On the value problem in weighted timed games

Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France

Joint work with Samy Jaziri and Nicolas Markey

An example: The task graph scheduling problem

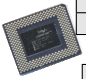
Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

P_1 (fast):

time	
+	2 picoseconds
×	3 picoseconds

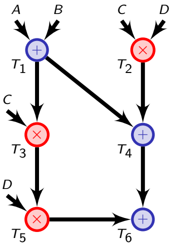
energy	
idle	10 Watt
in use	90 Watts

P_2 (slow):



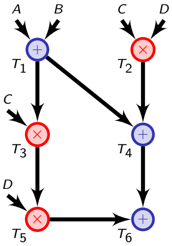
time	
+	5 picoseconds
×	7 picoseconds

energy	
idle	20 Watts
in use	30 Watts



An example: The task graph scheduling problem

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

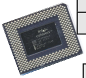


P_1 (fast):

	time
+	2 picoseconds
×	3 picoseconds

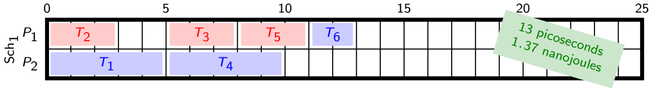
energy	
idle	10 Watt
in use	90 Watts

P_2 (slow):



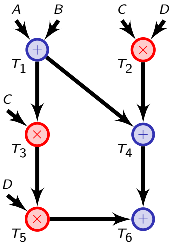
	time
+	5 picoseconds
×	7 picoseconds

energy	
idle	20 Watts
in use	30 Watts



An example: The task graph scheduling problem

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

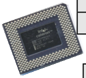


P_1 (fast):

time	
+	2 picoseconds
×	3 picoseconds

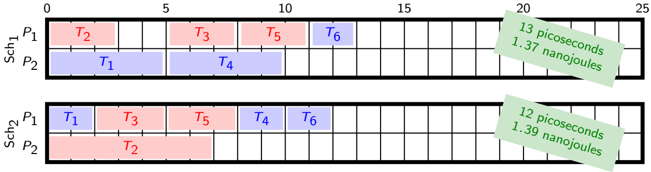
energy	
idle	10 Watt
in use	90 Watts

P_2 (slow):



time	
+	5 picoseconds
×	7 picoseconds

energy	
idle	20 Watts
in use	30 Watts



An example: The task graph scheduling problem

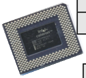
Compute $D\times(C\times(A+B))+(A+B)+(C\times D)$ using two processors:

P_1 (fast):

	time
+	2 picoseconds
×	3 picoseconds

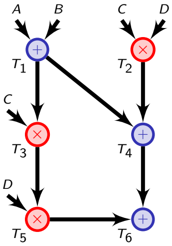
	energy
idle	10 Watt
in use	90 Watts

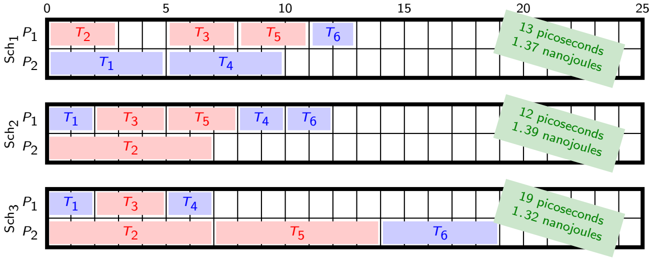
P_2 (slow):



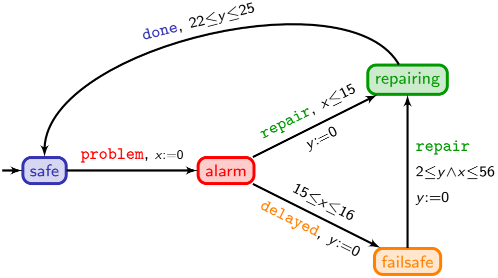
	time
+	5 picoseconds
×	7 picoseconds

	energy
idle	20 Watts
in use	30 Watts

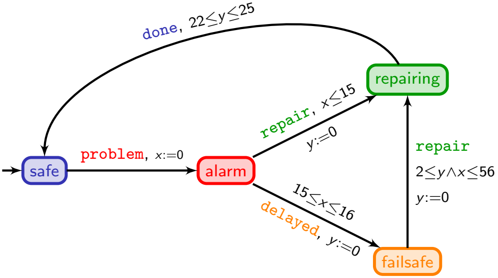




The model of timed automata



The model of timed automata



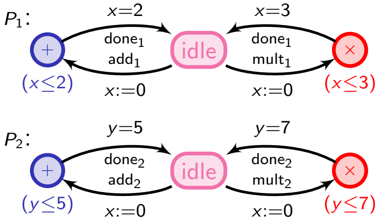
	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	⋯
y	0		23		23		38.6		0	

	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

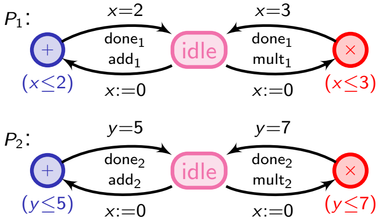
Modelling the task graph scheduling problem

- Processors

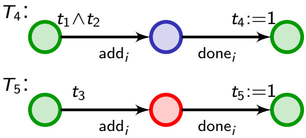


Modelling the task graph scheduling problem

- Processors



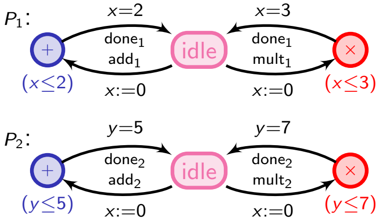
- Tasks



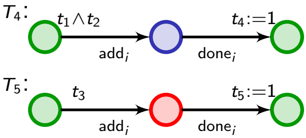
A schedule is a path in the product automaton

Modelling the task graph scheduling problem

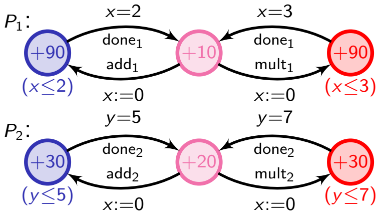
- Processors



- Tasks



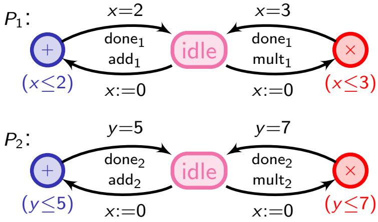
- Modelling energy



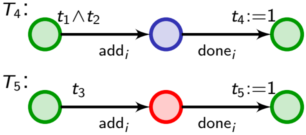
A good schedule is a path in the product automaton with a low cost

Modelling the task graph scheduling problem

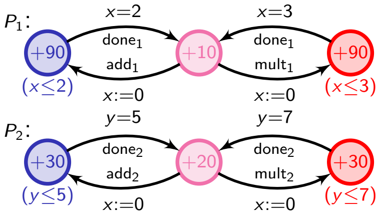
- Processors



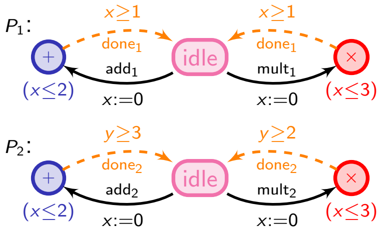
- Tasks



- Modelling energy



- Modelling uncertainty



Modelling the task graph scheduling problem

- Processors

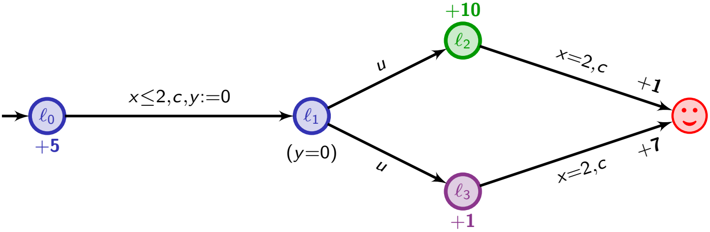
- Tasks

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

- Modelling energy

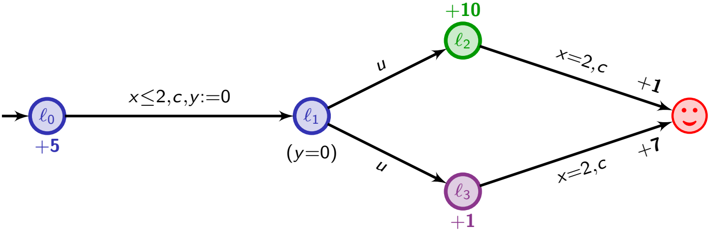
- Modelling uncertainty

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



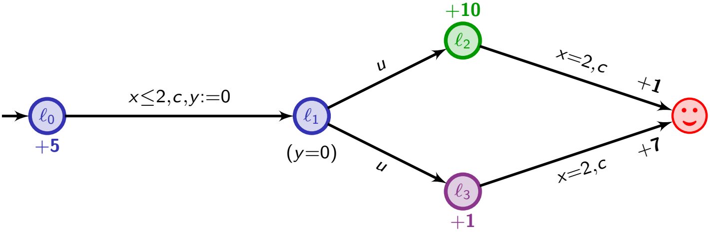
[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

Weighted/priced timed automata



	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		

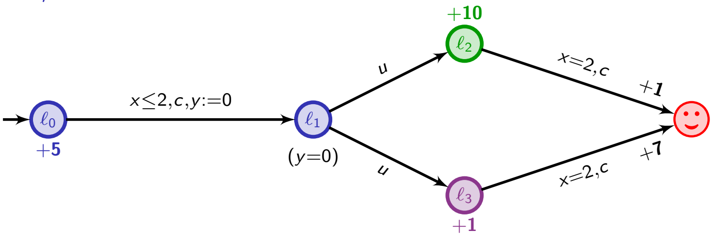
Weighted/priced timed automata



	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 :

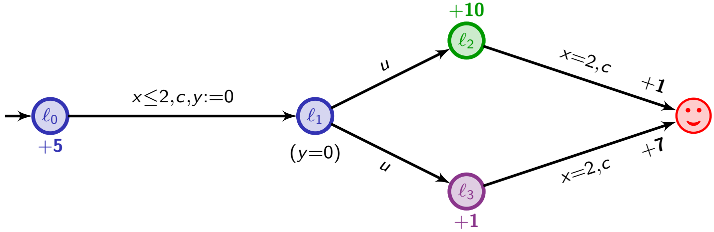
Weighted/priced timed automata



	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

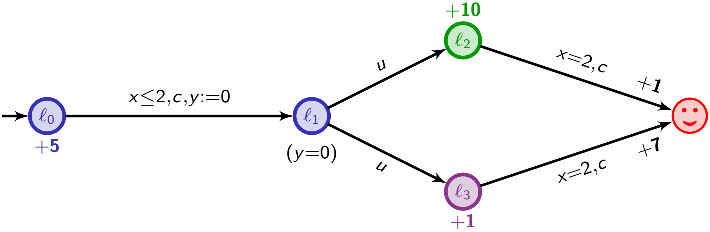
Weighted/priced timed automata



	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

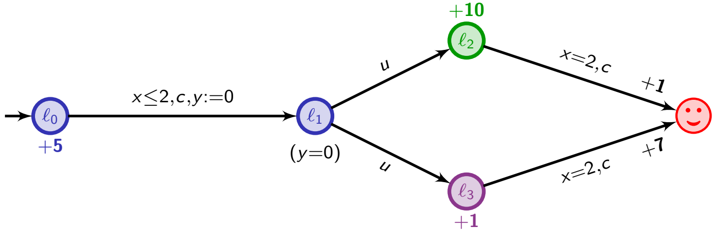
Weighted/priced timed automata



	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

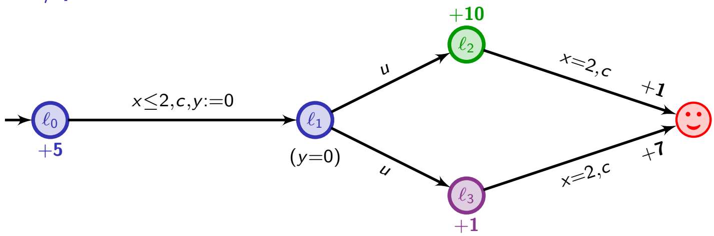
Weighted/priced timed automata



	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		

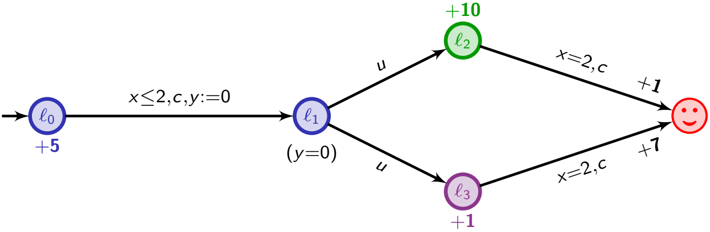
cost : 6.5 + 0 + 0 + 0.7

Weighted/priced timed automata



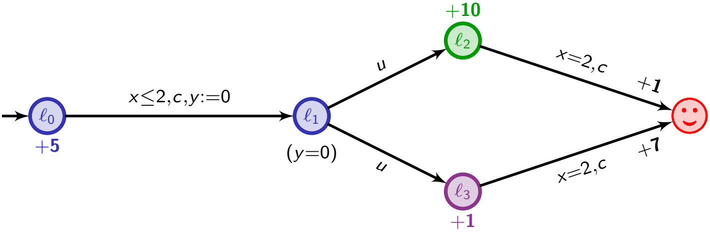
	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		
cost :		6.5	+	0	+	0	+	0.7	+	7	

Weighted/priced timed automata



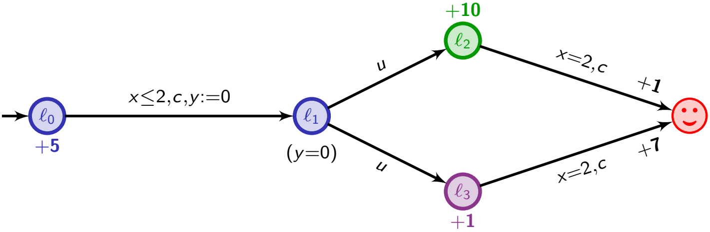
	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺			
x	0		1.3		1.3		1.3		2					
y	0		1.3		0		0		0.7					
cost :		6.5	+	0	+	0	+	0.7	+	7			=	14.2

Weighted/priced timed automata



Question: what is the optimal cost for reaching 🙂 ?

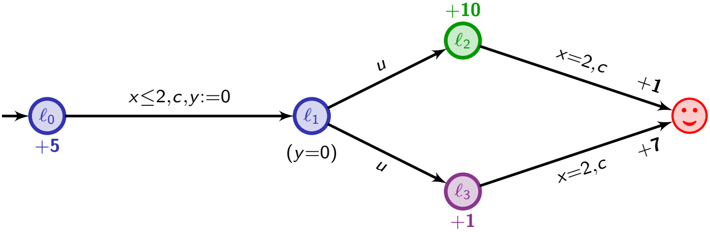
Weighted/priced timed automata



Question: what is the optimal cost for reaching 🙂?

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

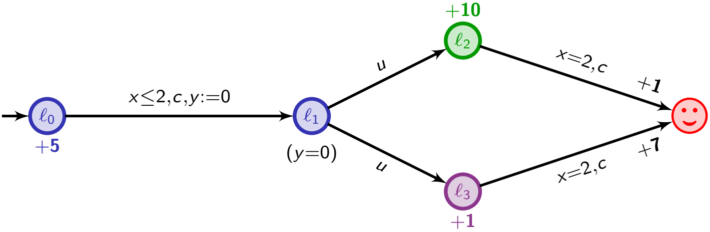
Weighted/priced timed automata



Question: what is the optimal cost for reaching 🙂?

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

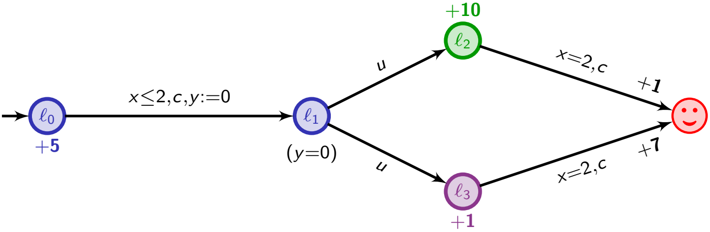
Weighted/priced timed automata



Question: what is the optimal cost for reaching ☺ ?

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

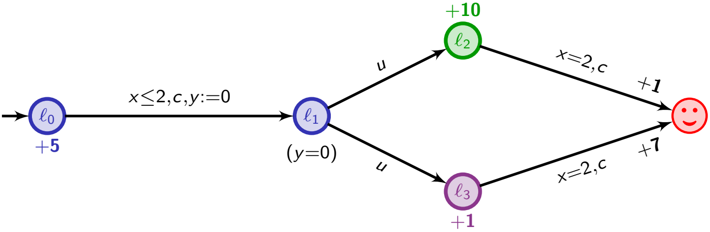
Weighted/priced timed automata



Question: what is the optimal cost for reaching  ?

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

Weighted/priced timed automata

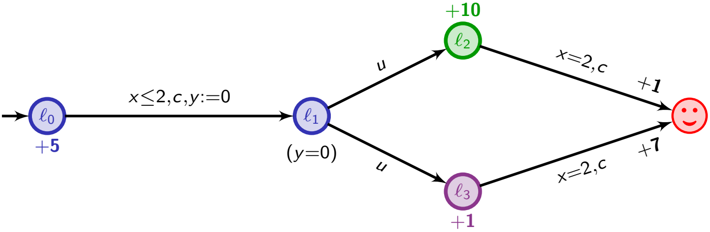


Question: what is the optimal cost for reaching ?

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

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

Weighted/priced timed automata



Question: what is the optimal cost for reaching 🙂?

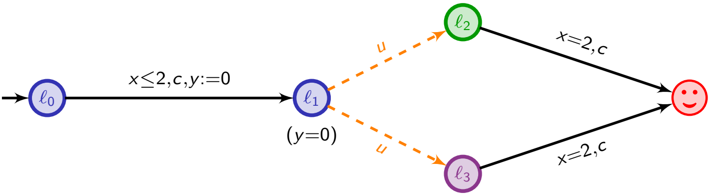
$$\inf_{0 \leq t \leq 2} \min\left(\, 5t + 10(2-t) + 1 \,,\; 5t + (2-t) + 7 \,\right) = 9$$

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

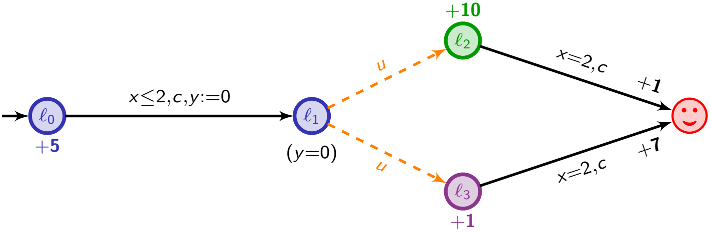
That can be generalized!

[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)*.

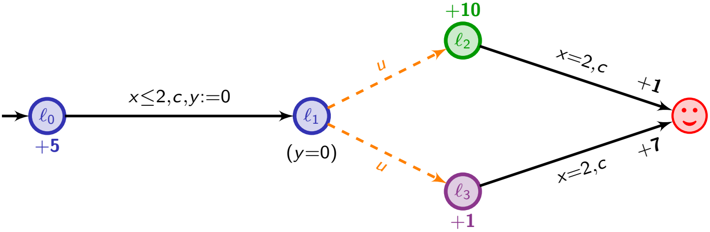
A simple timed game



A simple weighted timed game

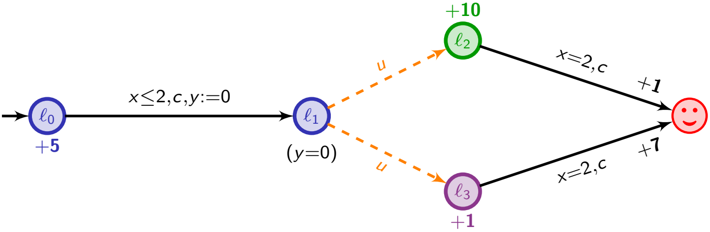


A simple weighted timed game



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

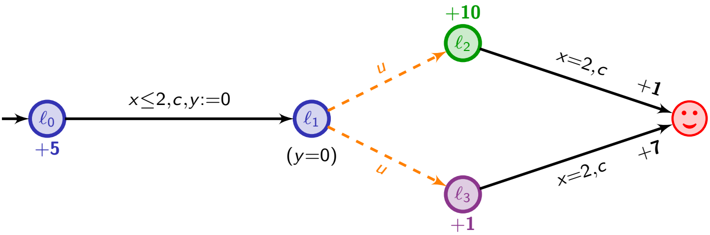
A simple weighted timed game



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

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

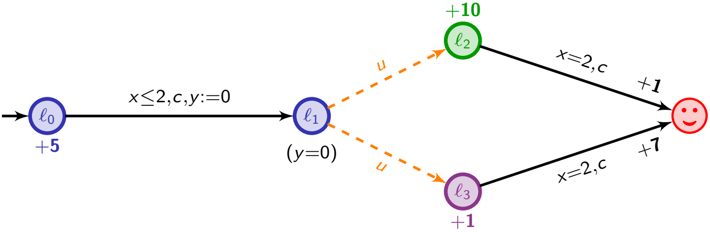
A simple weighted timed game



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

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

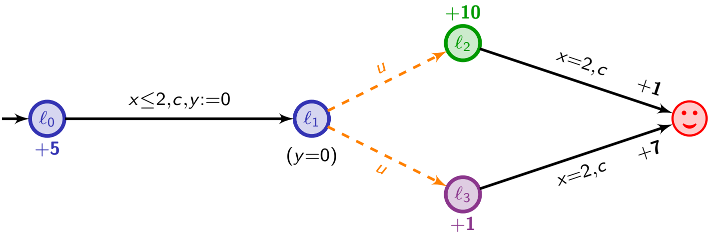
A simple weighted timed game



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

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

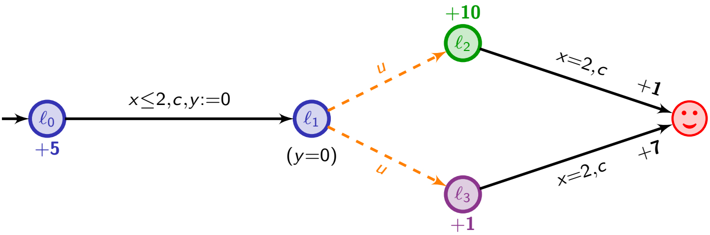
A simple weighted timed game



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 🙂?

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

$\rightsquigarrow$ *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...

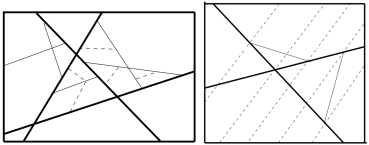
[LMM02,ABM04,BCFL04,BBR05,BBM06,BLMR06,Rut11,HIM13,BGK+14]

[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]

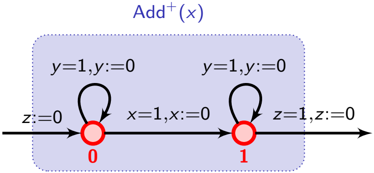
Turn-based optimal timed games are decidable in EXPTIME (resp. PTIME) when automata have a single clock (resp. with two rates). They are PTIME-hard.

Computing the optimal cost: why is that hard?

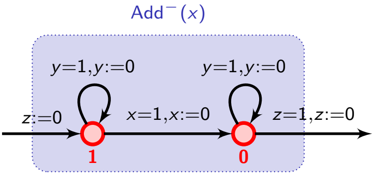
Given two clocks x and y, we can check whether $y = 2x$.

Computing the optimal cost: why is that hard?

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



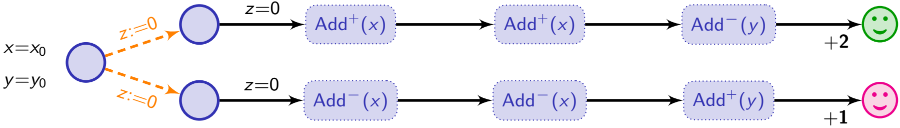
The cost is increased by x_0



The cost is increased by $1 - x_0$

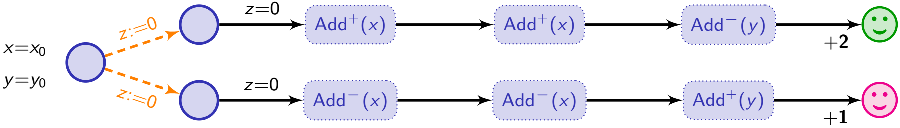
Computing the optimal cost: why is that hard?

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



Computing the optimal cost: why is that hard?

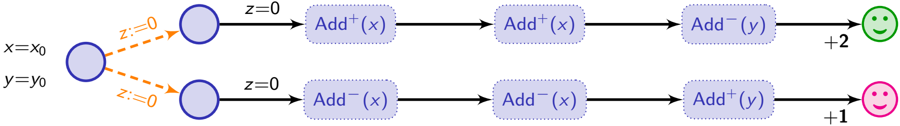
Given two clocks x and y, we can check whether $y = 2x$.



- 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 $y = 2x$.

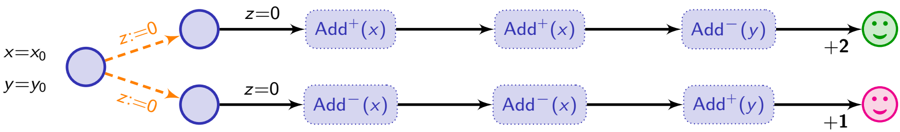


- In , cost $= 2x_0 + (1 - y_0) + 2$

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

Computing the optimal cost: why is that hard?

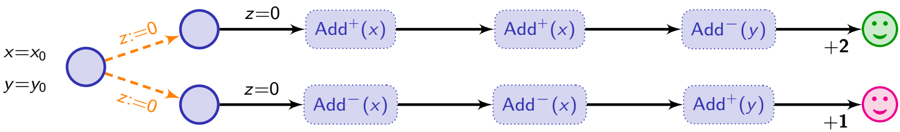
Given two clocks x and y, we can check whether $y = 2x$.



- 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 $y = 2x$.



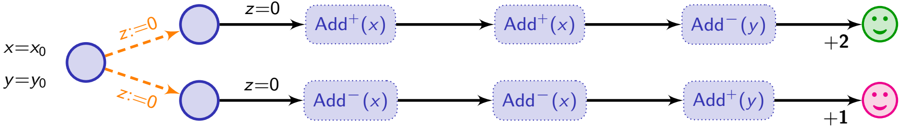
- In , cost $= 2x_0 + (1 - y_0) + 2$

  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?

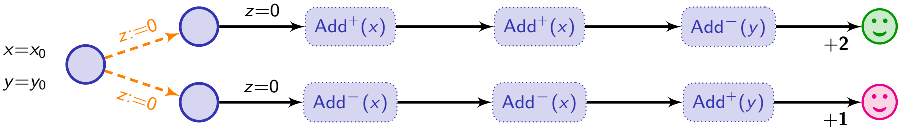
Given two clocks x and y, we can check whether $y = 2x$.



- 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?

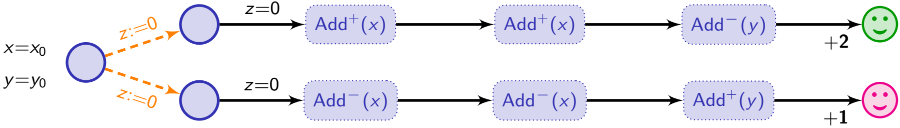
Given two clocks x and y, we can check whether $y = 2x$.



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

$$x = \frac{1}{2^{c_1}} \qquad \text{and} \qquad y = \frac{1}{3^{c_2}}$$

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:

$$x = \frac{1}{2^{c_1}} \qquad \text{and} \qquad y = \frac{1}{3^{c_2}}$$

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?

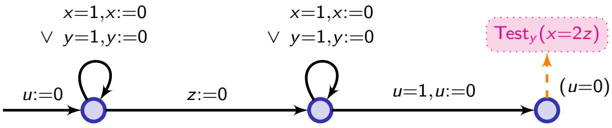
Player 1 will simulate a two-counter machine:

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

$$x = \frac{1}{2^{c_1}} \qquad \text{and} \qquad y = \frac{1}{3^{c_2}}$$

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

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



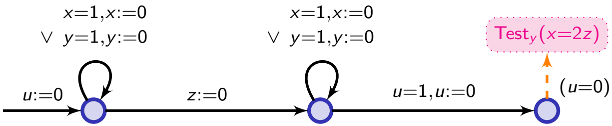
Computing the optimal cost: why is that hard?

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:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{3^{c_2}}$$

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



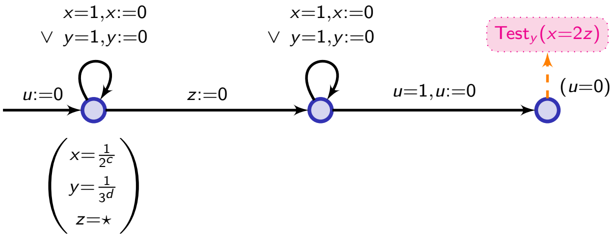
Computing the optimal cost: why is that hard?

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:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{3^{c_2}}$$

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



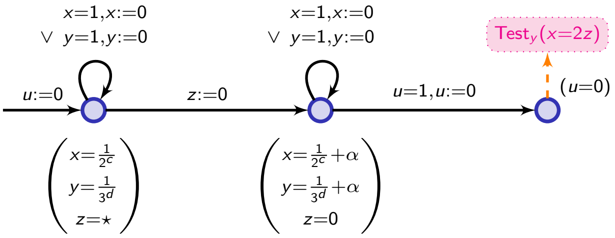
Computing the optimal cost: why is that hard?

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:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{3^{c_2}}$$

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



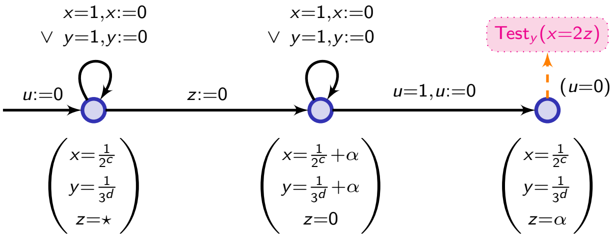
Computing the optimal cost: why is that hard?

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:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{3^{c_2}}$$

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



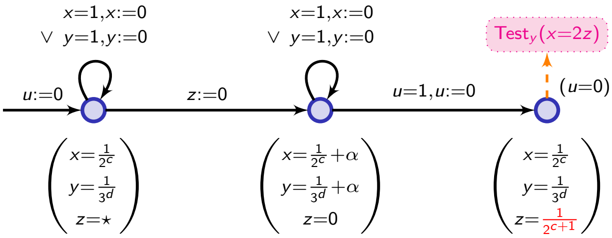
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

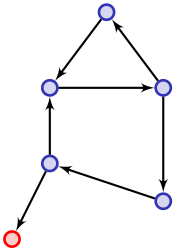
- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \qquad \text{and} \qquad y = \frac{1}{3^{c_2}}$$

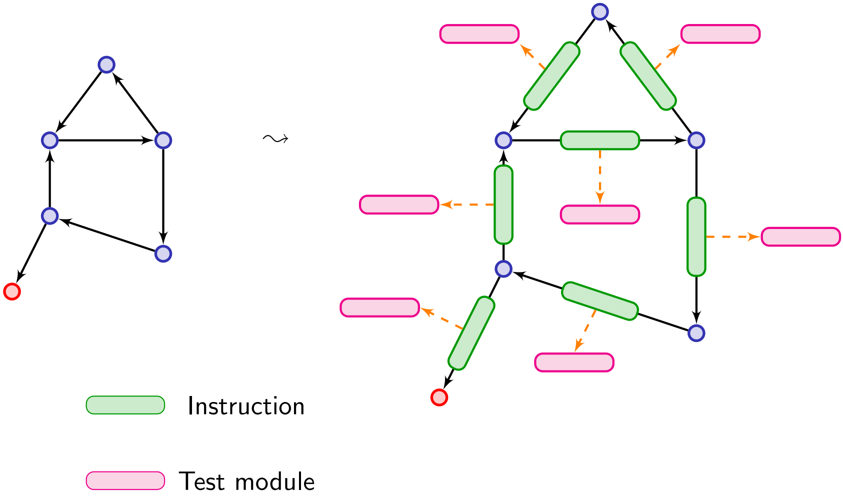
The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.



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:

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

- Optimal cost:

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

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

Two problems of interest

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

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:

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

- Optimal cost:

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

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

Two problems of interest

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

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:

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

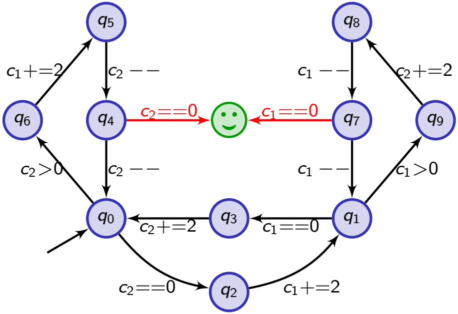
(set it to $+\infty$ if there is no winning strategy)

Two problems of interest

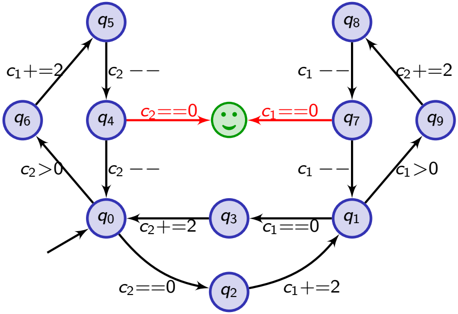
- The value problem asks, given $\mathcal{G}$ and a threshold $\bowtie c$, whether $\text{optcost}_{\mathcal{G}} \bowtie c$?
- The existence problem asks, given $\mathcal{G}$ and a threshold $\bowtie c$, whether there exists a winning strategy in $\mathcal{G}$ such that $\text{cost}(\sigma) \bowtie c$?

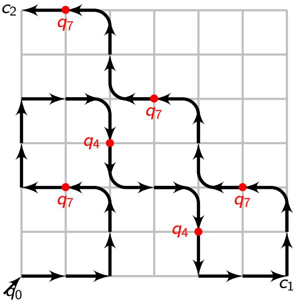
Note: These problems are distinct...

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

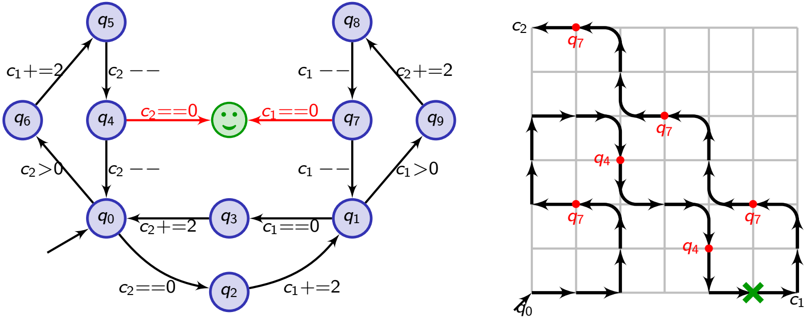


The value of the game is 3, but no strategy has cost 3.





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.

Our recent developments

1. The value problem is undecidable in weighted timed games
   - $\rightsquigarrow$ 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] – see Nicolas Markey's talk)
   - $\rightsquigarrow$ A second direct proof

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

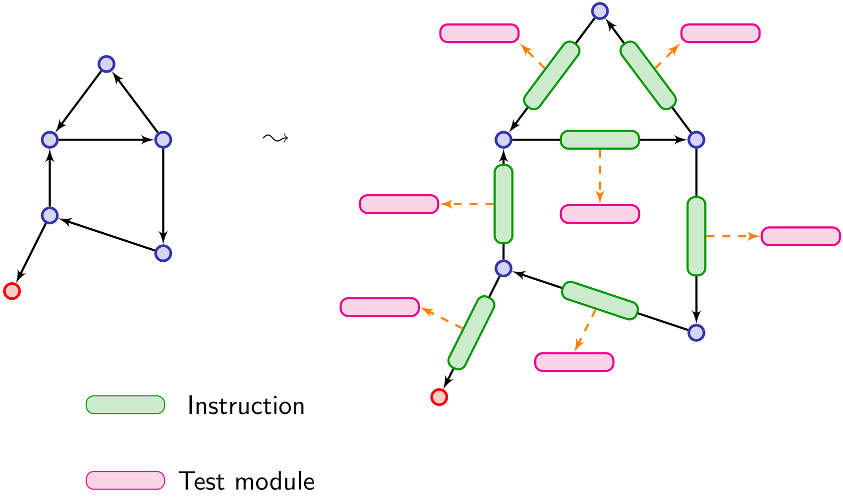
Our recent developments

1. The value problem is undecidable in weighted timed games
   - ⇝ Intellectually satisfactory to not have this discrepancy in the set of results
   - ⇝ A first proof based on a diagonal construction (originally proposed in the context of quantitative temporal logics [BMM14] – see Nicolas Markey's talk)
   - ⇝ A second direct proof

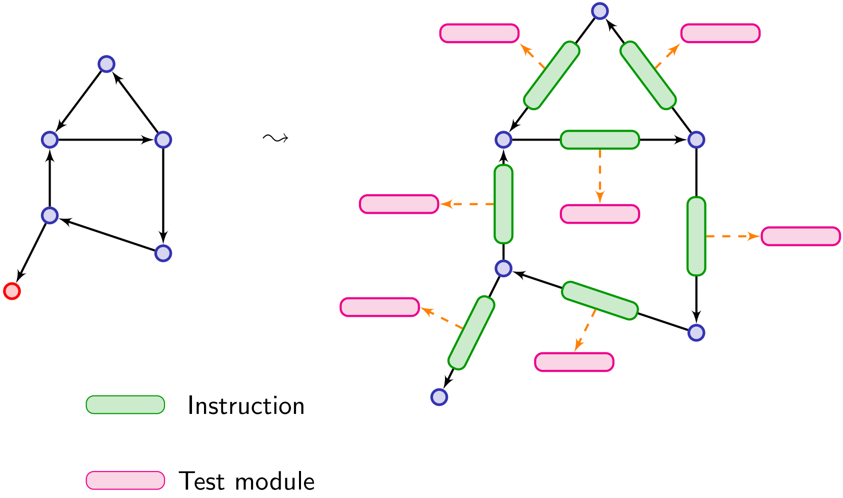
2. An approximation algorithm for a large class of weighted timed games (that comprises the class of games used for proving the above undecidability)
   - Almost-optimality in practice should be sufficient
   - Even when we know how to compute the value, we are only able to synthesize almost-optimal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

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

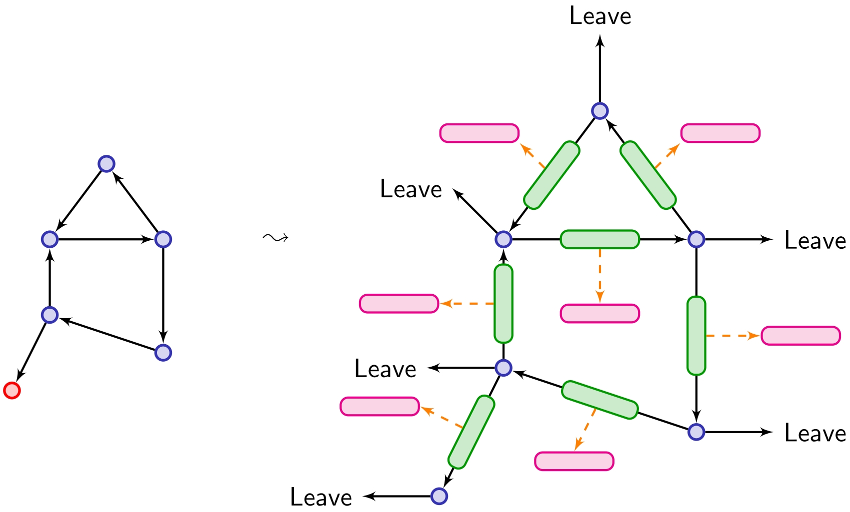
A snapshot on the undecidability proof



A snapshot on the undecidability proof



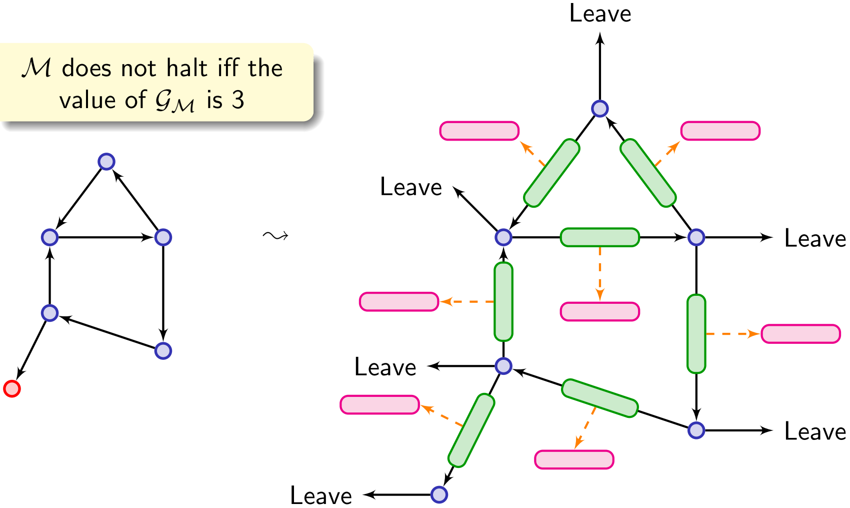
A snapshot on the undecidability proof



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



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

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,

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

Optimal cost is not computable...

... when cost is almost-strongly non-zeno.

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

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

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

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.

- 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

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

- 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

$\leadsto$ 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)

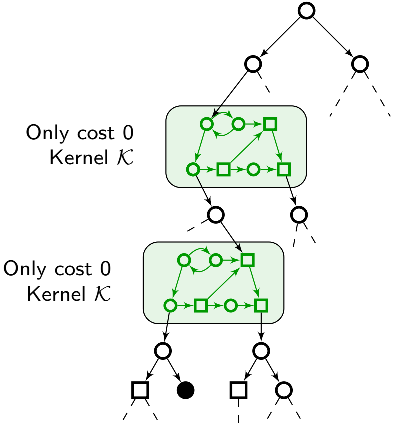
Idea for approximation

Idea

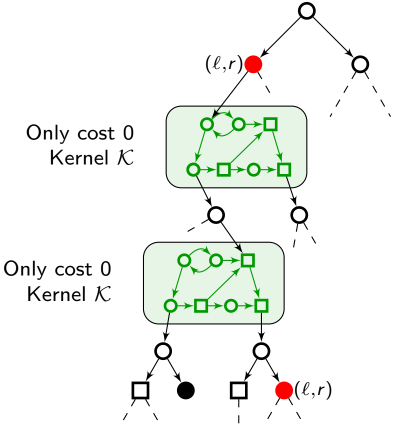
Only partially unfold the game:

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

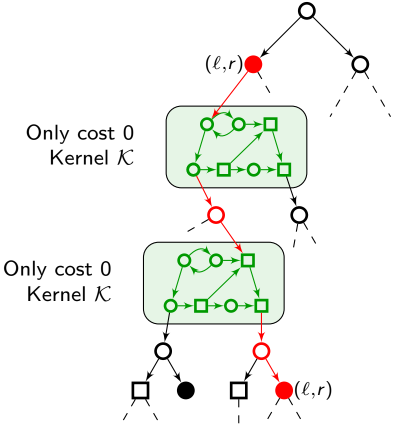
Semi-unfolding



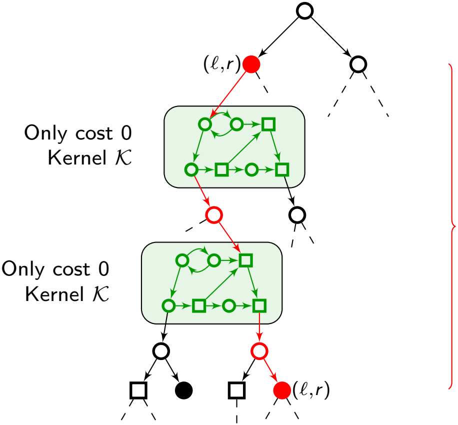
Semi-unfolding



Semi-unfolding

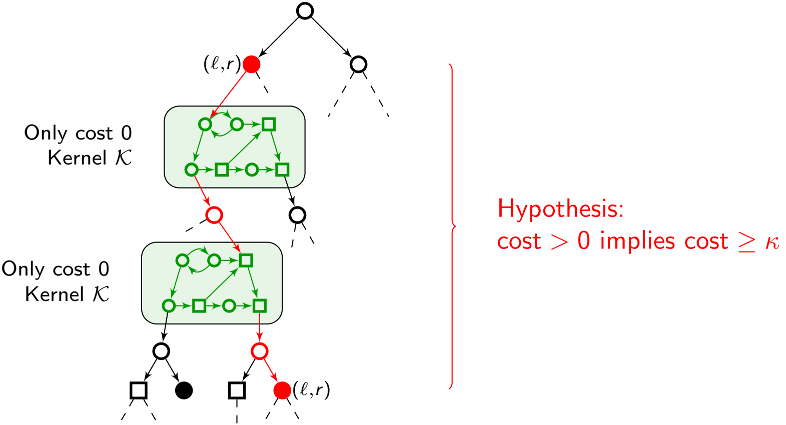


Semi-unfolding



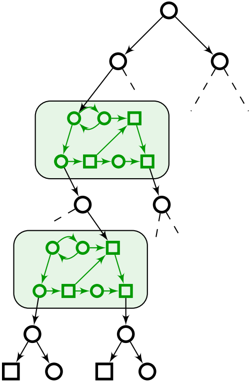
Hypothesis:
cost > 0 implies cost $\geq \kappa$

Semi-unfolding

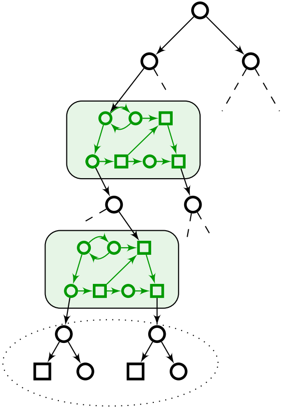


Conclusion: we can stop unfolding the game after N steps
(e.g. $N = (M+2) \cdot |\mathcal{R}(\mathcal{A})|$, where M is a pre-computed bound on $\text{optcost}_{\mathcal{G}}$)

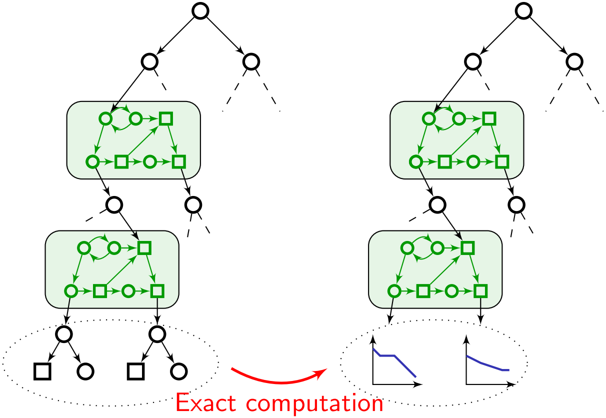
Approximation scheme



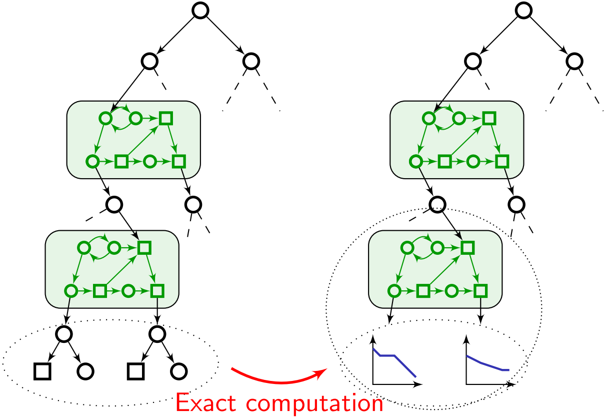
Approximation scheme



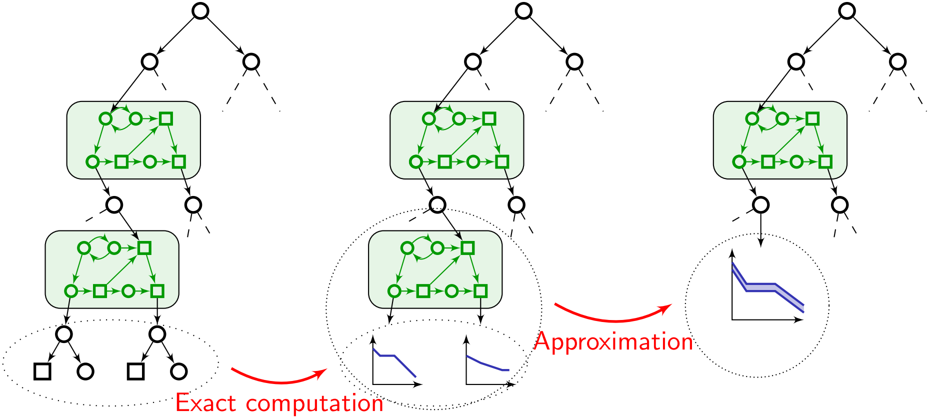
Approximation scheme



Approximation scheme



Approximation scheme



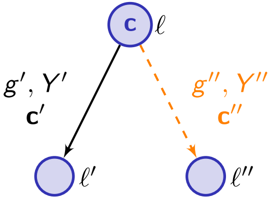
First step: Tree-like parts

$\rightsquigarrow$ 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

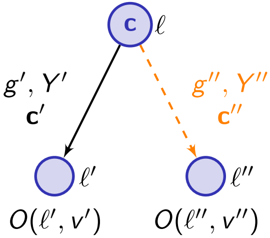
$\rightsquigarrow$ Goes back to [LMM02]



[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]

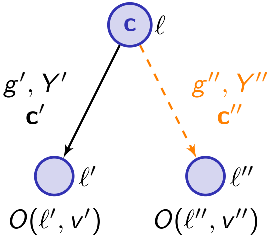


$O(\ell, v) =$

[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]

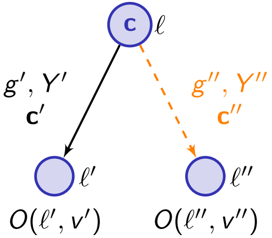


$$O(\ell, v) = \inf_{t' \mid v+t' \models g'}$$

[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]

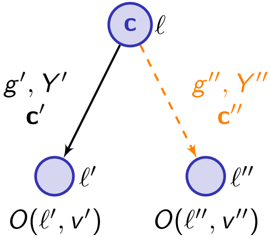


$$O(\ell, v) = \inf_{t' \mid v+t' \models g'} \max(\quad , \quad)$$

[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]



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

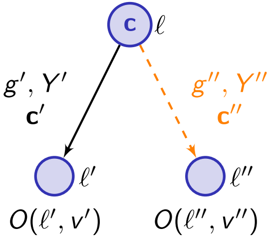
$$(\alpha) = t'\mathbf{c} + \mathbf{c}' + O(\ell', v')$$

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

[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]



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

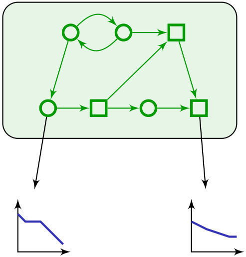
$$(\alpha) = t'\mathbf{c} + \mathbf{c}' + O(\ell', v')$$

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

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

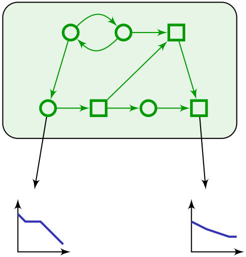
[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.

Second step: Kernels



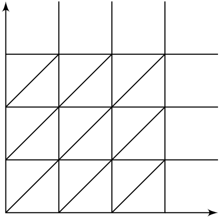
Output cost functions f

Second step: Kernels

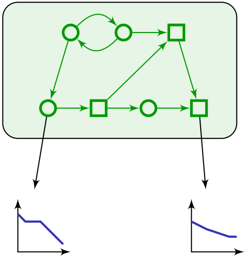


Output cost functions f

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

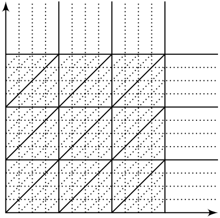


Second step: Kernels



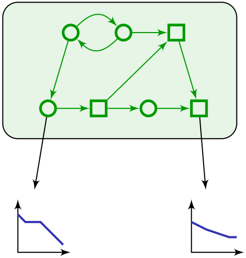
Output cost functions f

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

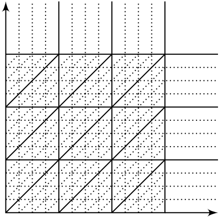


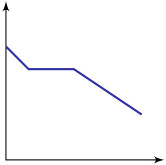
Second step: Kernels

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

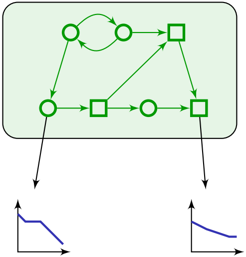


Output cost functions f



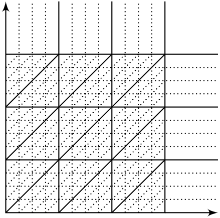


Second step: Kernels

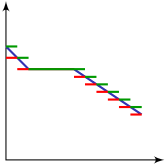


Output cost functions f

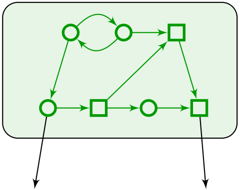
1. Refine the regions such that f differs of at most ϵ within a small region



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

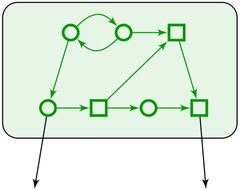


Second step: Kernels



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

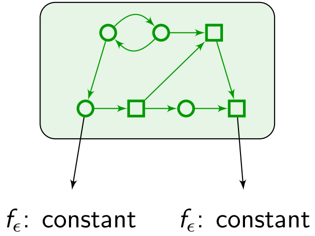
Second step: Kernels



f_ϵ: constant $\quad$ f_ϵ: constant

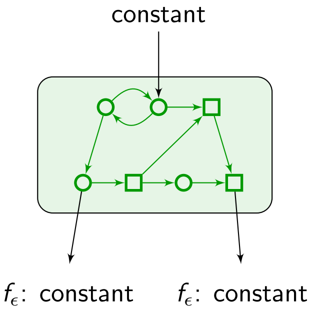
3. Refine/split the kernel along the new small regions and fix f_ϵ^- or f_ϵ^+, write f_ϵ
4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by f_ϵ)

Second step: Kernels



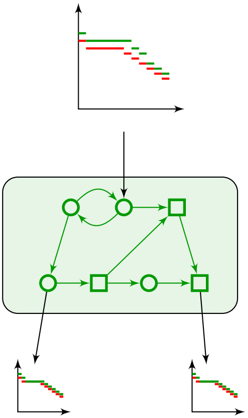
3. Refine/split the kernel along the new small regions and fix f_ϵ^- or f_ϵ^+, write f_ϵ
4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by f_ϵ)
5. Those can be solved using standard technics based on attractors: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

Conclusion

Summary of the talk

- Very quick overview of results concerning the optimal reachability problem in weighted timed games
- Some new insight into the value problem for this model:
  - Undecidability of this problem
  - Approximability of the optimal cost (under some conditions)

Conclusion

Summary of the talk

- Very quick overview of results concerning the optimal reachability problem in weighted timed games
- Some new insight into the value problem for this model:
  - Undecidability of this problem
  - Approximability of the optimal cost (under some conditions)

Future work

- Improve the approximation scheme $(2\mathsf{EXP}(|\mathcal{G}|) \cdot \left(1/\epsilon\right)^{|X|})$
- Extend to the whole class of weighted timed games, or understand why it is not possible
- Assume stochastic uncertainty?