point projection" of π]

From discrete to timed behaviours

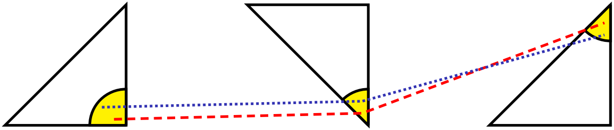
Approximation of abstract paths:



For any path Π of $\mathcal{A}_{\mathsf{cp}}$,

From discrete to timed behaviours

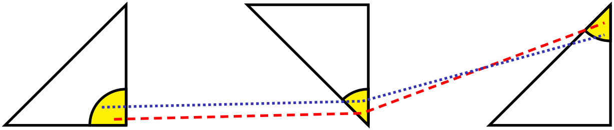
Approximation of abstract paths:



For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$,

From discrete to timed behaviours

Approximation of abstract paths:

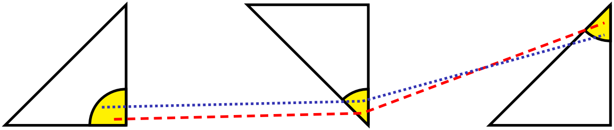


For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$, there exists a path π_ε of $\mathcal{A}$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$

From discrete to timed behaviours

Approximation of abstract paths:



For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$, there exists a path π_ε of $\mathcal{A}$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$

For every $\eta > 0$, there exists $\varepsilon > 0$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon \Rightarrow |\mathrm{cost}(\Pi) - \mathrm{cost}(\pi_\varepsilon)| < \eta$$

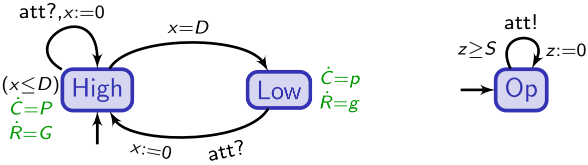
Use of the corner-point abstraction

It is a very interesting abstraction, that can be used in several other contexts:

- for mean-cost optimization [BBL04,BBL08]
- for discounted-cost optimization [FL08]
- for all concavely-priced timed automata [JT08]
- for deciding frequency objectives [BBBS11,Sta12]
- ...

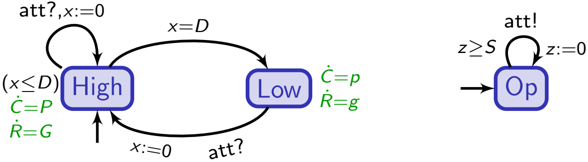
[BBL04] Bouyer, Brinksma, Larsen. Staying Alive As Cheaply As Possible *(HSCC'04)*.
[BBL08] Bouyer, Brinksma, Larsen. Optimal infinite scheduling for multi-priced timed automata *(Formal Methods in System Designs)*.
[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata *(INFINITY'08)*.
[JT08] Judziński, Trivedi. Concavely-priced timed automata *(FORMATS'08)*.
[BBBS11] Bertrand, Bouyer, Brihaye, Stainer. Emptiness and universality problems in timed automata with positive frequency *(ICALP'11)*.
[Sta12] Stainer. Frequencies in forgetful timed automata *(FORMATS'12)*.

Going further 1: mean-cost optimization



[BBL08] Bouyer, Brinksma, Larsen. Optimal infinite scheduling for multi-priced timed automata *(Formal Methods in System Designs)*.

Going further 1: mean-cost optimization

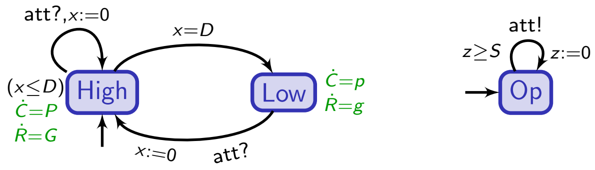


$\rightsquigarrow$ compute optimal infinite schedules that minimize

$$\text{mean-cost}(\pi) = \limsup_{n \to +\infty} \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)}$$

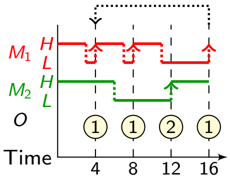
[BBL08] Bouyer, Brinksma, Larsen. Optimal infinite scheduling for multi-priced timed automata *(Formal Methods in System Designs)*.

Going further 1: mean-cost optimization

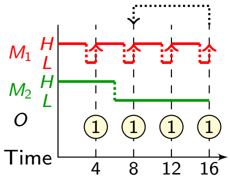


$\rightsquigarrow$ compute optimal infinite schedules that minimize

$$\text{mean-cost}(\pi) = \limsup_{n\to+\infty} \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)}$$



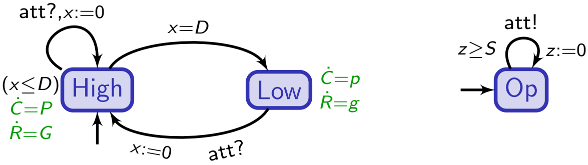
Schedule with ratio ≈1.455



Schedule with ratio ≈1.478

[BBL08] Bouyer, Brinksma, Larsen. Optimal infinite scheduling for multi-priced timed automata *(Formal Methods in System Designs)*.

Going further 1: mean-cost optimization



$\rightsquigarrow$ compute optimal infinite schedules that minimize

$$\text{mean-cost}(\pi) = \limsup_{n \to +\infty} \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)}$$

Theorem [BBL08]

In weighted timed automata, the optimal mean-cost can be compute in PSPACE.

$\rightsquigarrow$ the corner-point abstraction can be used

[BBL08] Bouyer, Brinksma, Larsen. Optimal infinite scheduling for multi-priced timed automata *(Formal Methods in System Designs)*.

From timed to discrete behaviours

- **Finite behaviours:** based on the following property

Lemma

Let Z be a bounded zone and f be a function

$$f : (t_1, \ldots, t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

From timed to discrete behaviours

- **Finite behaviours:** based on the following property

Lemma

Let Z be a bounded zone and f be a function

$$f : (t_1, \ldots, t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

$\rightsquigarrow$ for every finite path π in $\mathcal{A}$, there exists a path Π in $\mathcal{A}_{\mathsf{cp}}$ s.t.
mean-cost$(\Pi) \leq$ mean-cost(π)

From timed to discrete behaviours

- **Finite behaviours:** based on the following property

Lemma

Let Z be a bounded zone and f be a function

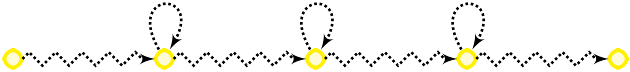
$$f : (t_1, \ldots, t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

$\rightsquigarrow$ for every finite path π in $\mathcal{A}$, there exists a path Π in $\mathcal{A}_{\text{cp}}$ s.t.

$$\text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi)$$

- **Infinite behaviours:** decompose each sufficiently long projection into cycles:



The (acyclic) linear part will be negligible!

From timed to discrete behaviours

- **Finite behaviours:** based on the following property

Lemma

Let Z be a bounded zone and f be a function

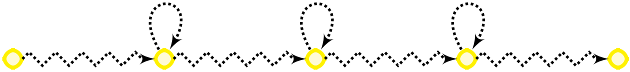
$$f : (t_1, \dots, t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

$\rightsquigarrow$ for every finite path π in $\mathcal{A}$, there exists a path Π in $\mathcal{A}_{\text{cp}}$ s.t.

$$\text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi)$$

- **Infinite behaviours:** decompose each sufficiently long projection into cycles:

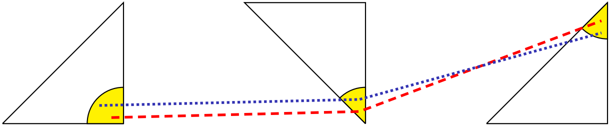


The (acyclic) linear part will be negligible!

$\rightsquigarrow$ the optimal cycle of $\mathcal{A}_{\text{cp}}$ is better than any infinite path of $\mathcal{A}$!

From discrete to timed behaviours

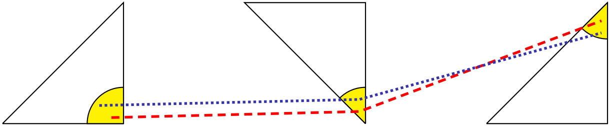
Approximation of abstract paths:



For any path Π of $\mathcal{A}_{\mathsf{cp}}$,

From discrete to timed behaviours

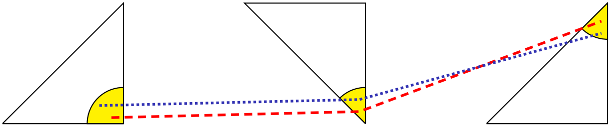
Approximation of abstract paths:



For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$,

From discrete to timed behaviours

Approximation of abstract paths:

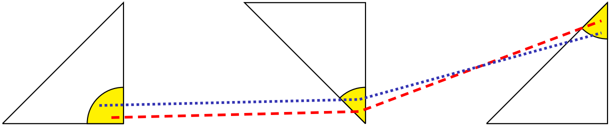


For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$, there exists a path π_ε of $\mathcal{A}$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$

From discrete to timed behaviours

Approximation of abstract paths:



For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$, there exists a path π_ε of $\mathcal{A}$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$

For every $\eta > 0$, there exists $\varepsilon > 0$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon \Rightarrow |\text{mean-cost}(\Pi) - \text{mean-cost}(\pi_\varepsilon)| < \eta$$

Going further 2: concavely-priced cost functions

$\rightsquigarrow$ A general abstract framework for quantitative timed systems

Theorem [JT08]

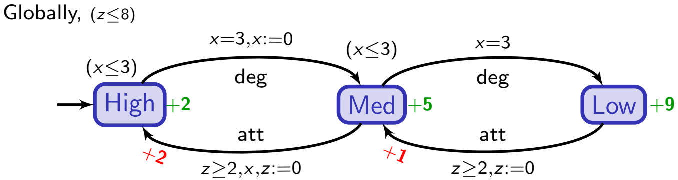
In concavely-priced timed automata, optimal cost is computable, if we restrict to quasi-concave cost functions. For the following cost functions, the (decision) problem is even PSPACE-complete:

- optimal-time and optimal-cost reachability;
- optimal discrete discounted cost;
- optimal mean-cost.

$\rightsquigarrow$ the corner-point abstraction can be used

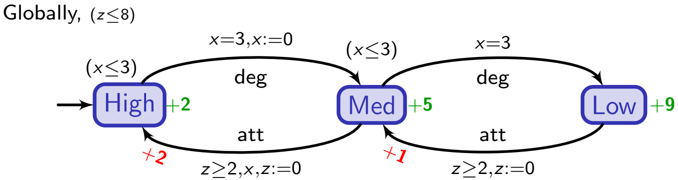
[JT08] Judziński, Trivedi. Concavely-priced timed automata *(FORMATS'08)*.

Going further 3: discounted-time cost optimization



[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata *(INFINITY'08)*.

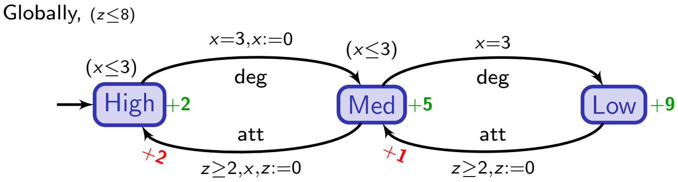
Going further 3: discounted-time cost optimization



$\rightsquigarrow$ compute optimal infinite schedules that minimize
discounted cost over time

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata *(INFINITY'08)*.

Going further 3: discounted-time cost optimization



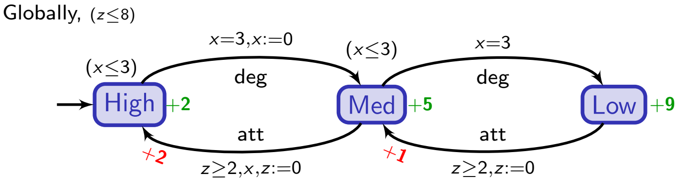
$\rightsquigarrow$ compute optimal infinite schedules that minimize

$$\text{discounted-cost}_{\lambda}(\pi) = \sum_{n\geq 0} \lambda^{T_n} \int_{t=0}^{\tau_{n+1}} \lambda^t \text{cost}(\ell_n)\, \mathrm{d}t + \lambda^{T_{n+1}} \text{cost}(\ell_n \xrightarrow{a_{n+1}} \ell_{n+1})$$

if $\pi = (\ell_0, v_0) \xrightarrow{\tau_1, a_1} (\ell_1, v_1) \xrightarrow{\tau_2, a_2} \cdots$ and $T_n = \sum_{i\leq n} \tau_i$

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata *(INFINITY'08)*.

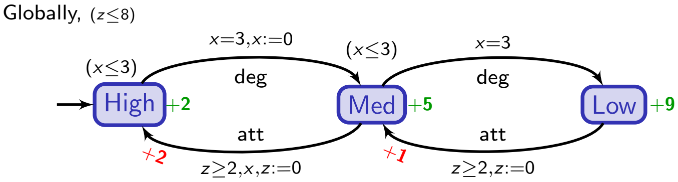
Going further 3: discounted-time cost optimization



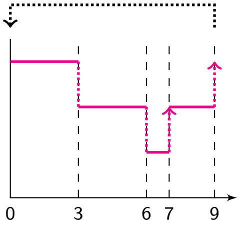
$\rightsquigarrow$ compute optimal infinite schedules that minimize discounted cost over time

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata *(INFINITY'08)*.

Going further 3: discounted-time cost optimization



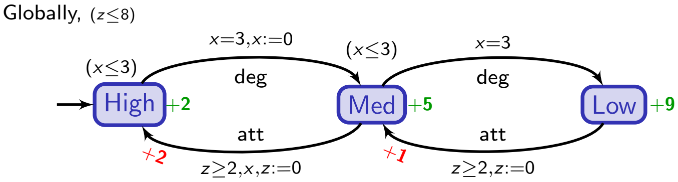
$\rightsquigarrow$ compute optimal infinite schedules that minimize
discounted cost over time



if $\lambda = e^{-1}$, the discounted cost of that infinite schedule is ≈ 2.16

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata *(INFINITY'08)*.

Going further 3: discounted-time cost optimization



$\rightsquigarrow$ compute optimal infinite schedules that minimize
discounted cost over time

Theorem [FL08]

In weighted timed automata, the optimal discounted cost is computable in EXPTIME.

$\rightsquigarrow$ the corner-point abstraction can be used

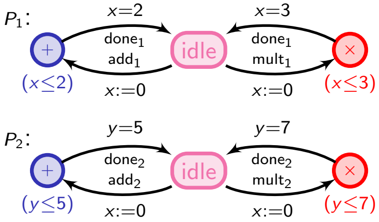
[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata *(INFINITY'08)*.

Outline

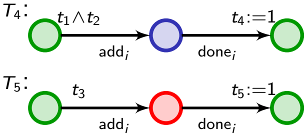
1. Timed automata
2. Weighted timed automata
3. Timed games
4. Weighted timed games
5. Conclusion

Modelling the task graph scheduling problem

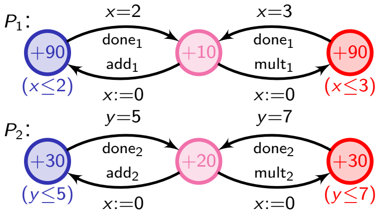
- Processors



- Tasks

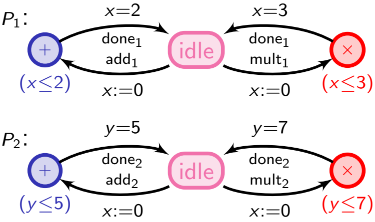


- Modelling energy

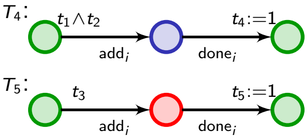


Modelling the task graph scheduling problem

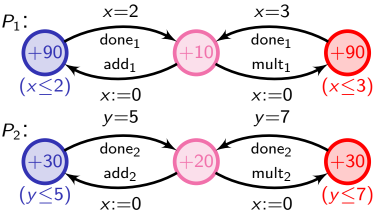
- Processors



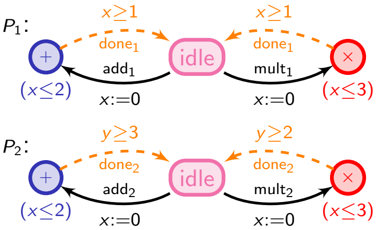
- Tasks



- Modelling energy



- Modelling uncertainty



Modelling the task graph scheduling problem

- Processors

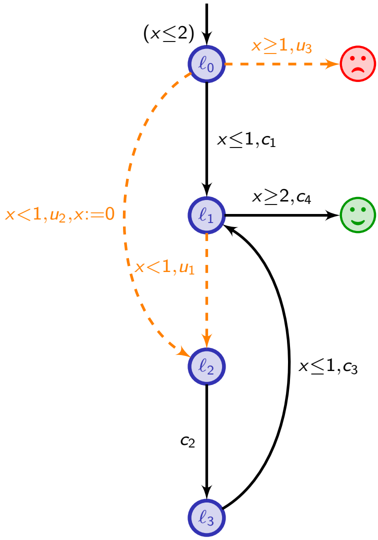
- Tasks

A (good) schedule is a strategy in the product game (with a low cost)

- Modelling energy

- Modelling uncertainty

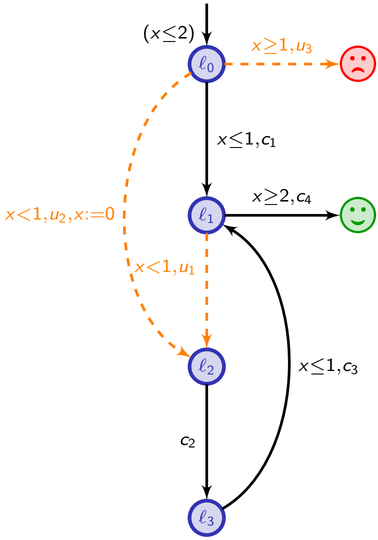
An example of a timed game



Rule of the game

- Aim: avoid ☹ and reach ☺

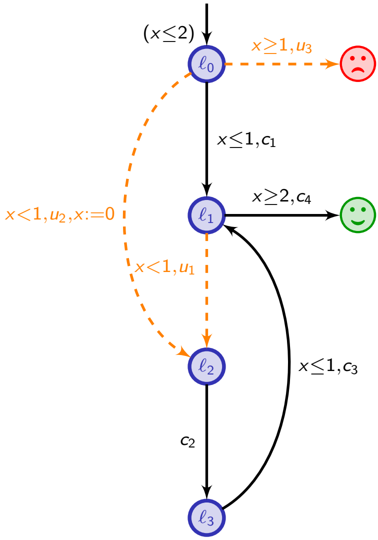
An example of a timed game



Rule of the game

- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

An example of a timed game

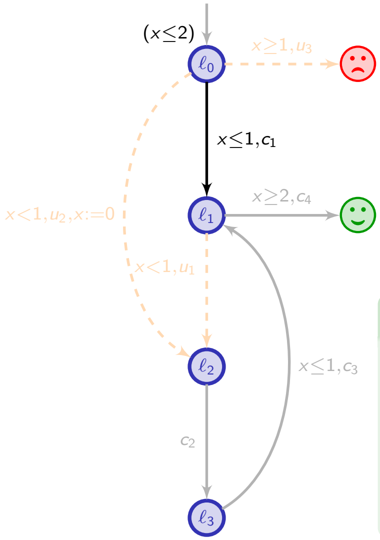


Rule of the game

- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

$$f : \text{history} \mapsto (\text{delay}, \text{cont. transition})$$

An example of a timed game



Rule of the game

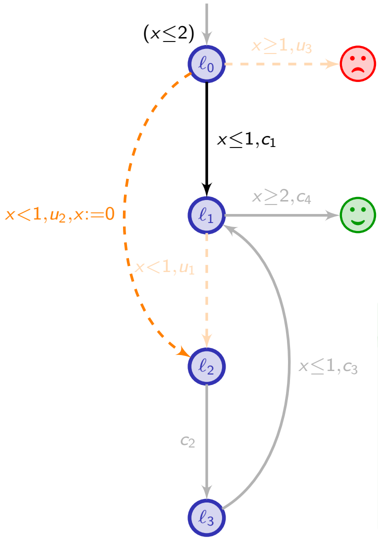
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

$$f : \text{history} \mapsto (\text{delay}, \text{cont. transition})$$

A (memoryless) winning strategy

- from $(\ell_0, 0)$, play $(0.5, c_1)$

An example of a timed game



Rule of the game

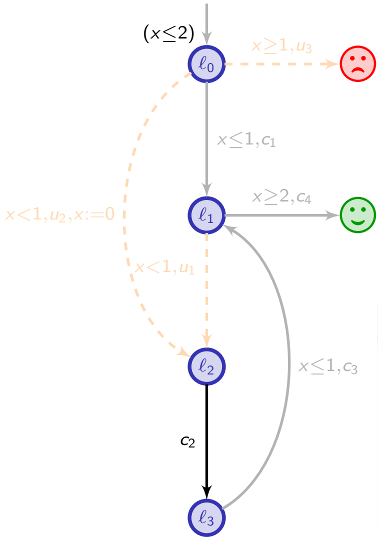
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

$$f : \text{history} \mapsto (\text{delay}, \text{cont. transition})$$

A (memoryless) winning strategy

- from $(\ell_0, 0)$, play $(0.5, c_1)$
  $\rightsquigarrow$ can be preempted by u_2

An example of a timed game



Rule of the game

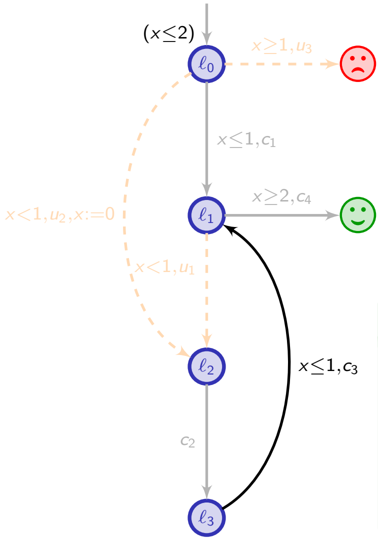
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

$$f : \text{history} \mapsto (\text{delay}, \text{cont. transition})$$

A (memoryless) winning strategy

- from $(\ell_0, 0)$, play $(0.5, c_1)$
  $\rightsquigarrow$ can be preempted by u_2
- from $(\ell_2, \star)$, play $(1 - \star, c_2)$

An example of a timed game



Rule of the game

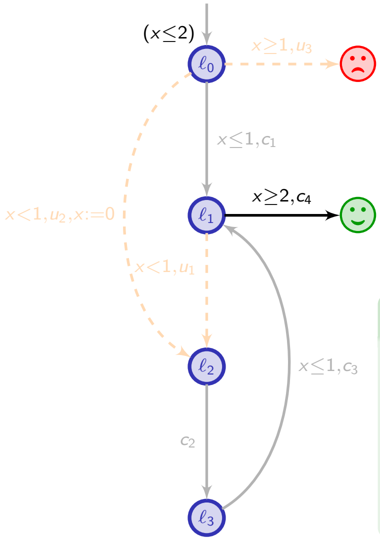
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

$$f : \text{history} \mapsto (\text{delay}, \text{cont. transition})$$

A (memoryless) winning strategy

- from $(\ell_0, 0)$, play $(0.5, c_1)$
  $\rightsquigarrow$ can be preempted by u_2
- from $(\ell_2, \star)$, play $(1 - \star, c_2)$
- from $(\ell_3, 1)$, play $(0, c_3)$

An example of a timed game



Rule of the game

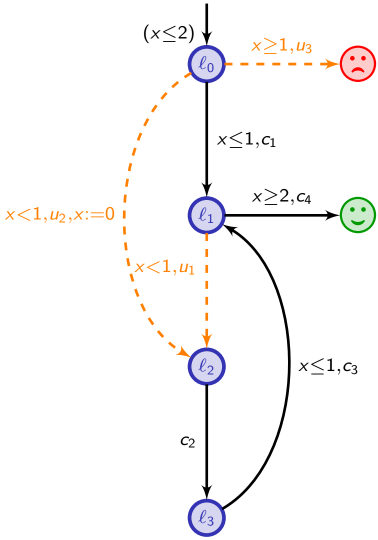
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

$$f : \text{history} \mapsto (\text{delay}, \text{cont. transition})$$

A (memoryless) winning strategy

- from $(\ell_0, 0)$, play $(0.5, c_1)$
  $\rightsquigarrow$ can be preempted by u_2
- from $(\ell_2, \star)$, play $(1-\star, c_2)$
- from $(\ell_3, 1)$, play $(0, c_3)$
- from $(\ell_1, 1)$, play $(1, c_4)$

An example of a timed game



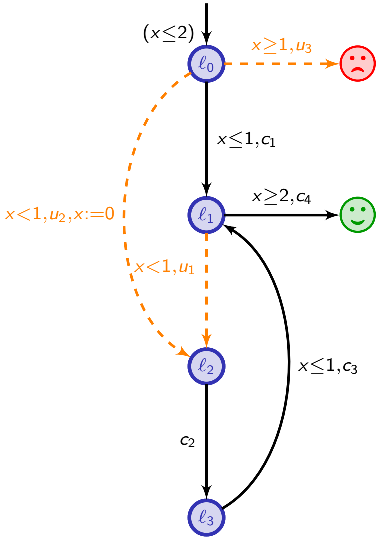
Rule of the game

- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

$$f : \text{history} \mapsto (\text{delay}, \text{cont. transition})$$

Problems to be considered

An example of a timed game



Rule of the game

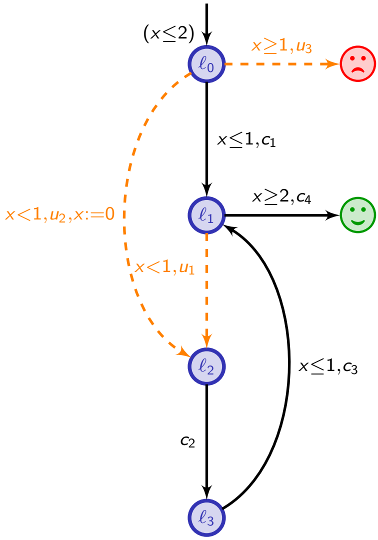
- Aim: avoid and reach
- How do we play? According to a strategy:

$$f : \text{history} \mapsto (\text{delay}, \text{cont. transition})$$

Problems to be considered

- Does there exist a winning strategy?

An example of a timed game



Rule of the game

- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

$$f : \text{history} \mapsto (\text{delay}, \text{cont. transition})$$

Problems to be considered

- Does there exist a winning strategy?
- If yes, compute one (as simple as possible).

Decidability of timed games

Theorem [AMPS98,HK99]

Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and "region-based" strategies are sufficient.

[AMPS98] Asarin, Maler, Pnueli, Sifakis. Controller synthesis for timed automata *(SSC'98)*.
[HK99] Henzinger, Kopke. Discrete-time control for rectangular hybrid automata *(Theoretical Computer Science)*.

Decidability of timed games

Theorem [AMPS98,HK99]

Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and "region-based" strategies are sufficient.

$\rightsquigarrow$ classical regions are sufficient for solving such problems

[AMPS98] Asarin, Maler, Pnueli, Sifakis. Controller synthesis for timed automata *(SSC'98)*.
[HK99] Henzinger, Kopke. Discrete-time control for rectangular hybrid automata *(Theoretical Computer Science)*.

Decidability of timed games

Theorem [AMPS98,HK99]

Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and “region-based” strategies are sufficient.

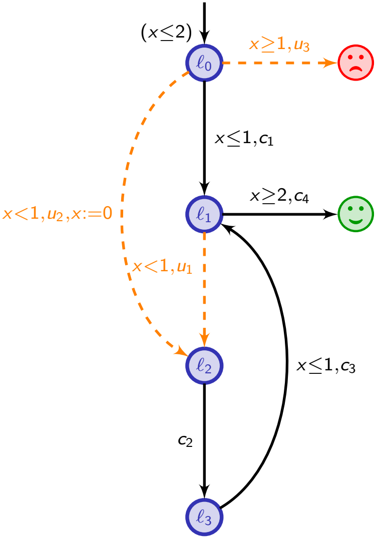
$\rightsquigarrow$ classical regions are sufficient for solving such problems

Theorem [AM99,BHPR07,JT07]

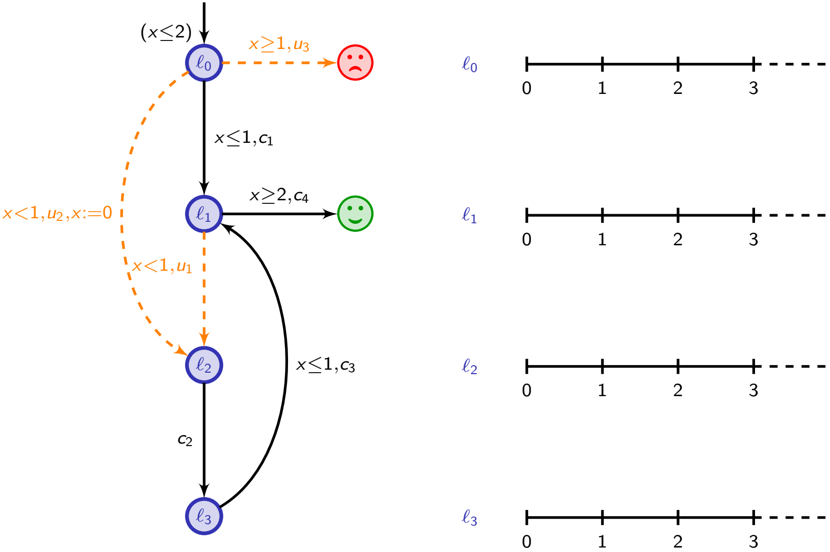
Optimal-time reachability timed games are decidable and EXPTIME-complete.

[AM99] Asarin, Maler. As soon as possible: time optimal control for timed automata *(HSCC'99)*.
[BHPR07] Brihaye, Henzinger, Prabhu, Raskin. Minimum-time reachability in timed games *(ICALP'07)*.
[JT07] Jurdziński, Trivedi. Reachability-time games on timed automata *(ICALP'07)*.

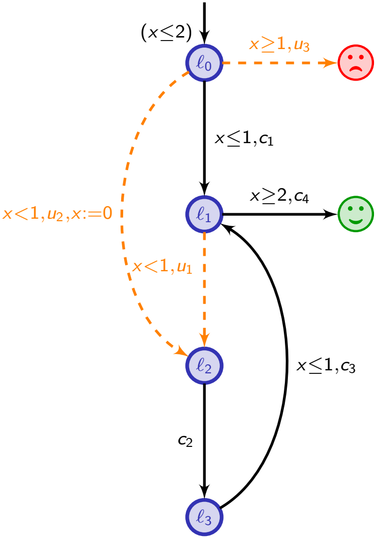
Back to the example: computing winning states

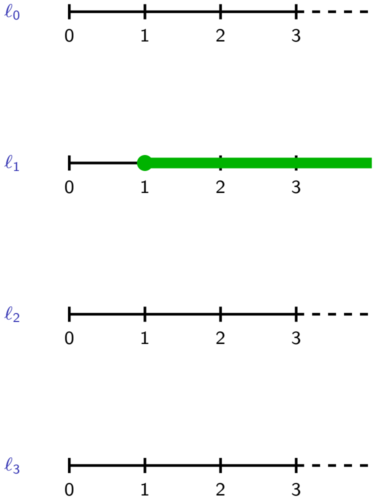


Back to the example: computing winning states

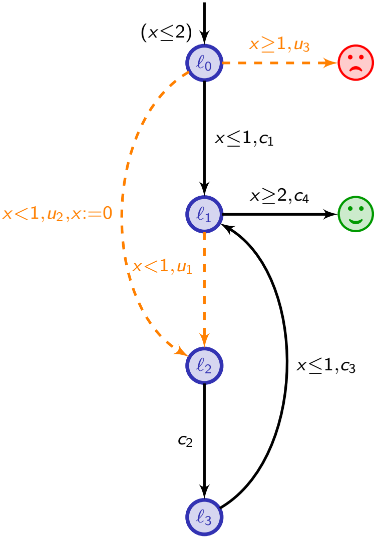


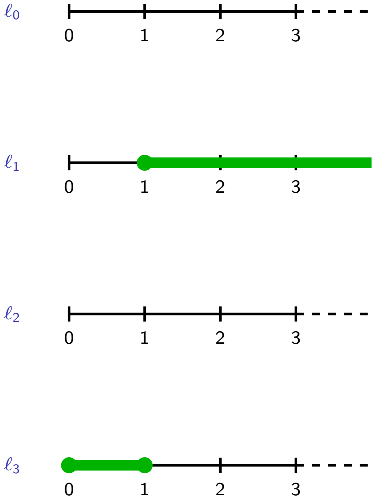
Back to the example: computing winning states



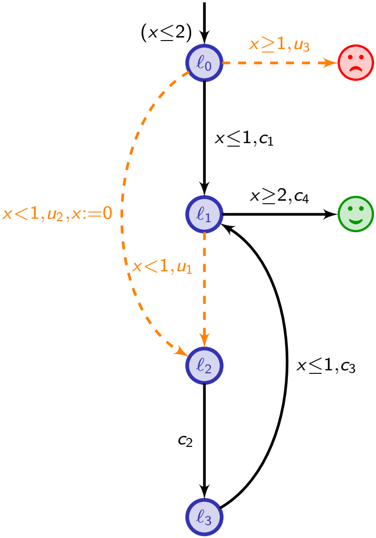


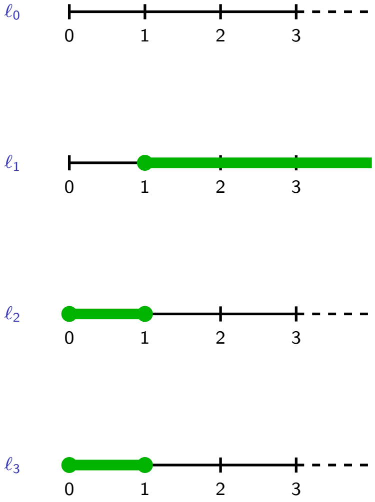
Back to the example: computing winning states



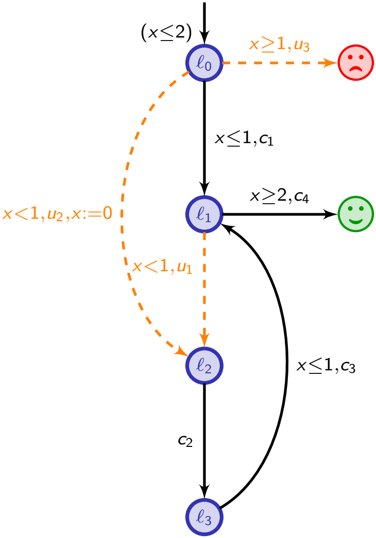


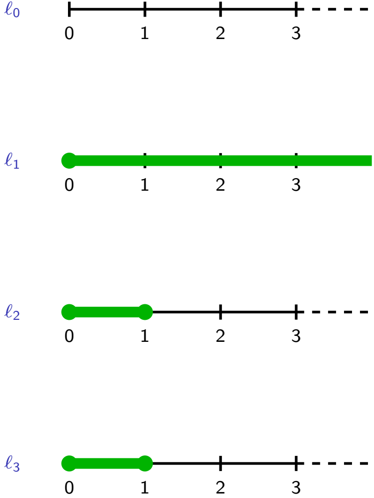
Back to the example: computing winning states



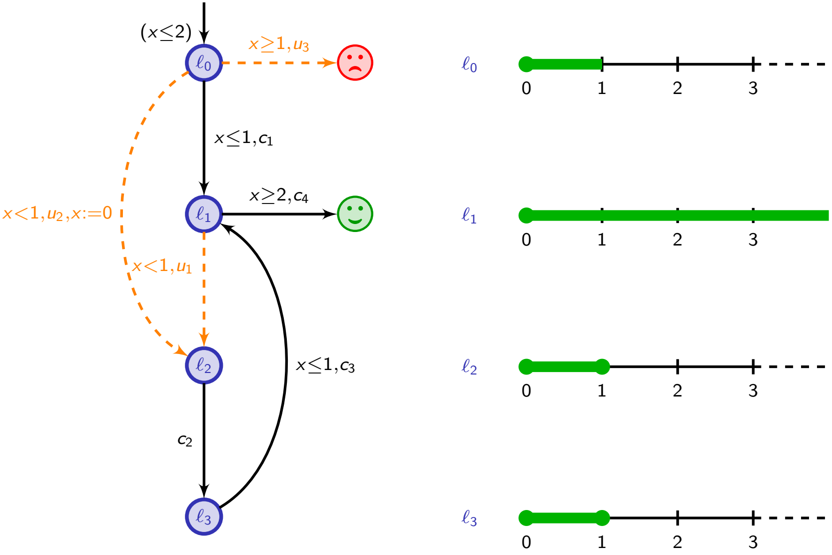


Back to the example: computing winning states

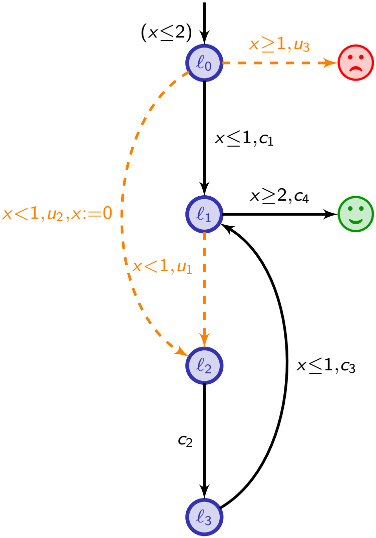


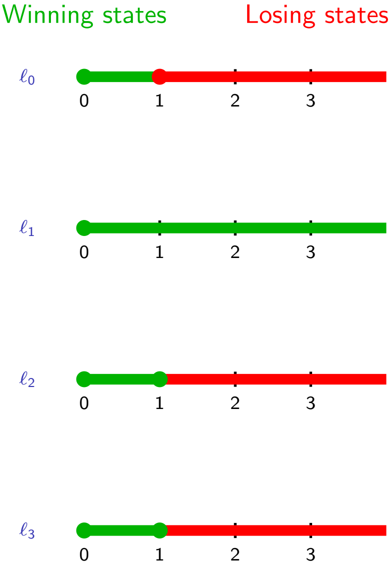


Back to the example: computing winning states



Back to the example: computing winning states





Decidability *via* attractors

Skip attractors

Decidability *via* attractors

- $\mathsf{Pred}^a(X) = \{\bullet \mid \bullet \xrightarrow{a} \bullet \in X\}$

Decidability *via* attractors

- $\text{Pred}^a(X) = \{\bullet \mid \bullet \xrightarrow{a} \bullet \in X\}$
- controllable and uncontrollable discrete predecessors:

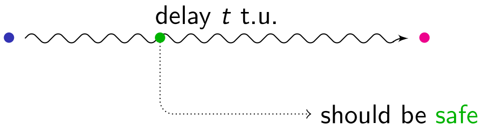
$$\text{cPred}(X) = \bigcup_{a \text{ cont.}} \text{Pred}^a(X) \qquad \text{uPred}(X) = \bigcup_{a \text{ uncont.}} \text{Pred}^a(X)$$

Decidability *via* attractors

- $\mathrm{Pred}^a(X) = \{\bullet \mid \bullet \xrightarrow{a} \bullet \in X\}$
- controllable and uncontrollable discrete predecessors:

$$\mathrm{cPred}(X) = \bigcup_{a \text{ cont.}} \mathrm{Pred}^a(X) \qquad \mathrm{uPred}(X) = \bigcup_{a \text{ uncont.}} \mathrm{Pred}^a(X)$$

- time controllable predecessors:

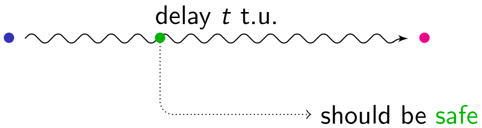


Decidability *via* attractors

- $\mathrm{Pred}^a(X) = \{\bullet \mid \bullet \xrightarrow{a} \bullet \in X\}$
- controllable and uncontrollable discrete predecessors:

$$\mathrm{cPred}(X) = \bigcup_{a \text{ cont.}} \mathrm{Pred}^a(X) \qquad \mathrm{uPred}(X) = \bigcup_{a \text{ uncont.}} \mathrm{Pred}^a(X)$$

- time controllable predecessors:



$$\mathrm{Pred}_\delta(X, \mathrm{Safe}) = \{\bullet \mid \exists t \geq 0,\ \bullet \xrightarrow{\delta(t)} \bullet$$
$$\text{and } \forall 0 \leq t' \leq t,\ \bullet \xrightarrow{\delta(t')} \bullet \in \mathrm{Safe}\}$$

Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \mathsf{Pred}_\delta(\mathsf{cPred}(X), \neg\mathsf{uPred}(\neg X))$$

Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \mathsf{Pred}_\delta(\mathsf{cPred}(X), \neg\mathsf{uPred}(\neg X))$$

- The states from which one can ensure ☺ in no more than 1 step is:

$$\mathsf{Attr}_1(☺) = \pi(☺)$$

Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \mathsf{Pred}_\delta(\mathsf{cPred}(X), \neg\mathsf{uPred}(\neg X))$$

- The states from which one can ensure ☺ in no more than 1 step is:

$$\mathsf{Attr}_1(☺) = \pi(☺)$$

- The states from which one can ensure ☺ in no more than 2 steps is:

$$\mathsf{Attr}_2(☺) = \pi(\mathsf{Attr}_1(☺))$$

Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \mathsf{Pred}_\delta(\mathsf{cPred}(X), \neg\mathsf{uPred}(\neg X))$$

- The states from which one can ensure $\smiley$ in no more than 1 step is:

$$\mathsf{Attr}_1(\smiley) = \pi(\smiley)$$

- The states from which one can ensure $\smiley$ in no more than 2 steps is:

$$\mathsf{Attr}_2(\smiley) = \pi(\mathsf{Attr}_1(\smiley))$$

- ...

Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \mathrm{Pred}_\delta(\mathrm{cPred}(X), \neg \mathrm{uPred}(\neg X))$$

- The states from which one can ensure ☺ in no more than 1 step is:

$$\mathrm{Attr}_1(☺) = \pi(☺)$$

- The states from which one can ensure ☺ in no more than 2 steps is:

$$\mathrm{Attr}_2(☺) = \pi(\mathrm{Attr}_1(☺))$$

- ...
- The states from which one can ensure ☺ in no more than n steps is:

$$\mathrm{Attr}_n(☺) \quad = \quad \pi(\mathrm{Attr}_{n-1}(☺))$$

Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \mathsf{Pred}_\delta(\mathsf{cPred}(X), \neg\mathsf{uPred}(\neg X))$$

- The states from which one can ensure ☺ in no more than 1 step is:

$$\mathsf{Attr}_1(\smiley) = \pi(\smiley)$$

- The states from which one can ensure ☺ in no more than 2 steps is:

$$\mathsf{Attr}_2(\smiley) = \pi(\mathsf{Attr}_1(\smiley))$$

- ...
- The states from which one can ensure ☺ in no more than *n* steps is:

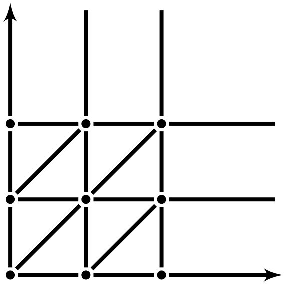
$$\begin{array}{rcl} \mathsf{Attr}_n(\smiley) & = & \pi(\mathsf{Attr}_{n-1}(\smiley)) \\ & = & \pi^n(\smiley) \end{array}$$

Stability w.r.t. regions

- if X is a union of regions, then:
  - $\mathsf{Pred}_a(X)$ is a union of regions,
  - and so are $\mathsf{cPred}(X)$ and $\mathsf{uPred}(X)$.

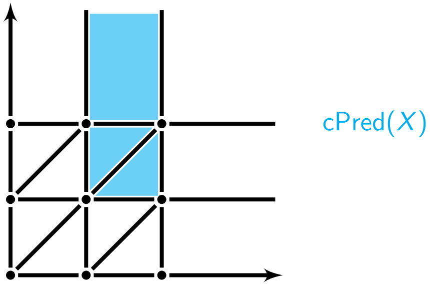
Stability w.r.t. regions

- if X is a union of regions, then:
  - $\mathsf{Pred}_a(X)$ is a union of regions,
  - and so are $\mathsf{cPred}(X)$ and $\mathsf{uPred}(X)$.
- Does π also preserve unions of regions?



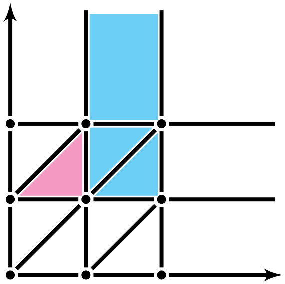
Stability w.r.t. regions

- if X is a union of regions, then:
  - $\mathsf{Pred}_a(X)$ is a union of regions,
  - and so are $\mathsf{cPred}(X)$ and $\mathsf{uPred}(X)$.
- Does π also preserve unions of regions?



Stability w.r.t. regions

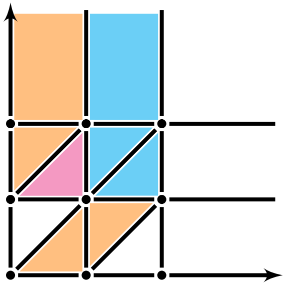
- if X is a union of regions, then:
  - $\mathrm{Pred}_a(X)$ is a union of regions,
  - and so are $\mathrm{cPred}(X)$ and $\mathrm{uPred}(X)$.
- Does π also preserve unions of regions?



cPred(X)
uPred($\neg X$)

Stability w.r.t. regions

- if X is a union of regions, then:
  - $\mathsf{Pred}_a(X)$ is a union of regions,
  - and so are $\mathsf{cPred}(X)$ and $\mathsf{uPred}(X)$.
- Does π also preserve unions of regions?



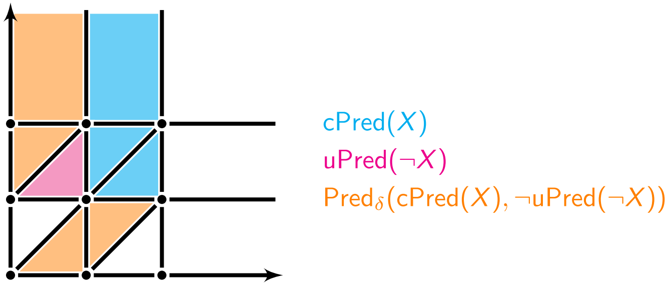
$\mathsf{cPred}(X)$

$\mathsf{uPred}(\neg X)$

$\mathsf{Pred}_\delta(\mathsf{cPred}(X), \neg\mathsf{uPred}(\neg X))$

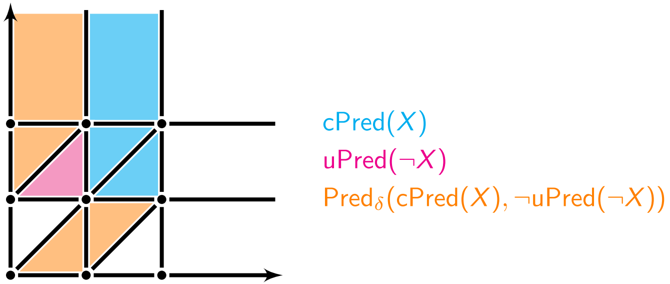
Stability w.r.t. regions

- if X is a union of regions, then:
  - $\mathrm{Pred}_a(X)$ is a union of regions,
  - and so are $\mathrm{cPred}(X)$ and $\mathrm{uPred}(X)$.
- Does π also preserve unions of regions? Yes!



Stability w.r.t. regions

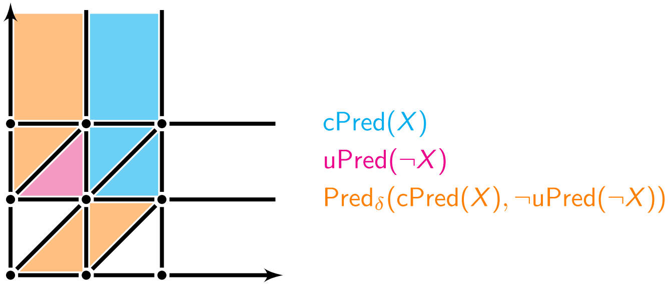
- if X is a union of regions, then:
  - $\mathsf{Pred}_a(X)$ is a union of regions,
  - and so are $\mathsf{cPred}(X)$ and $\mathsf{uPred}(X)$.
- Does π also preserve unions of regions? Yes!



(but it generates non-convex unions of regions...)

Stability w.r.t. regions

- if X is a union of regions, then:
  - $\mathrm{Pred}_a(X)$ is a union of regions,
  - and so are $\mathrm{cPred}(X)$ and $\mathrm{uPred}(X)$.
- Does π also preserve unions of regions? Yes!

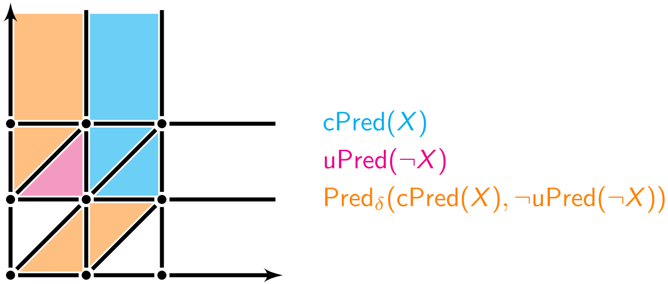


(but it generates non-convex unions of regions...)

$\rightsquigarrow$ the computation of $\pi^*(☺)$ terminates!

Stability w.r.t. regions

- if X is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does π also preserve unions of regions? Yes!



(but it generates non-convex unions of regions...)

$\rightsquigarrow$ the computation of $\pi^*(☺)$ terminates!
... and is correct

Timed games with a safety objective

- We can use operator $\widetilde{\pi}$ defined by

$$\widetilde{\pi}(X) = \mathsf{Pred}_\delta(X \cap \mathsf{cPred}(X), \neg\mathsf{uPred}(\neg X))$$

instead of π, and compute $\widetilde{\pi}^*(\neg\text{☹})$

Timed games with a safety objective

- We can use operator $\widetilde{\pi}$ defined by

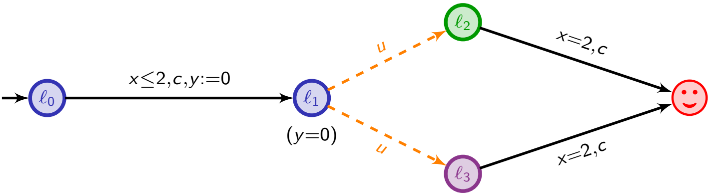
$$\widetilde{\pi}(X) = \mathsf{Pred}_\delta(X \cap \mathsf{cPred}(X), \neg \mathsf{uPred}(\neg X))$$

  instead of π, and compute $\widetilde{\pi}^*(\neg \frownie)$

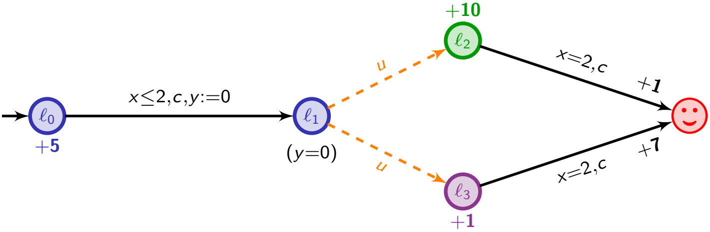
- It is also stable w.r.t. regions.

Outline

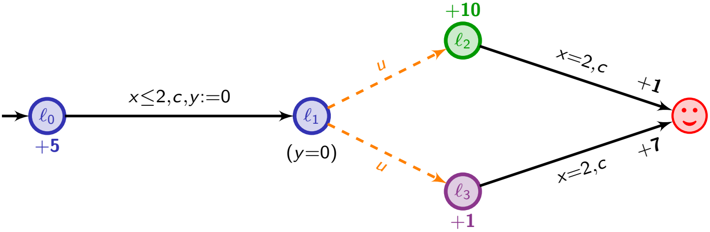
A simple timed game



A simple weighted timed game

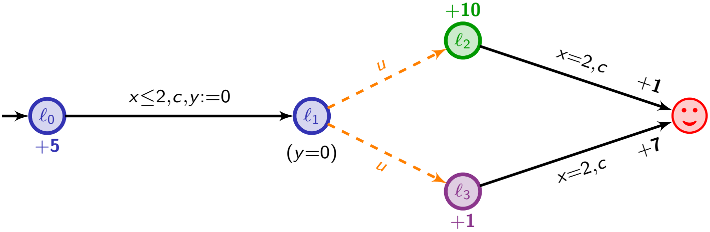


A simple weighted timed game



Question: what is the optimal cost we can ensure while reaching ☺?

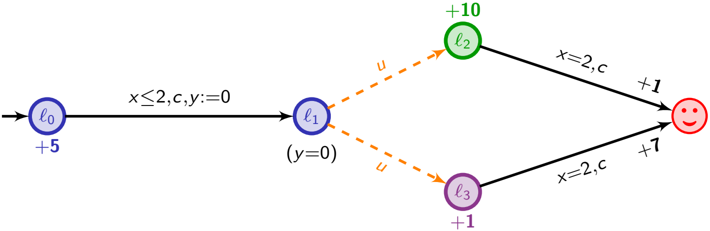
A simple weighted timed game



Question: what is the optimal cost we can ensure while reaching 🙂 ?

$$5t + 10(2-t) + 1$$

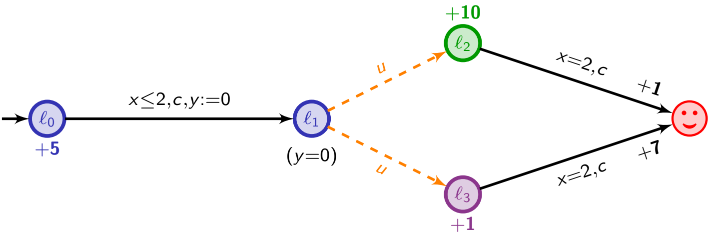
A simple weighted timed game



Question: what is the optimal cost we can ensure while reaching 🙂?

$$5t + 10(2-t) + 1 \;,\; 5t + (2-t) + 7$$

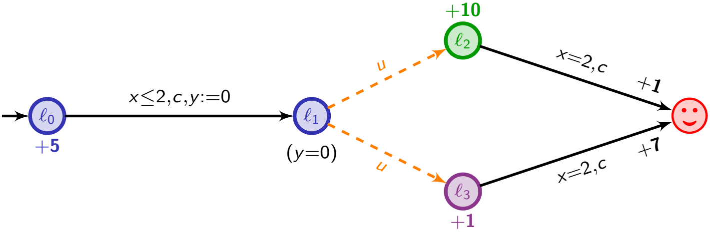
A simple weighted timed game



Question: what is the optimal cost we can ensure while reaching 🙂 ?

$$\max\ (\ 5t + 10(2-t) + 1\ ,\ 5t + (2-t) + 7\ )$$

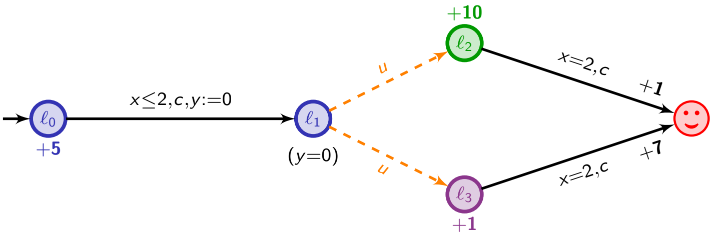
A simple weighted timed game



Question: what is the optimal cost we can ensure while reaching 🙂?

$$\inf_{0\leq t\leq 2} \max\left(\, 5t+10(2-t)+1 \,,\, 5t+(2-t)+7 \,\right) = 14+\frac{1}{3}$$

A simple weighted timed game



Question: what is the optimal cost we can ensure while reaching 🙂?

$$\inf_{0\leq t\leq 2} \max\left(\ 5t + 10(2-t) + 1\ ,\ 5t + (2-t) + 7\ \right) = 14 + \frac{1}{3}$$

$\rightsquigarrow$ *strategy:* wait in ℓ_0, and when $t = \frac{4}{3}$, go to ℓ_1

Optimal reachability in weighted timed games (1)

This topic has been fairly hot these last fifteen years...

[LMM02,ABM04,BCFL04,BBR05,BBM06,BLMR06,Rut11,HIM13,BGK+14]

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.
[ABM04] Alur, Bernardsky, Madhusudan. Optimal reachability in weighted timed games *(ICALP'04)*.
[BCFL04] Bouyer, Cassez, Fleury, Larsen. Optimal strategies in priced timed game automata *(FSTTCS'04)*.
[BBR05] Brihaye, Bruyère, Raskin. On optimal timed strategies *(FORMATS'05)*.
[BBM06] Bouyer, Brihaye, Markey. Improved undecidability results on weighted timed automata *(Information Processing Letters)*.
[BLMR06] Bouyer, Larsen, Markey, Rasmussen. Almost-optimal strategies in one-clock priced timed automata *(FSTTCS'06)*.
[Rut11] Rutkowski. Two-player reachability-price games on single-clock timed automata *(QAPL'11)*.
[HIM13] Hansen, Ibsen-Jensen, Miltersen. A faster algorithm for solving one-clock priced timed games *(CONCUR'13)*.
[BGK+14] Brihaye, Geeraerts, Krishna, Manasa, Monmege, Trivedi. Adding Negative Prices to Priced Timed Games *(CONCUR'14)*.

Optimal reachability in weighted timed games (1)

This topic has been fairly hot these last fifteen years...

[LMM02,ABM04,BCFL04,BBR05,BBM06,BLMR06,Rut11,HIM13,BGK+14]

[LMM02]

Tree-like weighted timed games can be solved in 2EXPTIME.

Optimal reachability in weighted timed games (1)

This topic has been fairly hot these last fifteen years...

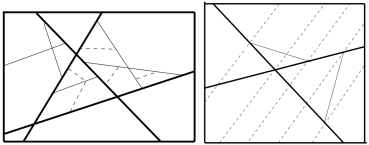
[LMM02,ABM04,BCFL04,BBR05,BBM06,BLMR06,Rut11,HIM13,BGK+14]

[LMM02]

Tree-like weighted timed games can be solved in 2EXPTIME.

[ABM04,BCFL04]

Depth-k weighted timed games can be solved in EXPTIME. There is a symbolic algorithm to solve weighted timed games **with a strongly non-Zeno cost**.



Optimal reachability in weighted timed games (2)

[BBR05,BBM06]

In weighted timed games, the optimal cost cannot be computed, as soon as games have three clocks or more.

Optimal reachability in weighted timed games (2)

[BBR05,BBM06]

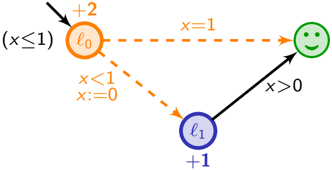
In weighted timed games, the optimal cost cannot be computed, as soon as games have three clocks or more.

[BLMR06,Rut11,HIM13,BGK+14]

Turn-based optimal timed games are decidable in EXPTIME (resp. PTIME) when automata have a single clock (resp. with two rates). They are PTIME-hard.

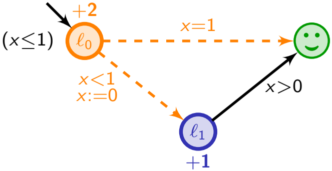
What is easier with a single clock?

- Memoryless strategies can be non-optimal...



What is easier with a single clock?

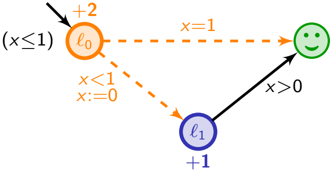
- Memoryless strategies can be non-optimal...



... but memoryless almost-optimal strategies will be sufficient.

What is easier with a single clock?

- Memoryless strategies can be non-optimal...

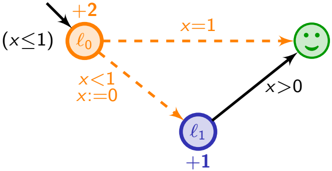


... but memoryless almost-optimal strategies will be sufficient.

- Key: resetting the clock somehow resets the history...

What is easier with a single clock?

- Memoryless strategies can be non-optimal...

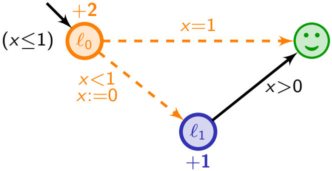


... but memoryless almost-optimal strategies will be sufficient.

- Key: resetting the clock somehow resets the history...
- By unfolding and removing one by one the locations,we can synthesize memoryless almost-optimal winning strategies.

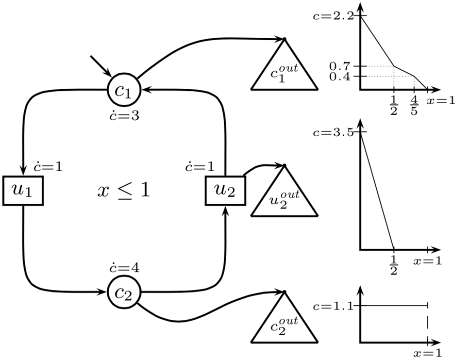
What is easier with a single clock?

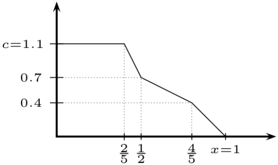
- Memoryless strategies can be non-optimal...



... but memoryless almost-optimal strategies will be sufficient.

- Key: resetting the clock somehow resets the history...
- By unfolding and removing one by one the locations,we can synthesize memoryless almost-optimal winning strategies.
- Rather involved proofs of correctness





$$\sigma(c_2, x) = \begin{cases} c_2^{out} & \text{if } 0 \leq x < 2/5 \\ c_2 & \text{if } 2/5 \leq x < 1/2 \\ u_2 & \text{if } 1/2 \leq x \leq 1 \end{cases}$$

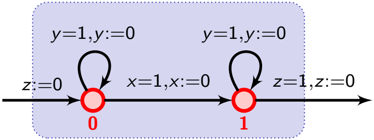
Computing the optimal cost: why is that hard?

Given two clocks x and y, we can check whether $y = 2x$.

Computing the optimal cost: why is that hard?

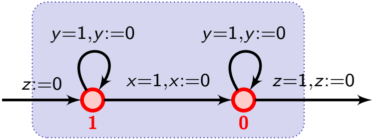
Given two clocks x and y, we can check whether $y = 2x$.

$\mathsf{Add}^+(x)$



The cost is increased by x_0

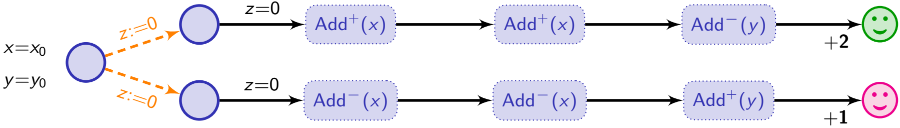
$\mathsf{Add}^-(x)$



The cost is increased by $1-x_0$

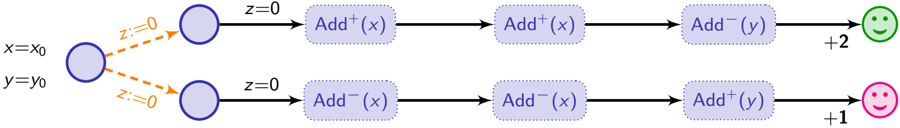
Computing the optimal cost: why is that hard?

Given two clocks x and y, we can check whether $y = 2x$.



Computing the optimal cost: why is that hard?

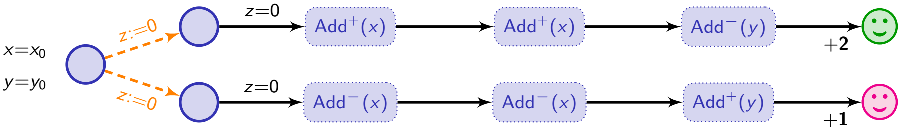
Given two clocks x and y, we can check whether $y = 2x$.



- In 🙂, cost $= 2x_0 + (1 - y_0) + 2$

Computing the optimal cost: why is that hard?

Given two clocks x and y, we can check whether $y = 2x$.

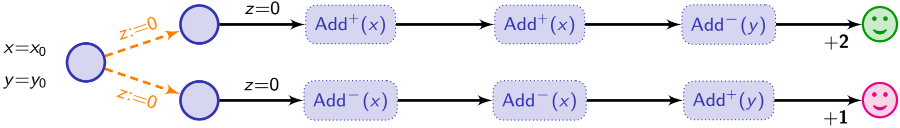


- In 🙂 (green), cost $= 2x_0 + (1 - y_0) + 2$

  In 🙂 (pink), cost $= 2(1 - x_0) + y_0 + 1$

Computing the optimal cost: why is that hard?

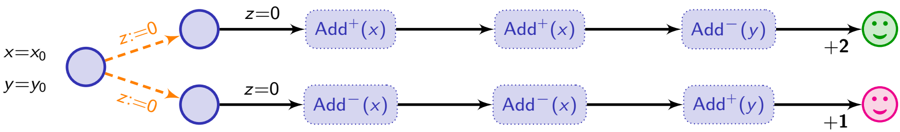
Given two clocks x and y, we can check whether $y = 2x$.



- In , cost $= 2x_0 + (1 - y_0) + 2$
  In , cost $= 2(1 - x_0) + y_0 + 1$
- if $y_0 < 2x_0$, player 2 chooses the first branch: cost > 3

Computing the optimal cost: why is that hard?

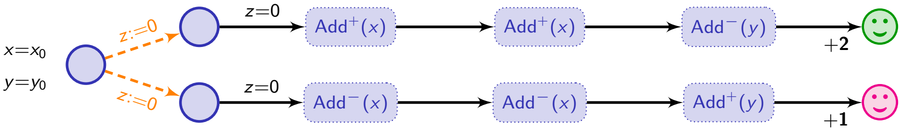
Given two clocks x and y, we can check whether $y = 2x$.



- In , cost $= 2x_0 + (1 - y_0) + 2$
  In , cost $= 2(1 - x_0) + y_0 + 1$
- if $y_0 < 2x_0$, player 2 chooses the first branch: cost > 3
  if $y_0 > 2x_0$, player 2 chooses the second branch: cost > 3

Computing the optimal cost: why is that hard?

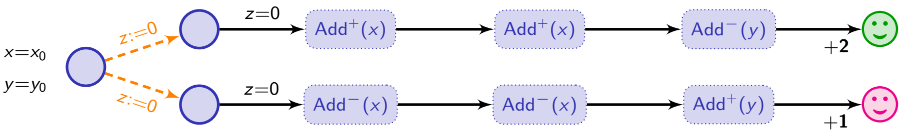
Given two clocks x and y, we can check whether $y = 2x$.



- In , cost $= 2x_0 + (1 - y_0) + 2$
  In , cost $= 2(1 - x_0) + y_0 + 1$
- if $y_0 < 2x_0$, player 2 chooses the first branch: cost > 3
  if $y_0 > 2x_0$, player 2 chooses the second branch: cost > 3
  if $y_0 = 2x_0$, in both branches, cost $= 3$

Computing the optimal cost: why is that hard?

Given two clocks x and y, we can check whether $y = 2x$.



- In , cost $= 2x_0 + (1 - y_0) + 2$

  In , cost $= 2(1 - x_0) + y_0 + 1$

- if $y_0 < 2x_0$, player 2 chooses the first branch: cost > 3

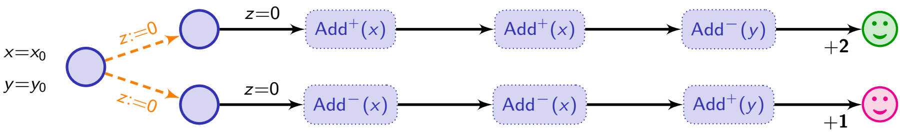
  if $y_0 > 2x_0$, player 2 chooses the second branch: cost > 3

  if $y_0 = 2x_0$, in both branches, cost $= 3$

  $\rightsquigarrow$ player 2 can enforce cost $3 + |y_0 - 2x_0|$

Computing the optimal cost: why is that hard?

Given two clocks x and y, we can check whether $y = 2x$.



- In , cost $= 2x_0 + (1 - y_0) + 2$
  In , cost $= 2(1 - x_0) + y_0 + 1$
- if $y_0 < 2x_0$, player 2 chooses the first branch: cost > 3
  if $y_0 > 2x_0$, player 2 chooses the second branch: cost > 3
  if $y_0 = 2x_0$, in both branches, cost $= 3$
  $\rightsquigarrow$ player 2 can enforce cost $3 + |y_0 - 2x_0|$
- Player 1 has a winning strategy with cost ≤ 3 iff $y_0 = 2x_0$

Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$

Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \qquad \text{and} \qquad y = \frac{1}{2^{c_2}}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.

Computing the optimal cost: why is that hard?

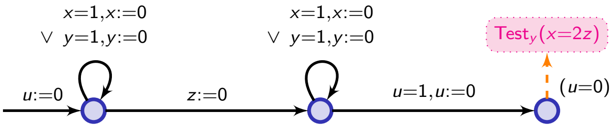
Player 1 will simulate a two-counter machine:

- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.

Globally, $(x \leq 1, y \leq 1, u \leq 1)$



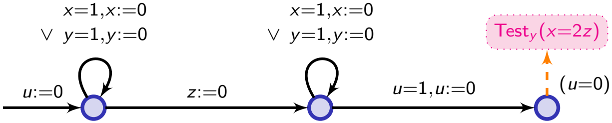
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \qquad \text{and} \qquad y = \frac{1}{2^{c_2}}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.



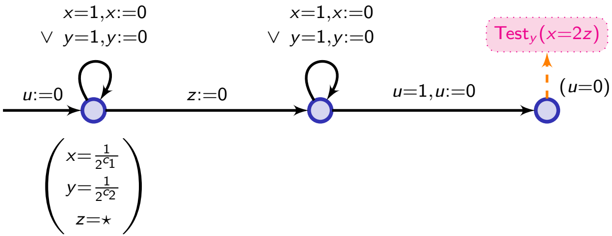
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.



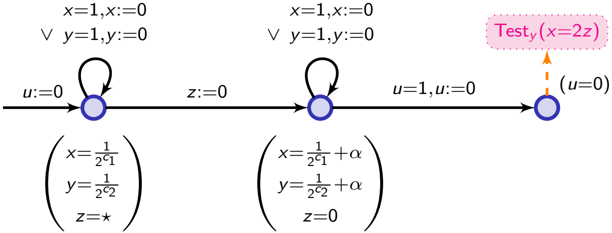
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.



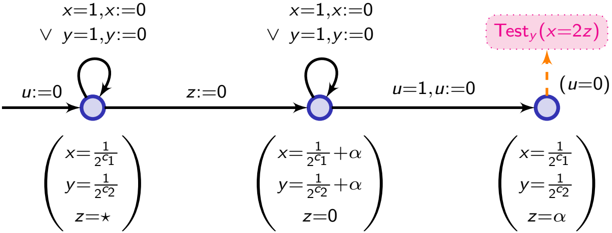
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.



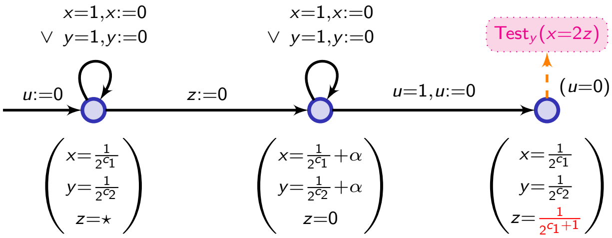
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

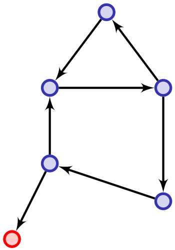
- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$

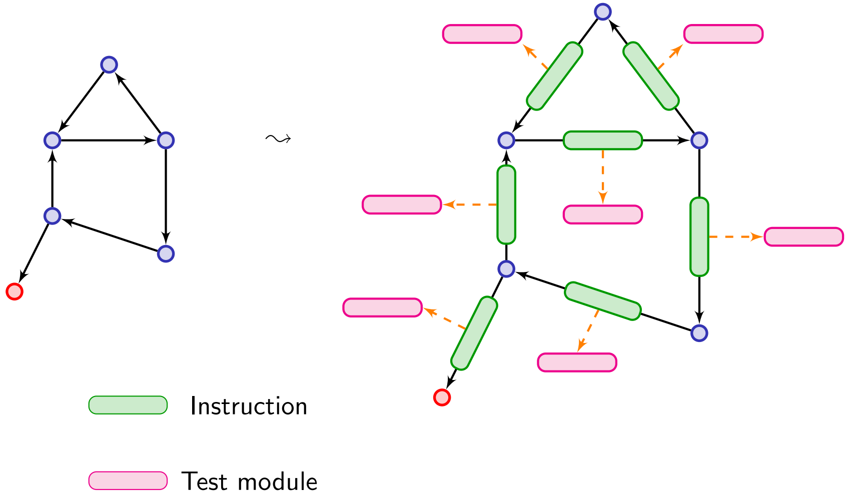
The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.



Shape of the reduction



Shape of the reduction



Are we done?

Are we done? No! Let's be a bit more precise!

Are we done? No! Let's be a bit more precise!

Given $\mathcal{G}$ a weighted timed game,

- a strategy σ is winning whenever all its outcomes are winning;

Are we done? No! Let's be a bit more precise!

Given $\mathcal{G}$ a weighted timed game,

- a strategy σ is winning whenever all its outcomes are winning;
- Cost of a winning strategy σ:

$$\text{cost}(\sigma) = \sup\{\text{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target}\}$$

Are we done? No! Let's be a bit more precise!

Given $\mathcal{G}$ a weighted timed game,

- a strategy σ is winning whenever all its outcomes are winning;
- Cost of a winning strategy σ:

$$\mathsf{cost}(\sigma) = \sup\{\mathsf{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target}\}$$

- Optimal cost:

$$\mathsf{optcost}_{\mathcal{G}} = \inf_{\sigma \text{ winning strat.}} \mathsf{cost}(\sigma)$$

(set it to $+\infty$ if there is no winning strategy)

Are we done? No! Let's be a bit more precise!

Given $\mathcal{G}$ a weighted timed game,

- a strategy σ is winning whenever all its outcomes are winning;
- Cost of a winning strategy σ:

$$\text{cost}(\sigma) = \sup\{\text{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target}\}$$

- Optimal cost:

$$\text{optcost}_{\mathcal{G}} = \inf_{\sigma \text{ winning strat.}} \text{cost}(\sigma)$$

(set it to $+\infty$ if there is no winning strategy)

Two problems of interest

- The value problem asks, given $\mathcal{G}$ and a threshold $\bowtie c$, whether $\text{optcost}_{\mathcal{G}} \bowtie c$?

Are we done? No! Let's be a bit more precise!

Given $\mathcal{G}$ a weighted timed game,

- a strategy σ is winning whenever all its outcomes are winning;
- Cost of a winning strategy σ:

$$\text{cost}(\sigma) = \sup\{\text{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target}\}$$

- Optimal cost:

$$\text{optcost}_{\mathcal{G}} = \inf_{\sigma \text{ winning strat.}} \text{cost}(\sigma)$$

(set it to $+\infty$ if there is no winning strategy)

Two problems of interest

- The value problem asks, given $\mathcal{G}$ and a threshold $\bowtie c$, whether $\text{optcost}_{\mathcal{G}} \bowtie c$?
- The existence problem asks, given $\mathcal{G}$ and a threshold $\bowtie c$, whether there exists a winning strategy in $\mathcal{G}$ such that $\text{cost}(\sigma) \bowtie c$?

Are we done? No! Let's be a bit more precise!

Given $\mathcal{G}$ a weighted timed game,

- a strategy σ is winning whenever all its outcomes are winning;
- Cost of a winning strategy σ:

$$\text{cost}(\sigma) = \sup\{\text{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target}\}$$

- Optimal cost:

$$\text{optcost}_{\mathcal{G}} = \inf_{\sigma \text{ winning strat.}} \text{cost}(\sigma)$$

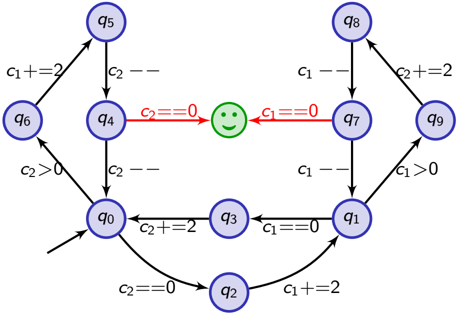
(set it to $+\infty$ if there is no winning strategy)

Two problems of interest

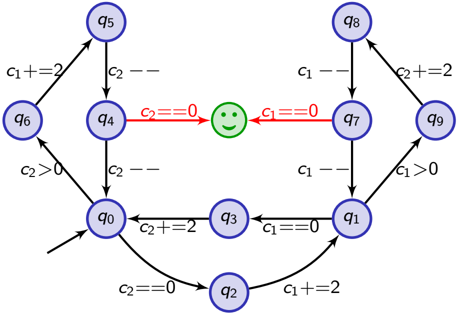
- The value problem asks, given $\mathcal{G}$ and a threshold $\bowtie c$, whether $\text{optcost}_{\mathcal{G}} \bowtie c$?
- The existence problem asks, given $\mathcal{G}$ and a threshold $\bowtie c$, whether there exists a winning strategy in $\mathcal{G}$ such that $\text{cost}(\sigma) \bowtie c$?

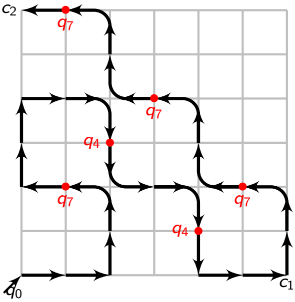
Note: These problems are distinct...

The value of the game is 3, but no strategy has cost 3.

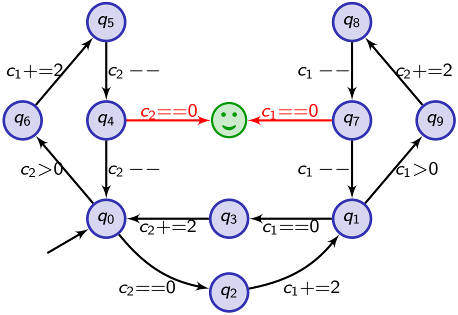


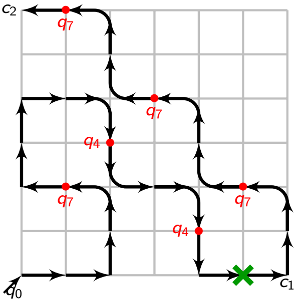
The value of the game is 3, but no strategy has cost 3.





The value of the game is 3, but no strategy has cost 3.





- Weighted timed automata

In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.

- Weighted timed automata

~~In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.~~
The value problem is PSPACE-complete in weighted timed automata.
Almost-optimal winning schedules can be computed.

- Weighted timed automata

~~In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.~~
The value problem is PSPACE-complete in weighted timed automata.
Almost-optimal winning schedules can be computed.

- Weighted timed games

Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.

- Weighted timed automata

~~In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.~~
The value problem is PSPACE-complete in weighted timed automata.
Almost-optimal winning schedules can be computed.

- Weighted timed games

~~Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.~~
The value problem is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.

- Weighted timed automata

~~In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.~~
The value problem is PSPACE-complete in weighted timed automata.
Almost-optimal winning schedules can be computed.

- Weighted timed games

~~Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.~~
The value problem is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.

There is a symbolic algorithm to solve weighted timed games with a strongly non-Zeno cost.

- Weighted timed automata

~~In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.~~
The value problem is PSPACE-complete in weighted timed automata. Almost-optimal winning schedules can be computed.

- Weighted timed games

~~Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.~~
The value problem is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.

~~There is a symbolic algorithm to solve weighted timed games with a strongly non-Zeno cost.~~
The value problem can be decided in EXPTIME in weighted timed games with a strongly non-Zeno cost. Almost-optimal winning strategies can be computed.

- Weighted timed automata

~~In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.~~
The value problem is PSPACE-complete in weighted timed automata. Almost-optimal winning schedules can be computed.

- Weighted timed games

~~Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.~~
The value problem is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.

~~There is a symbolic algorithm to solve weighted timed games with a strongly non-Zeno cost.~~
The value problem can be decided in EXPTIME in weighted timed games with a strongly non-Zeno cost. Almost-optimal winning strategies can be computed.

In weighted timed games, the optimal cost cannot be computed, as soon as games have three clocks or more.

- Weighted timed automata

~~In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.~~
The value problem is PSPACE-complete in weighted timed automata.
Almost-optimal winning schedules can be computed.

- Weighted timed games

~~Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.~~
The value problem is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.

~~There is a symbolic algorithm to solve weighted timed games with a strongly non-Zeno cost.~~
The value problem can be decided in EXPTIME in weighted timed games with a strongly non-Zeno cost. Almost-optimal winning strategies can be computed.

~~In weighted timed games, the optimal cost cannot be computed, as soon as games have three clocks or more.~~
The existence problem is undecidable in weighted timed games.

Outline of the rest of the talk

1. Show that the value problem is undecidable in weighted timed games

Outline of the rest of the talk

1. Show that the value problem is undecidable in weighted timed games
   - $\rightsquigarrow$ This is intellectually satisfactory to not have this discrepancy in the set of results

Outline of the rest of the talk

1. Show that the value problem is undecidable in weighted timed games
   - $\leadsto$ This is intellectually satisfactory to not have this discrepancy in the set of results
   - $\leadsto$ A first proof based on a diagonal construction (originally proposed in the context of quantitative temporal logics [BMM14])

[BMM14] Bouyer, Markey, Matteplackel. Averaging in LTL *(CONCUR'14)*.

Outline of the rest of the talk

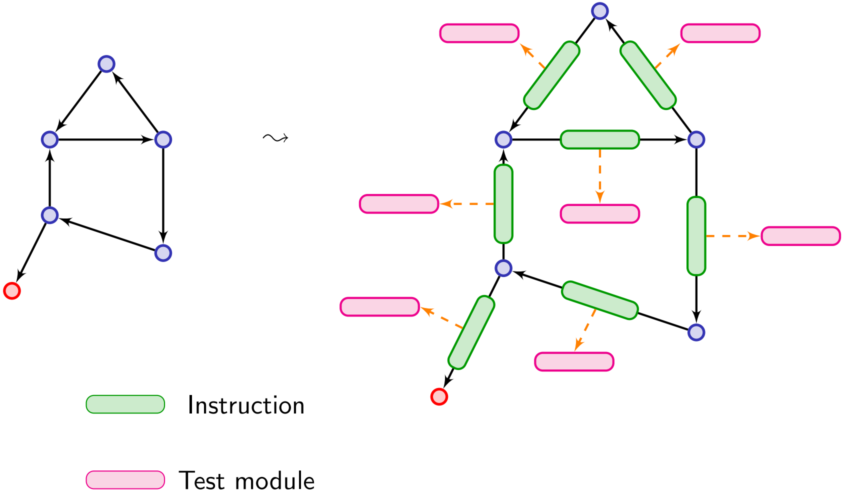
1. Show that the value problem is undecidable in weighted timed games
   - $\leadsto$ This is intellectually satisfactory to not have this discrepancy in the set of results
   - $\leadsto$ A first proof based on a diagonal construction (originally proposed in the context of quantitative temporal logics [BMM14])
   - $\leadsto$ A second direct proof

Outline of the rest of the talk

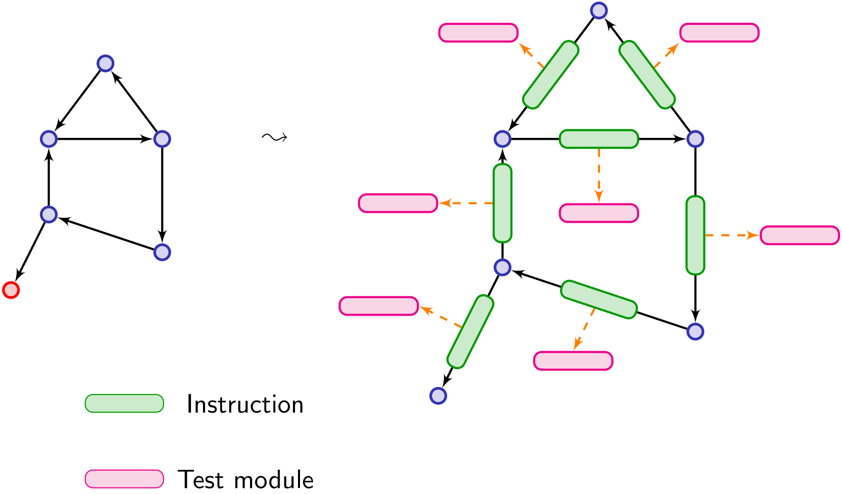
1. Show that the value problem is undecidable in weighted timed games
   - $\rightsquigarrow$ This is intellectually satisfactory to not have this discrepancy in the set of results
   - $\rightsquigarrow$ A first proof based on a diagonal construction (originally proposed in the context of quantitative temporal logics [BMM14])
   - $\rightsquigarrow$ A second direct proof

2. Propose an approximation algorithm for a large class of weighted timed games (that comprises the class of games used for proving the above undecidability)
   - Almost-optimality in practice should be sufficient
   - Even when we know how to compute the value, we are only able to synthesize almost-optimal strategies...

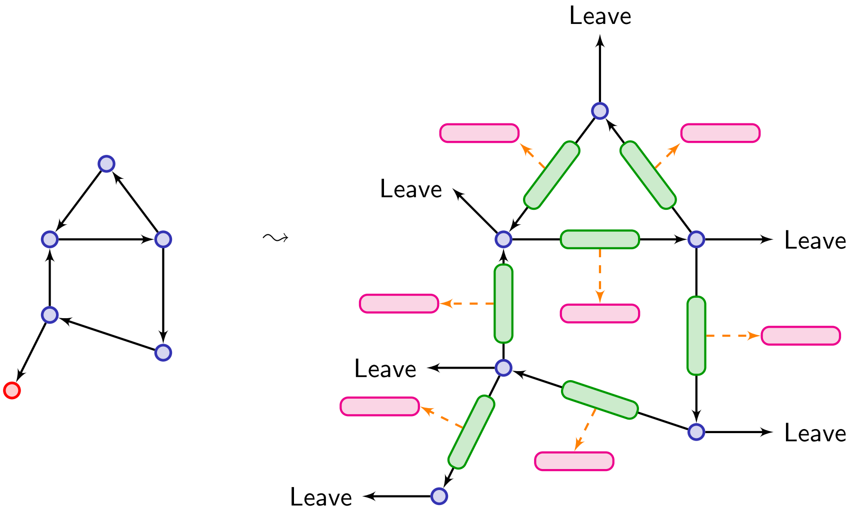
A snapshot on the undecidability proof



A snapshot on the undecidability proof



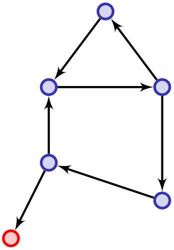
A snapshot on the undecidability proof



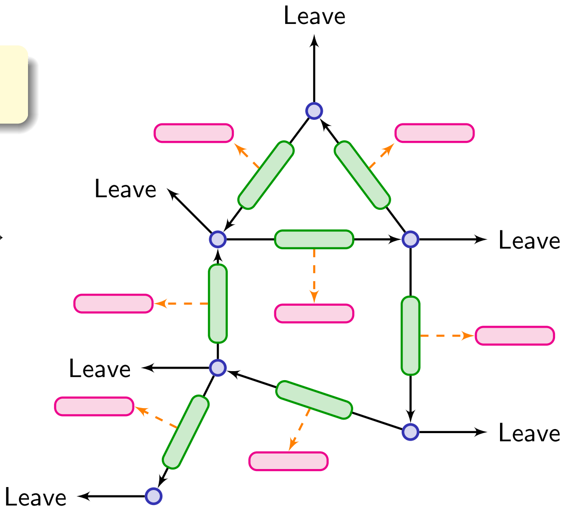
Leave with cost $3 + 1/2^n$ (n: length of the path)

A snapshot on the undecidability proof

$\mathcal{M}$ does not halt iff the value of $\mathcal{G}_{\mathcal{M}}$ is 3



$\rightsquigarrow$



Leave with cost $3 + 1/2^n$ (n: length of the path)

Theorem [BJM15]

The value problem is undecidable in weighted timed games (with four clocks or more).

- Remark on the reduction:
  - Cost 0 within the core of the game
  - The rest of the game is acyclic

[BJM15] Bouyer, Jaziri, Markey. On the value problem in weighted timed games.

Optimal cost is computable...

... when cost is strongly non-zeno. [AM04,BCFL04]

That is, there exists $\kappa > 0$ such that for every region cycle C, for every real run ϱ read on C,

$$\mathsf{cost}(\varrho) \geq \kappa$$

Optimal cost is not computable...

... when cost is almost-strongly non-zeno. [BJM15]

That is, there exists $\kappa > 0$ such that for every region cycle C, for every real run ϱ read on C,

$$\mathsf{cost}(\varrho) \geq \kappa \quad \text{or} \quad \mathsf{cost}(\varrho) = 0$$

Note: In both cases, we can assume $\kappa = 1$.

[BJM15] Bouyer, Jaziri, Markey. On the value problem in weighted timed games.

Optimal cost is computable...

... when cost is strongly non-zeno. [AM04,BCFL04]

That is, there exists $\kappa > 0$ such that for every region cycle C, for every real run ϱ read on C,

$$\mathsf{cost}(\varrho) \geq \kappa$$

Optimal cost is not computable... but is approximable!

... when cost is almost-strongly non-zeno. [BJM15]

That is, there exists $\kappa > 0$ such that for every region cycle C, for every real run ϱ read on C,

$$\mathsf{cost}(\varrho) \geq \kappa \quad \text{or} \quad \mathsf{cost}(\varrho) = 0$$

Note: In both cases, we can assume $\kappa = 1$.

[BJM15] Bouyer, Jaziri, Markey. On the value problem in weighted timed games.

Approximation of the optimal cost

Theorem

Let $\mathcal{G}$ be a weighted timed game, in which the cost is almost-strongly non-zeno. For every $\epsilon > 0$, one can compute:

- two values v_ϵ^- and v_ϵ^+ such that

$$|v_\epsilon^+ - v_\epsilon^-| < \epsilon \quad \text{and} \quad v_\epsilon^- \leq \mathsf{optcost}_{\mathcal{G}} \leq v_\epsilon^+$$

Approximation of the optimal cost

Theorem

Let $\mathcal{G}$ be a weighted timed game, in which the cost is almost-strongly non-zeno. For every $\epsilon > 0$, one can compute:

- two values v_ϵ^- and v_ϵ^+ such that

$$|v_\epsilon^+ - v_\epsilon^-| < \epsilon \quad \text{and} \quad v_\epsilon^- \leq \mathsf{optcost}_\mathcal{G} \leq v_\epsilon^+$$

- one strategy σ_ϵ such that

$$\mathsf{optcost}_\mathcal{G} \leq \mathsf{cost}(\sigma_\epsilon) \leq \mathsf{optcost}_\mathcal{G} + \epsilon$$

  [it is an ϵ-optimal winning strategy]

Approximation of the optimal cost

Theorem

Let $\mathcal{G}$ be a weighted timed game, in which the cost is almost-strongly non-zeno. For every $\epsilon > 0$, one can compute:

- two values v_ϵ^- and v_ϵ^+ such that

$$|v_\epsilon^+ - v_\epsilon^-| < \epsilon \quad \text{and} \quad v_\epsilon^- \leq \mathsf{optcost}_{\mathcal{G}} \leq v_\epsilon^+$$

- one strategy σ_ϵ such that

$$\mathsf{optcost}_{\mathcal{G}} \leq \mathsf{cost}(\sigma_\epsilon) \leq \mathsf{optcost}_{\mathcal{G}} + \epsilon$$

[it is an ϵ-optimal winning strategy]

Skip approximation scheme

Approximation of the optimal cost

Theorem

Let $\mathcal{G}$ be a weighted timed game, in which the cost is almost-strongly non-zeno. For every $\epsilon > 0$, one can compute:

- two values v_ϵ^- and v_ϵ^+ such that

$$|v_\epsilon^+ - v_\epsilon^-| < \epsilon \quad \text{and} \quad v_\epsilon^- \leq \mathsf{optcost}_\mathcal{G} \leq v_\epsilon^+$$

- one strategy σ_ϵ such that

$$\mathsf{optcost}_\mathcal{G} \leq \mathsf{cost}(\sigma_\epsilon) \leq \mathsf{optcost}_\mathcal{G} + \epsilon$$

  [it is an ϵ-optimal winning strategy]

- Standard technics: unfold the game to get more precision, and compute two adjacency sequences

Approximation of the optimal cost

Theorem

Let $\mathcal{G}$ be a weighted timed game, in which the cost is almost-strongly non-zeno. For every $\epsilon > 0$, one can compute:

- two values v_ϵ^- and v_ϵ^+ such that

$$|v_\epsilon^+ - v_\epsilon^-| < \epsilon \quad \text{and} \quad v_\epsilon^- \leq \mathrm{optcost}_{\mathcal{G}} \leq v_\epsilon^+$$

- one strategy σ_ϵ such that

$$\mathrm{optcost}_{\mathcal{G}} \leq \mathrm{cost}(\sigma_\epsilon) \leq \mathrm{optcost}_{\mathcal{G}} + \epsilon$$

  [it is an ϵ-optimal winning strategy]

- Standard technics: unfold the game to get more precision, and compute two adjacency sequences

$\leadsto$ This is not possible here
There might be runs with prefixes of arbitrary length and cost 0 (e.g. the game of the undecidability proof)

Idea for approximation

Idea

Only partially unfold the game:

- Keep components with cost 0 untouched – we call it the kernel
- Unfold the rest of the game

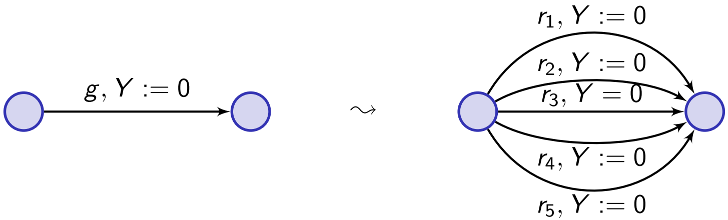
Idea for approximation

Idea

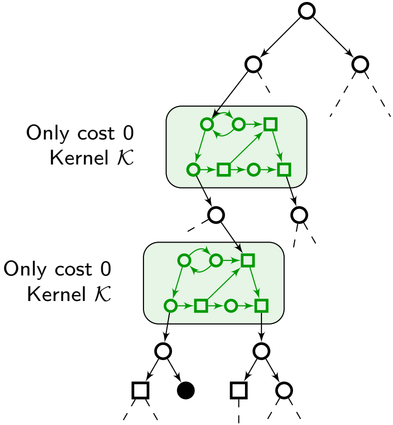
Only partially unfold the game:

- Keep components with cost 0 untouched – we call it the kernel
- Unfold the rest of the game

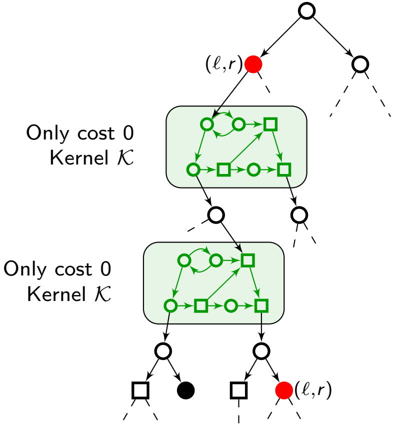
First: split the game along regions!



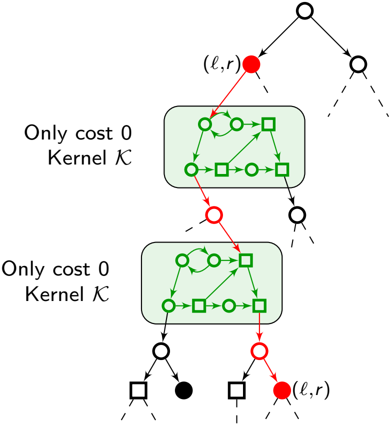
Semi-unfolding



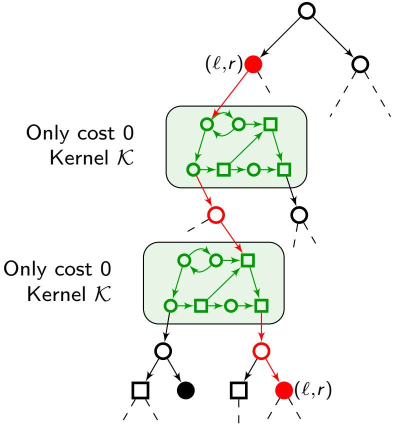
Semi-unfolding



Semi-unfolding

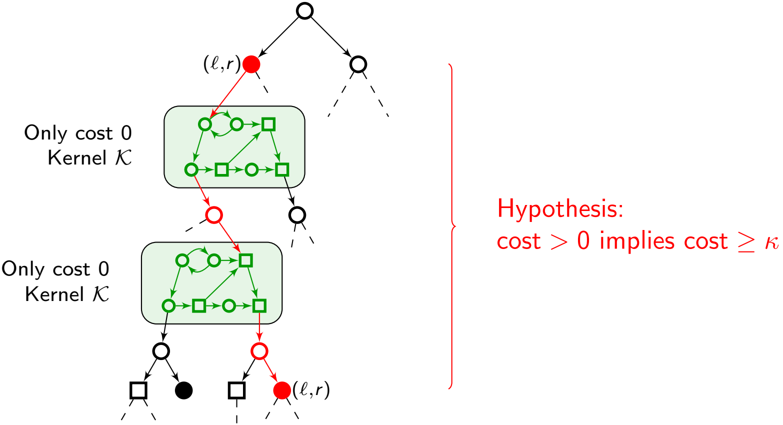


Semi-unfolding



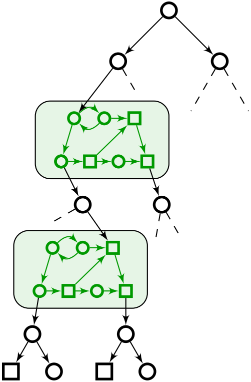
Hypothesis:
cost > 0 implies cost $\geq \kappa$

Semi-unfolding

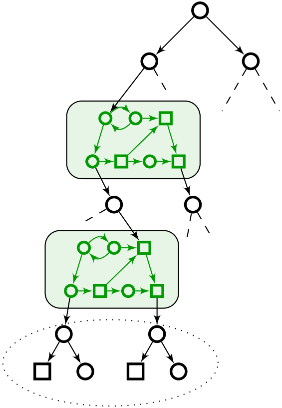


Conclusion: we can stop unfolding the game after N steps
(e.g. $N = (M+2) \cdot |\mathcal{R}(\mathcal{A})|$, where M is a pre-computed bound on $\text{optcost}_{\mathcal{G}}$)

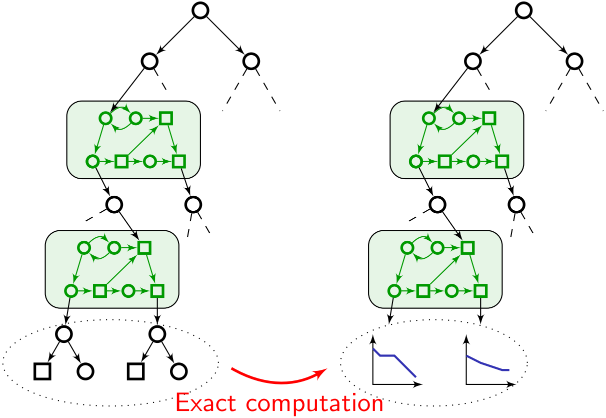
Approximation scheme



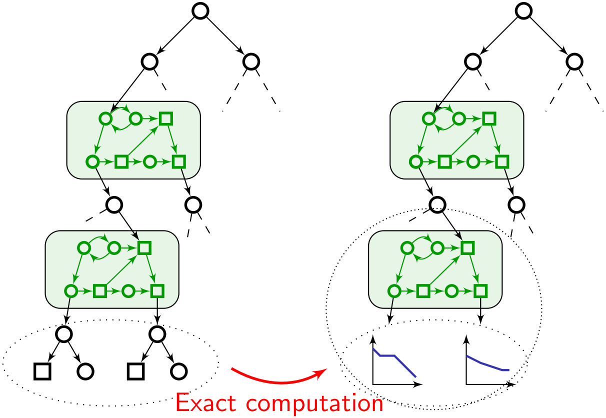
Approximation scheme



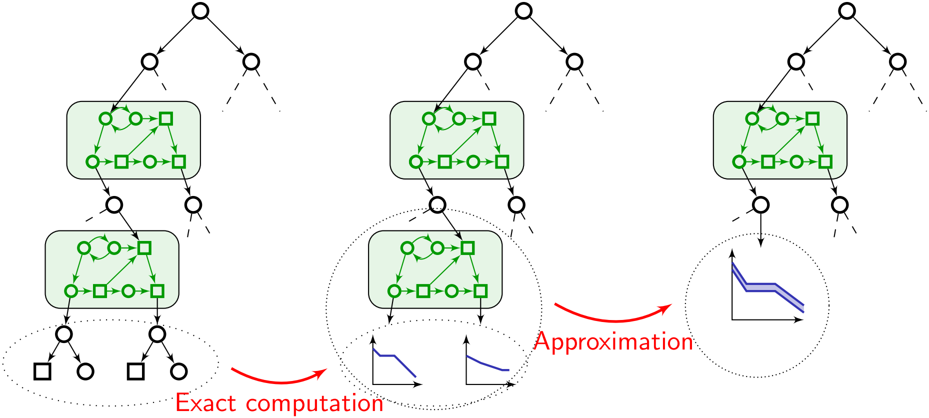
Approximation scheme



Approximation scheme



Approximation scheme



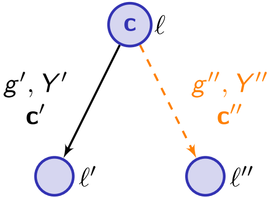
First step: Tree-like parts

$\rightsquigarrow$ Goes back to [LMM02]

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.

First step: Tree-like parts

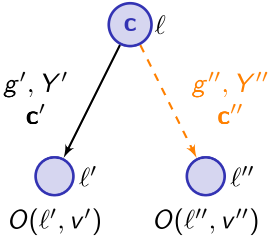
$\rightsquigarrow$ Goes back to [LMM02]



[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.

First step: Tree-like parts

$\rightsquigarrow$ Goes back to [LMM02]

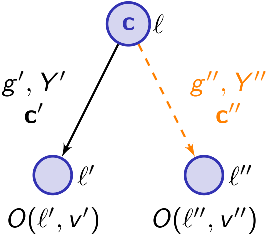


$O(\ell, v) =$

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.

First step: Tree-like parts

$\rightsquigarrow$ Goes back to [LMM02]

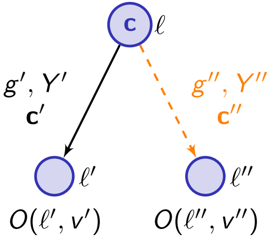


$$O(\ell, v) = \inf_{t' \mid v+t' \models g'}$$

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.

First step: Tree-like parts

$\rightsquigarrow$ Goes back to [LMM02]

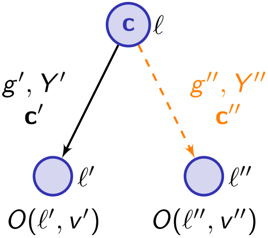


$$O(\ell, v) = \inf_{t' \mid v+t' \models g'} \max(\quad , \quad)$$

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.

First step: Tree-like parts

$\rightsquigarrow$ Goes back to [LMM02]



$$O(\ell, v) = \inf_{t' \mid v+t' \models g'} \max\big((\alpha), \qquad \big)$$

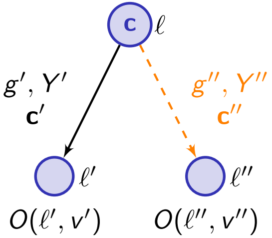
$$(\alpha) = t'\mathbf{c} + \mathbf{c}' + O(\ell', v')$$

$v' = [Y' \leftarrow 0](v+t')$

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.

First step: Tree-like parts

$\rightsquigarrow$ Goes back to [LMM02]



$$O(\ell, v) = \inf_{t' \mid v+t' \models g'} \max\big((\alpha), (\beta)\big)$$

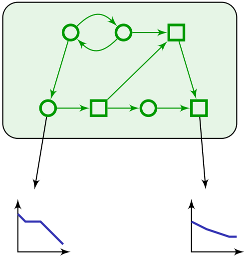
$$(\alpha) = t'\mathbf{c} + \mathbf{c}' + O(\ell', v')$$

$$(\beta) = \sup_{t'' \leq t' \mid v+t'' \models g''} t''\mathbf{c} + \mathbf{c}'' + O(\ell'', v'')$$

$$v' = [Y' \leftarrow 0](v+t')$$
$$v'' = [Y'' \leftarrow 0](v+t'')$$

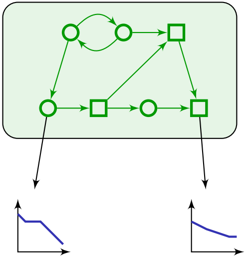
[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.

Second step: Kernels



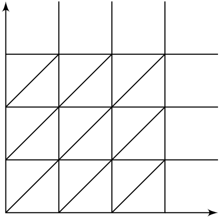
Output cost functions f

Second step: Kernels

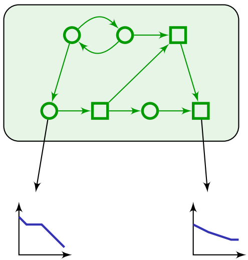


Output cost functions f

1. Refine the regions such that f differs of at most ϵ within a small region

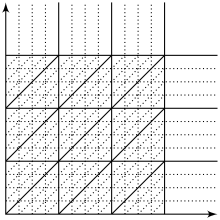


Second step: Kernels

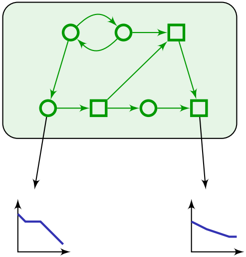


Output cost functions f

1. Refine the regions such that f differs of at most ϵ within a small region

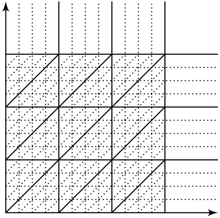


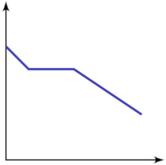
Second step: Kernels



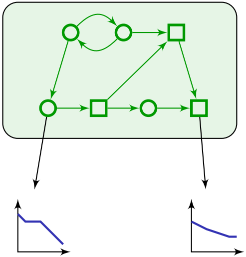
Output cost functions f

1. Refine the regions such that f differs of at most ϵ within a small region



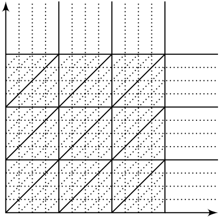


Second step: Kernels

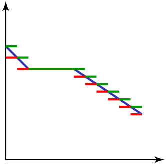


Output cost functions f

1. Refine the regions such that f differs of at most ϵ within a small region

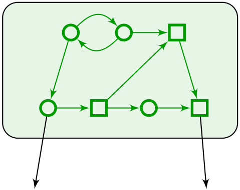


2. Under- and over-approximate by piecewise constant functions f_ϵ^- and f_ϵ^+



Second step: Kernels

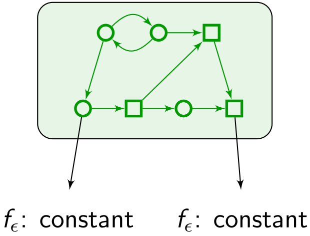
3. Refine/split the kernel along the new small regions and fix f_ϵ^- or f_ϵ^+, write f_ϵ



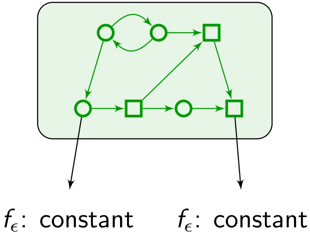
f_ϵ: constant f_ϵ: constant

Second step: Kernels

3. Refine/split the kernel along the new small regions and fix f_ϵ^- or f_ϵ^+, write f_ϵ
4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by f_ϵ)

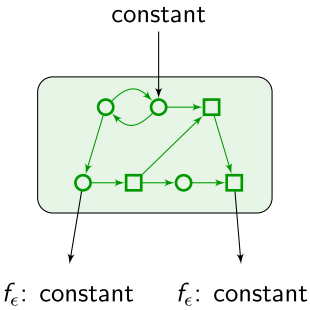


Second step: Kernels



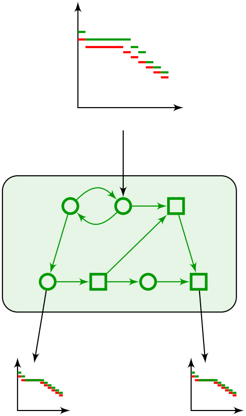
3. Refine/split the kernel along the new small regions and fix f_ϵ^- or f_ϵ^+, write f_ϵ
4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by f_ϵ)
5. Those can be solved using standard technics based on attractors: small regions are sufficient, and the local optimal cost (for output f_ϵ) is constant within a small region

Second step: Kernels



3. Refine/split the kernel along the new small regions and fix f_ϵ^- or f_ϵ^+, write f_ϵ
4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by f_ϵ)
5. Those can be solved using standard technics based on attractors: small regions are sufficient, and the local optimal cost (for output f_ϵ) is constant within a small region

Second step: Kernels



3. Refine/split the kernel along the new small regions and fix f_ϵ^- or f_ϵ^+, write f_ϵ
4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by f_ϵ)
5. Those can be solved using standard technics based on attractors: small regions are sufficient, and the local optimal cost (for output f_ϵ) is constant within a small region

$\rightsquigarrow$ We have computed ϵ-approximations of the optimal cost, which are constant within small regions. Corresponding strategies can be inferred

Outline

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- New insight into the value problem for this model

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- New insight into the value problem for this model

Future work

- Improve the approximation scheme ($2\mathrm{EXP}(|\mathcal{G}|) \cdot \left(1/\epsilon\right)^{|X|}$) and extend it to the whole class of weighted timed games

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- New insight into the value problem for this model

Future work

- Improve the approximation scheme $(2\mathrm{EXP}(|\mathcal{G}|) \cdot \left(1/\epsilon\right)^{|X|})$ and extend it to the whole class of weighted timed games
- Assume stochastic uncertainty

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- New insight into the value problem for this model

Future work

- Improve the approximation scheme ($2\mathrm{EXP}(|\mathcal{G}|) \cdot \left(1/\epsilon\right)^{|X|}$) and extend it to the whole class of weighted timed games
- Assume stochastic uncertainty
- Is the value of any game a rational number?

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- New insight into the value problem for this model

Future work

- Improve the approximation scheme ($2\mathrm{EXP}(|\mathcal{G}|) \cdot \left(1/\epsilon\right)^{|X|}$) and extend it to the whole class of weighted timed games
- Assume stochastic uncertainty
- Is the value of any game a rational number?
- Understand the multiplayer setting