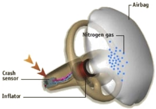
On the optimal reachability problem in weighted timed automata and games

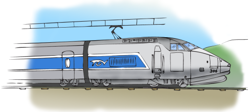
Patricia Bouyer-Decitre

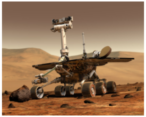
LSV, CNRS & ENS Cachan, France

Time-dependent systems

- We are interested in timed systems

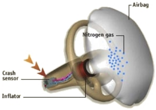


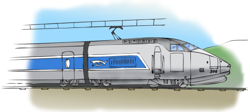


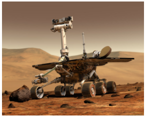


Time-dependent systems

- We are interested in timed systems



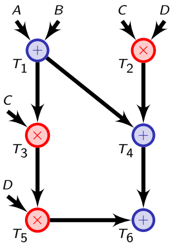




- ... and in their analysis and control

An example: The task graph scheduling problem

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

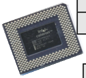


P_1 (fast):

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

energy	
idle	10 Watt
in use	90 Watts

P_2 (slow):



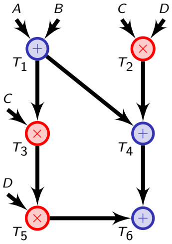
time	
$+$	5 picoseconds
$\times$	7 picoseconds

energy	
idle	20 Watts
in use	30 Watts

[BFLM10] Bouyer, Fahrenberg, Larsen, Markey. Quantitative Analysis of Real-Time Systems using Priced Timed Automata *(Communication of the ACM)*.

An example: The task graph scheduling problem

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

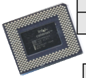


P_1 (fast):

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

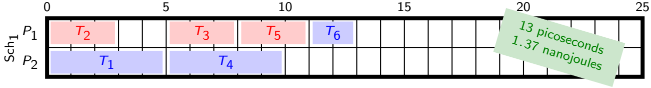
energy	
idle	10 Watt
in use	90 Watts

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 *(Communication of the ACM)*.

An example: The task graph scheduling problem

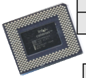
Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

P_1 (fast):

	time
+	2 picoseconds
×	3 picoseconds

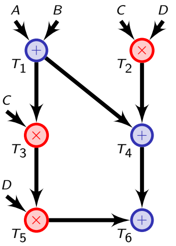
	energy
idle	10 Watt
in use	90 Watts

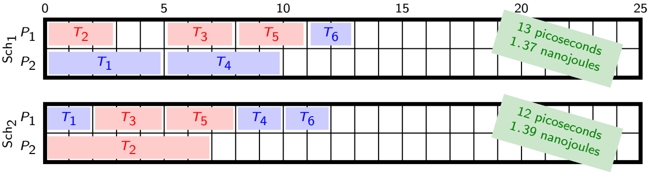
P_2 (slow):



	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 *(Communication of the ACM).*

An example: The task graph scheduling problem

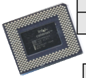
Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

P_1 (fast):

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

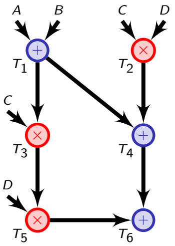
energy	
idle	10 Watt
in use	90 Watts

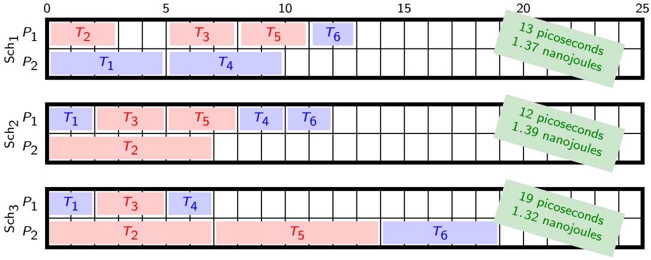
P_2 (slow):



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

energy	
idle	20 Watts
in use	30 Watts

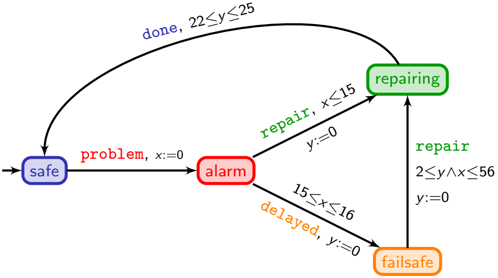




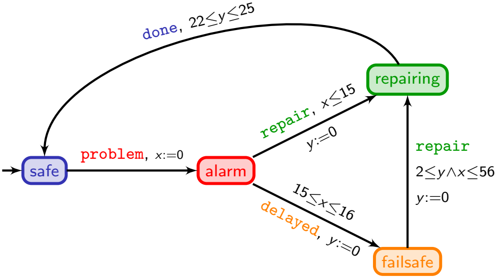
[BFLM10] Bouyer, Fahrenberg, Larsen, Markey. Quantitative Analysis of Real-Time Systems using Priced Timed Automata *(Communication of the ACM).*

Outline

The model of timed automata



The model of timed automata



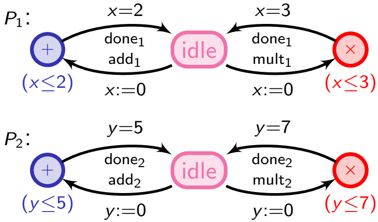
	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	⋯
y	0		23		23		38.6		0	

	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

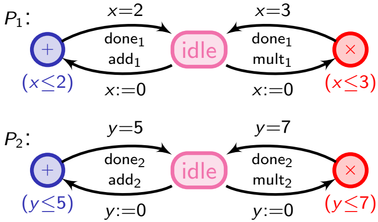
Modelling the task graph scheduling problem

- Processors

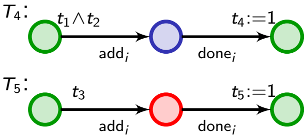


Modelling the task graph scheduling problem

- Processors

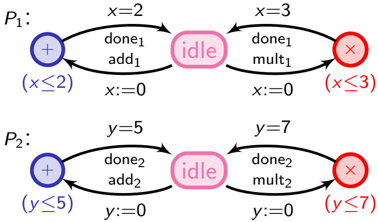


- Tasks

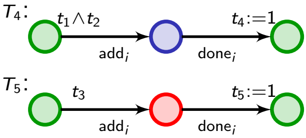


Modelling the task graph scheduling problem

- Processors



- Tasks

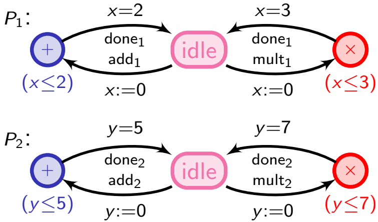


$\rightsquigarrow$ build the synchronized product of all these automata

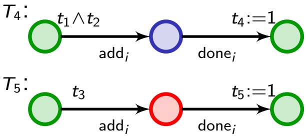
$$(P_1 \parallel P_2) \parallel_s (T_1 \parallel T_2 \parallel \cdots \parallel T_6)$$

Modelling the task graph scheduling problem

- Processors



- Tasks



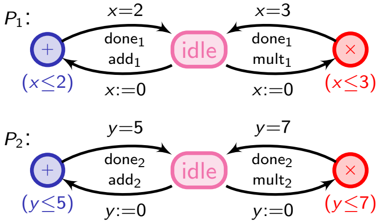
$\rightsquigarrow$ build the synchronized product of all these automata

$$(P_1 \parallel P_2) \parallel_s (T_1 \parallel T_2 \parallel \cdots \parallel T_6)$$

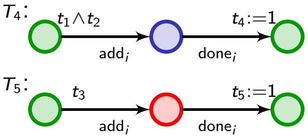
A schedule: a path in the global system which reaches $t_1 \wedge \cdots \wedge t_6$

Modelling the task graph scheduling problem

- Processors



- Tasks



$\rightsquigarrow$ build the synchronized product of all these automata

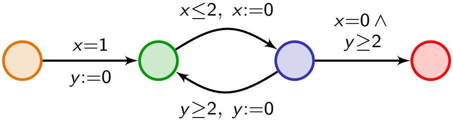
$$(P_1 \parallel P_2) \parallel_s (T_1 \parallel T_2 \parallel \cdots \parallel T_6)$$

A schedule: a path in the global system which reaches $t_1 \wedge \cdots \wedge t_6$

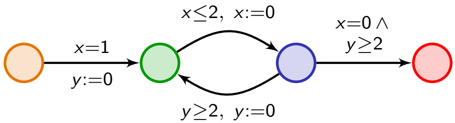
Questions one can ask

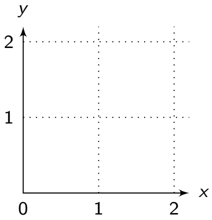
- Can the computation be made in no more than 10 time units?
- Is there a scheduling along which no processor is ever idle?
- ...

Analyzing timed automata

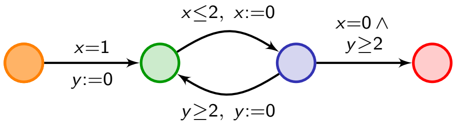


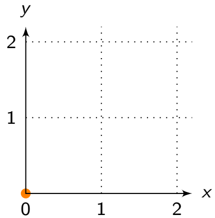
Analyzing timed automata



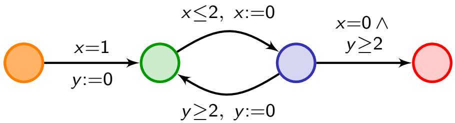


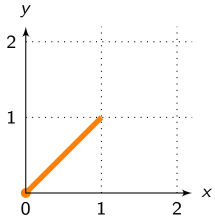
Analyzing timed automata



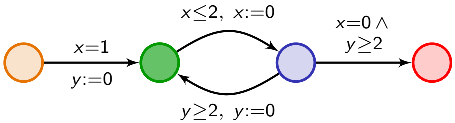


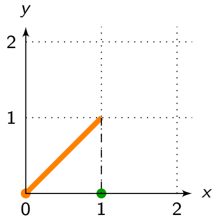
Analyzing timed automata



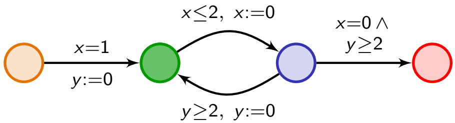


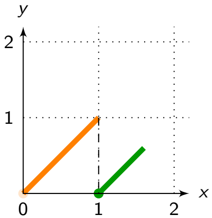
Analyzing timed automata



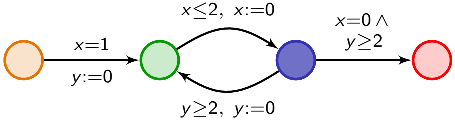


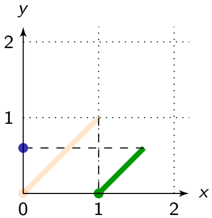
Analyzing timed automata



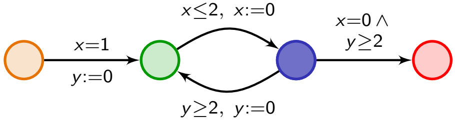


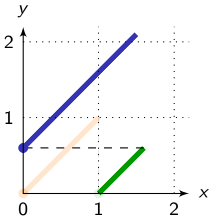
Analyzing timed automata



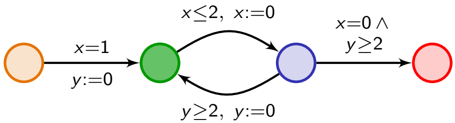


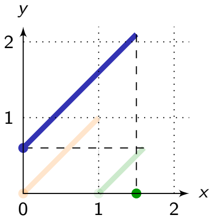
Analyzing timed automata



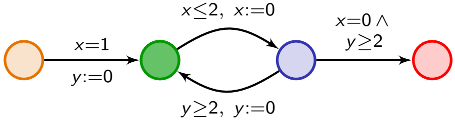


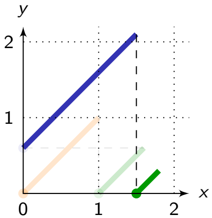
Analyzing timed automata



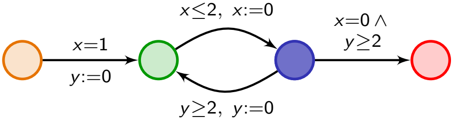


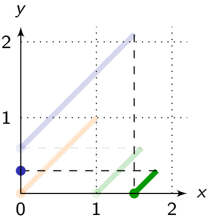
Analyzing timed automata



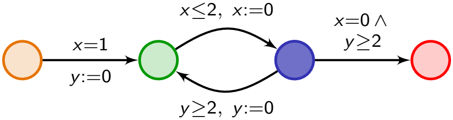


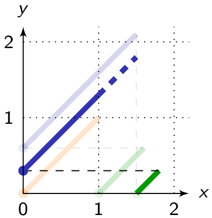
Analyzing timed automata



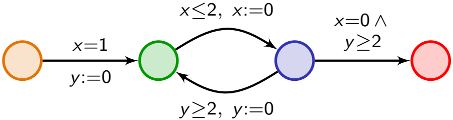


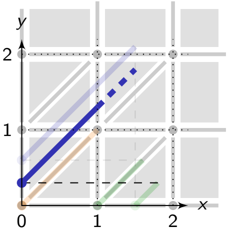
Analyzing timed automata



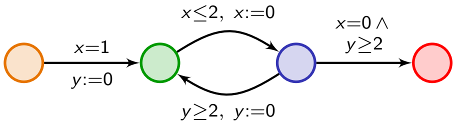


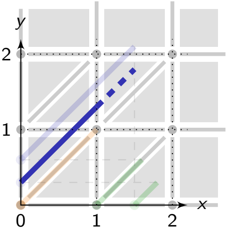
Analyzing timed automata





Analyzing timed automata





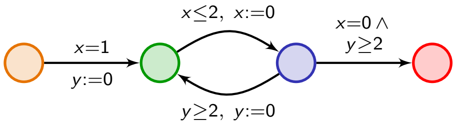
Theorem [AD94]

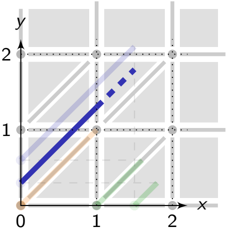
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction

[AD94] Alur, Dill. A Theory of Timed Automata *(Theoretical Computer Science)*.

Analyzing timed automata





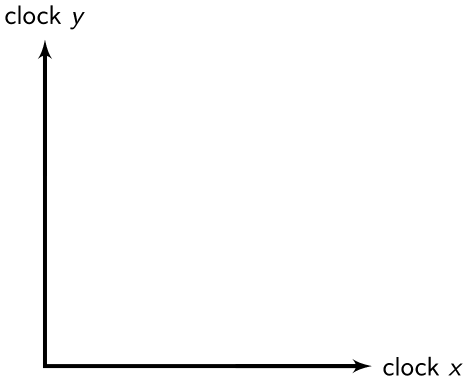
Theorem [AD94]

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

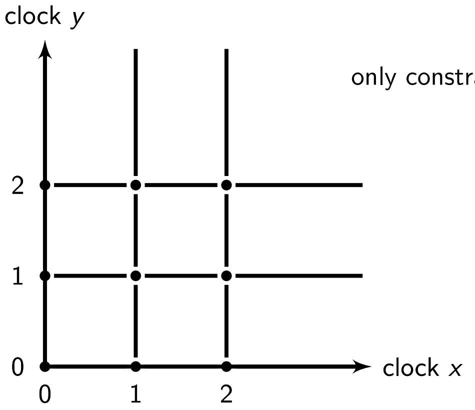
- Technical tool: region abstraction
- Efficient symbolic technics based on zones, implemented in tools

[AD94] Alur, Dill. A Theory of Timed Automata *(Theoretical Computer Science)*.

Technical tool: Region abstraction



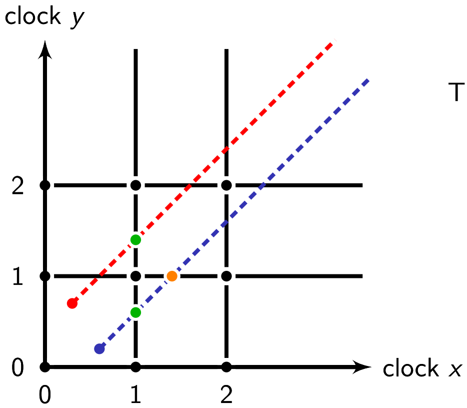
Technical tool: Region abstraction



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

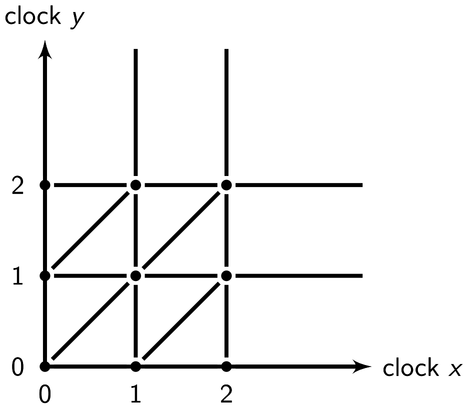


The path 

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

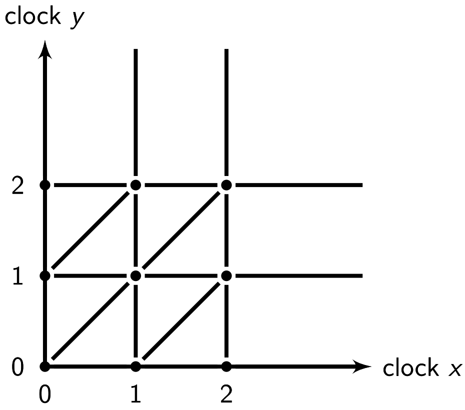
- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

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

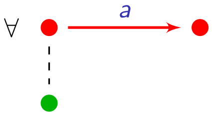
$\rightsquigarrow$ This is a finite time-abstract bisimulation!

Time-abstract bisimulation

This is a relation between • and • such that:

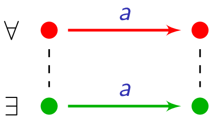
Time-abstract bisimulation

This is a relation between • and • such that:



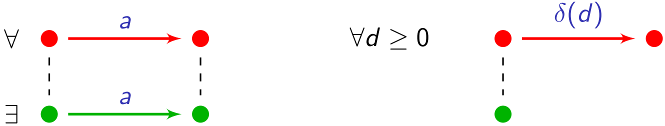
Time-abstract bisimulation

This is a relation between • and • such that:



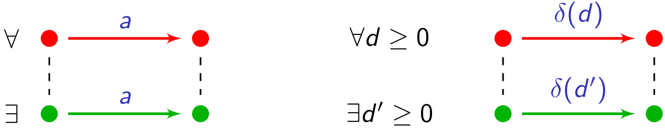
Time-abstract bisimulation

This is a relation between • and • such that:



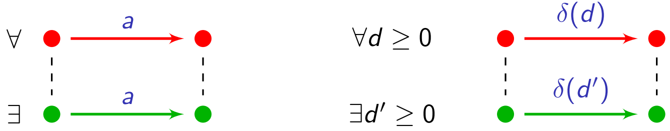
Time-abstract bisimulation

This is a relation between • and • such that:



Time-abstract bisimulation

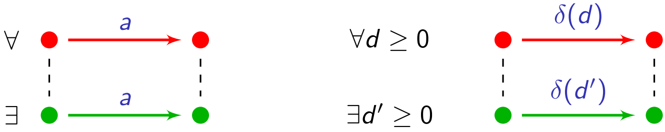
This is a relation between • and • such that:



... and vice-versa (swap • and •).

Time-abstract bisimulation

This is a relation between • and • such that:



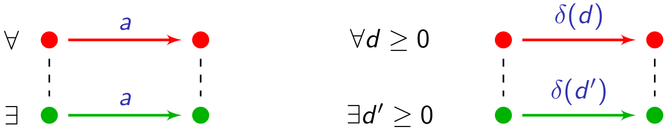
... and vice-versa (swap • and •).

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 •).

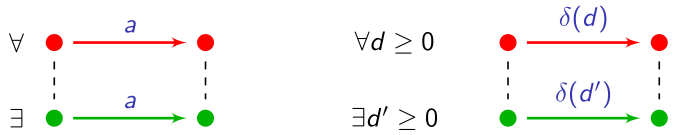
Consequence

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

$$(\ell_1, R_1) \xrightarrow{a_1} (\ell_2, R_2) \xrightarrow{a_2} (\ell_3, R_3) \xrightarrow{a_3} \cdots \quad \text{with } v_i \in R_i$$

Time-abstract bisimulation

This is a relation between • and • such that:



... and vice-versa (swap • and •).

Consequence

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

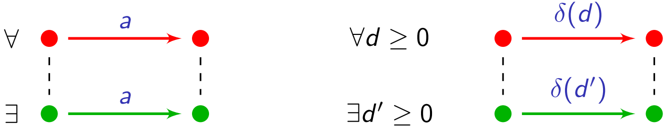
$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$(\ell_1, R_1) \xrightarrow{a_1} (\ell_2, R_2) \xrightarrow{a_2} (\ell_3, R_3) \xrightarrow{a_3} \cdots \quad \text{with } v_i \in R_i$$

$\forall v_1' \in R_1$

Time-abstract bisimulation

This is a relation between • and • such that:

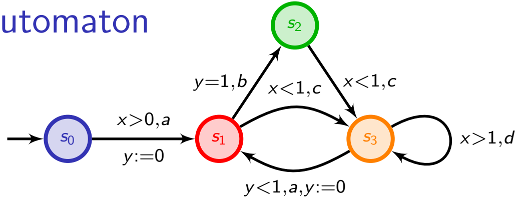


... and vice-versa (swap • and •).

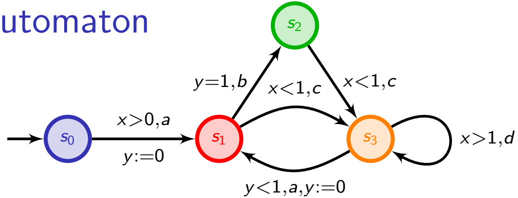
Consequence

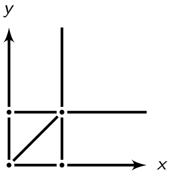
$$
\begin{array}{cccccccccc}
\forall & (\ell_1, v_1) & \xrightarrow{d_1, a_1} & (\ell_2, v_2) & \xrightarrow{d_2, a_2} & (\ell_3, v_3) & \xrightarrow{d_3, a_3} & \cdots & \\
 & \downarrow & & \downarrow & & \downarrow & & & \\
 & (\ell_1, R_1) & \xrightarrow{a_1} & (\ell_2, R_2) & \xrightarrow{a_2} & (\ell_3, R_3) & \xrightarrow{a_3} & \cdots & \text{with } v_i \in R_i \\
 & \downarrow & & \downarrow & & \downarrow & & & \\
\forall v'_1 \in R_1 \; \exists & (\ell_1, v'_1) & \xrightarrow{d'_1, a_1} & (\ell_2, v'_2) & \xrightarrow{d'_2, a_2} & (\ell_3, v'_3) & \xrightarrow{d'_3, a_3} & \cdots & \text{with } v'_i \in R_i
\end{array}
$$

The region automaton

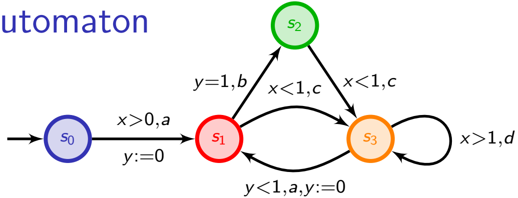


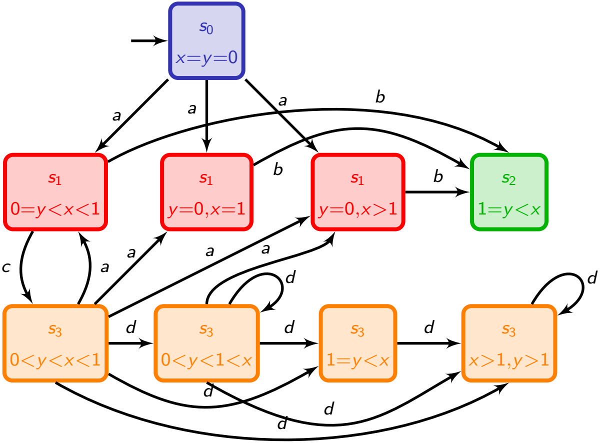
The region automaton

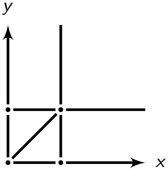




The region automaton







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,
  - ...

$\leadsto$ 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,

  $\rightsquigarrow$ 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

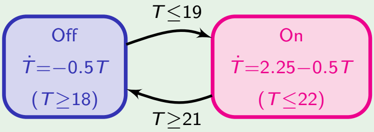
Modelling resources in timed systems

- 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



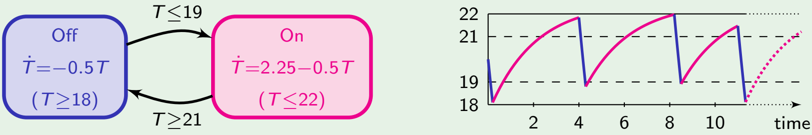
Modelling resources in timed systems

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

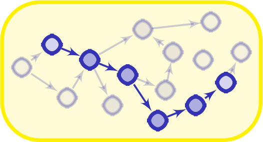
$\rightsquigarrow$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

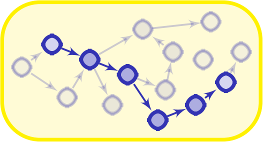
The thermostat example



Ok...

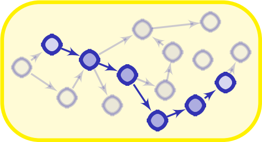


Ok...

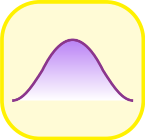


Easy...

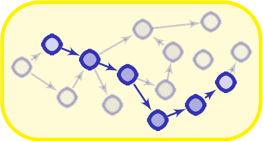
Ok...



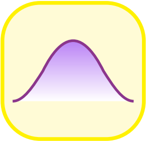
Easy...



Ok...

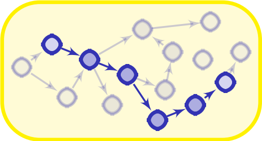


Easy...

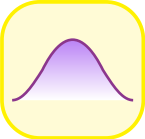


Easy...

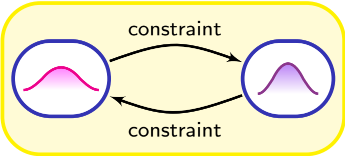
Ok... but?



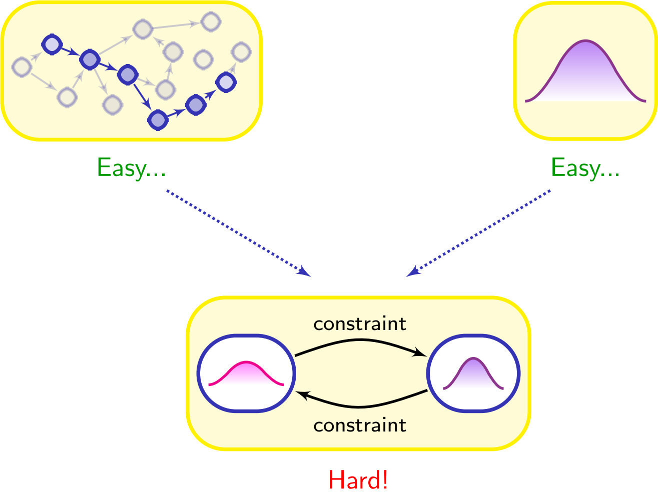
Easy...



Easy...



Ok... but?



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]

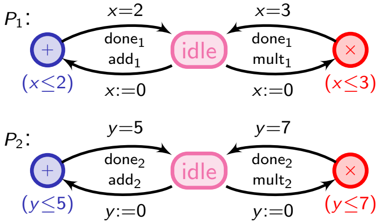
The reachability problem is undecidable in hybrid automata. Even for the simplest, the so-called stopwatch automata (clocks can be stopped).

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

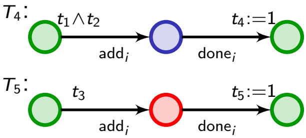
[HKPV95] Henzinger, Kopke, Puri, Varaiya. What's decidable wbout hybrid automata? *(SToC'95)*.
[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata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

Modelling the task graph scheduling problem

- Processors

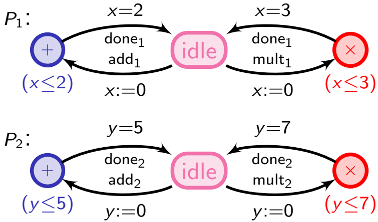


- Tasks

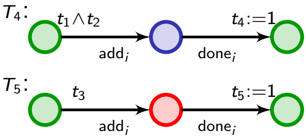


Modelling the task graph scheduling problem

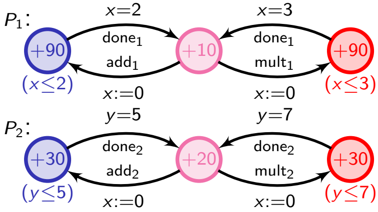
- Processors



- Tasks

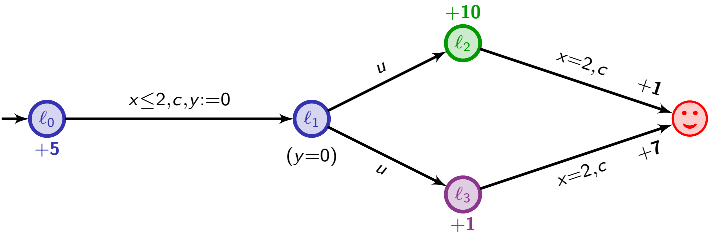


- Modelling energy



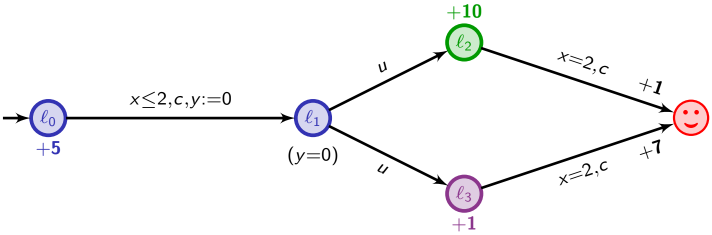
A good schedule is a path in the product automaton with a low cost

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)*.

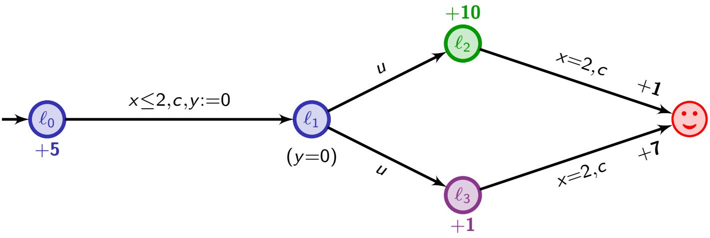
Weighted/priced timed automata [ALP01,BFH+01]



	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		

[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]

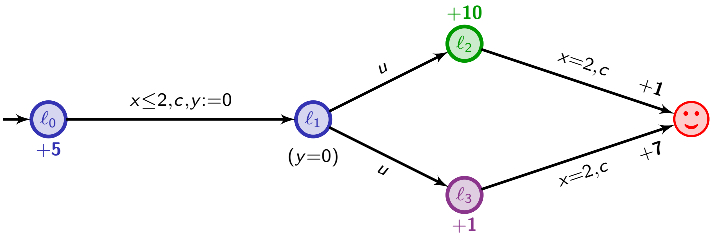


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

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]

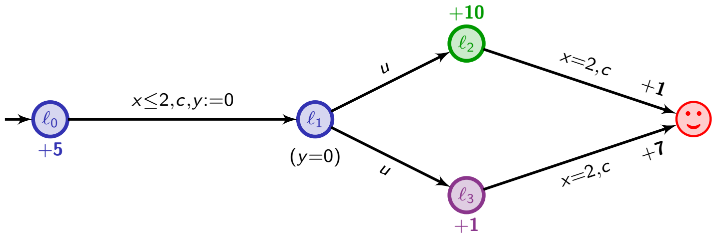


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

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]

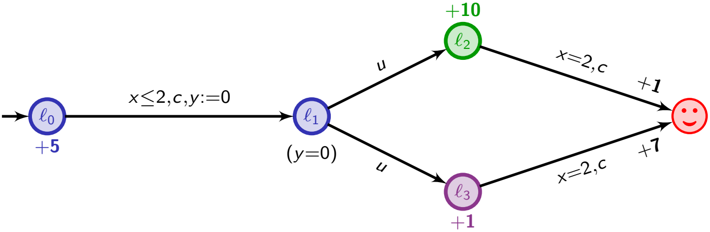


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

cost : 6.5 + 0

[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]

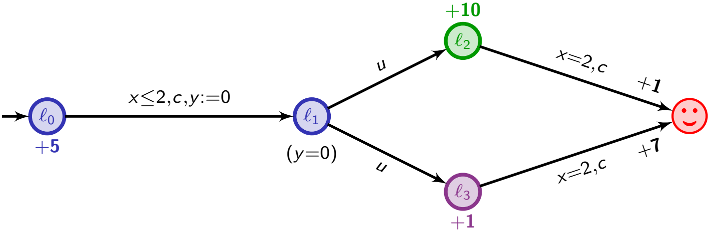


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

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)*.

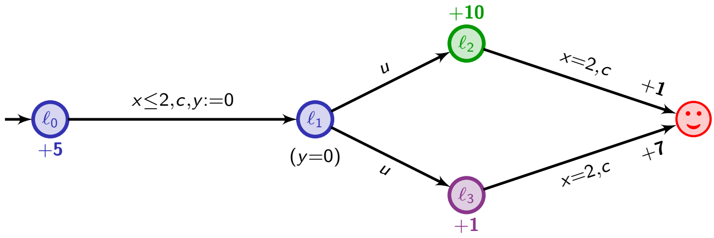
Weighted/priced timed automata [ALP01,BFH+01]



	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)*.

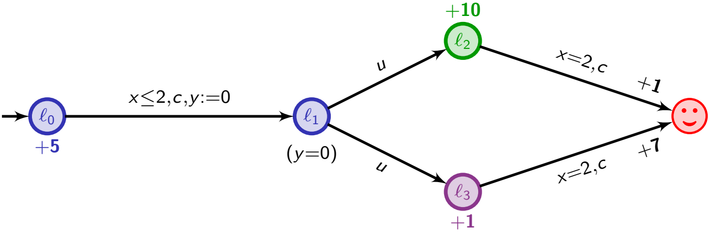
Weighted/priced timed automata [ALP01,BFH+01]



	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]

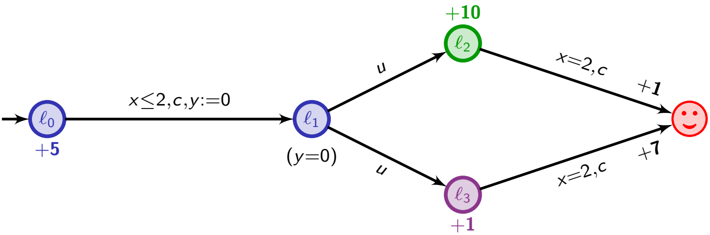


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

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)*.

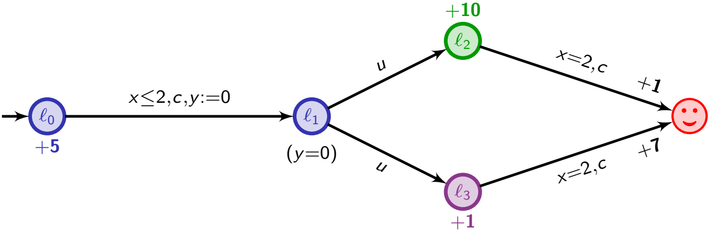
Weighted/priced timed automata [ALP01,BFH+01]



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]

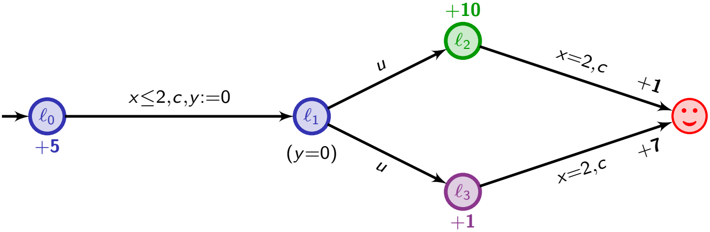


Question: what is the optimal cost for reaching ☺?

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

[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]

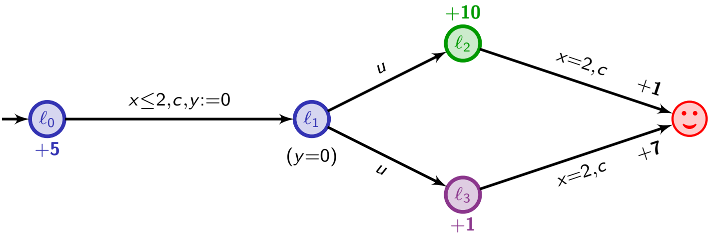


Question: what is the optimal cost for reaching ☺ ?

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

[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]

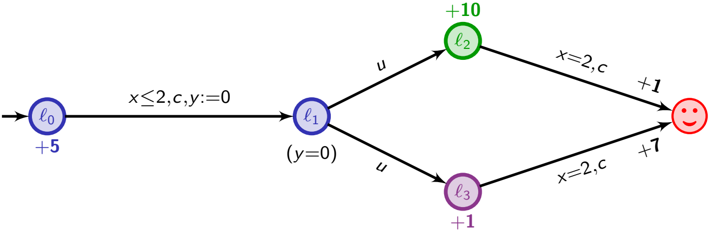


Question: what is the optimal cost for reaching 🙂?

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

[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]

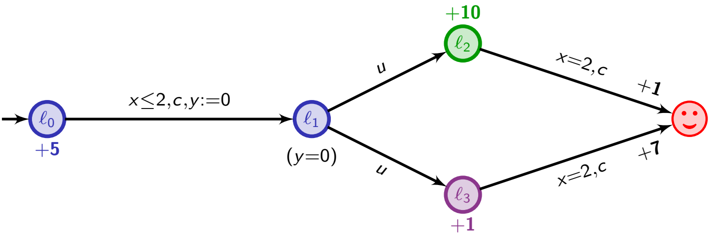


Question: what is the optimal cost for reaching ☺?

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

[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.

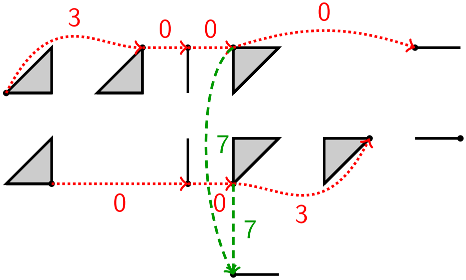
[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

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



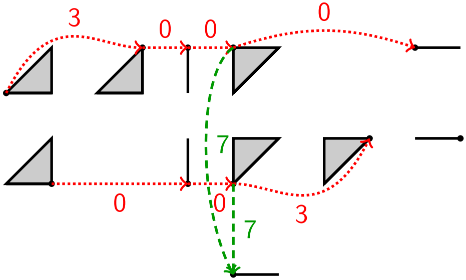
[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.
[BBBR07] Bouyer, Brihaye, Bruyère, Raskin. On the optimal reachability problem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

Optimal-cost reachability

Theorem [ALP01,BFH+01,BBBR07]

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

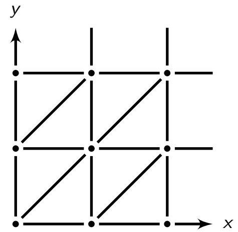
- Technical tool: a refinement of the regions, the corner-point abstraction



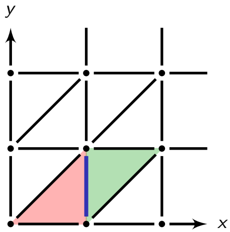
- Symbolic technics based on priced zones

[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)*.

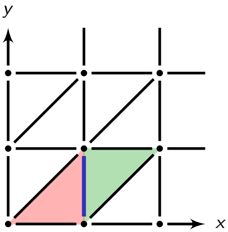
Technical tool: the corner-point abstraction



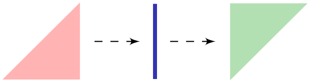
Technical tool: the corner-point abstraction



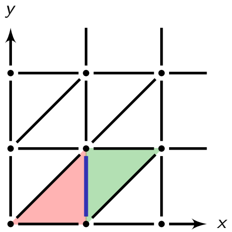
Technical tool: the corner-point abstraction



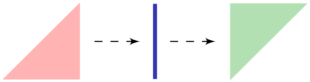
Abstract time successors:



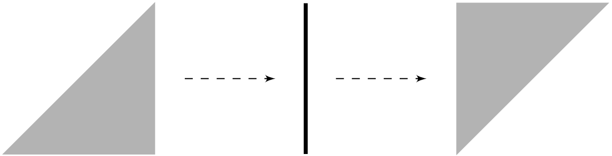
Technical tool: the corner-point abstraction



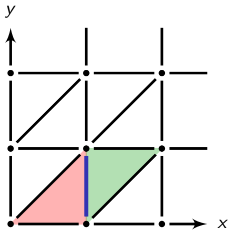
Abstract time successors:



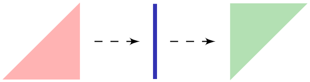
Concrete time successors:



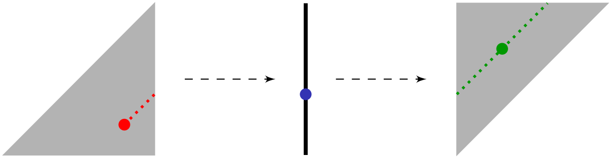
Technical tool: the corner-point abstraction



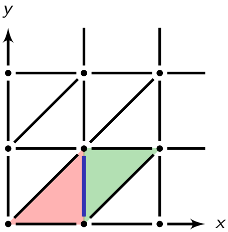
Abstract time successors:



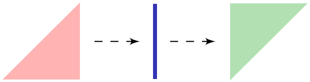
Concrete time successors:



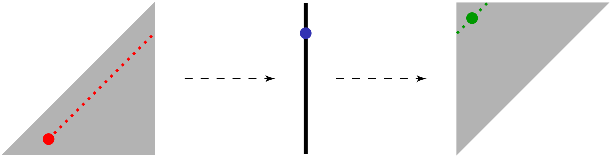
Technical tool: the corner-point abstraction



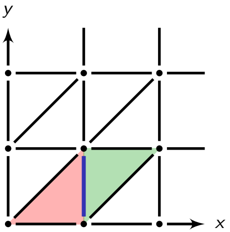
Abstract time successors:



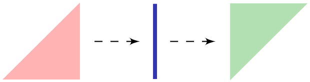
Concrete time successors:



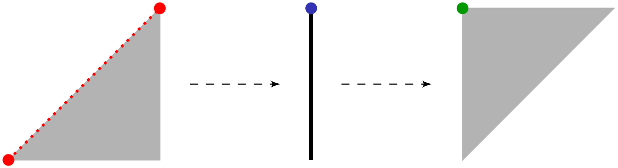
Technical tool: the corner-point abstraction



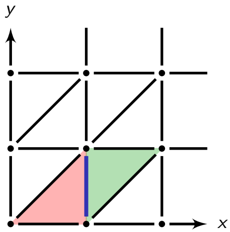
Abstract time successors:



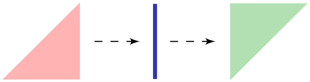
Concrete time successors:



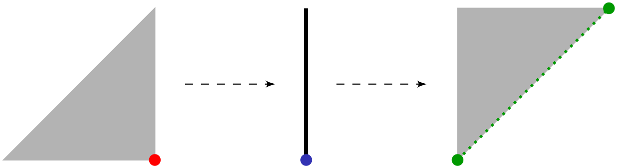
Technical tool: the corner-point abstraction



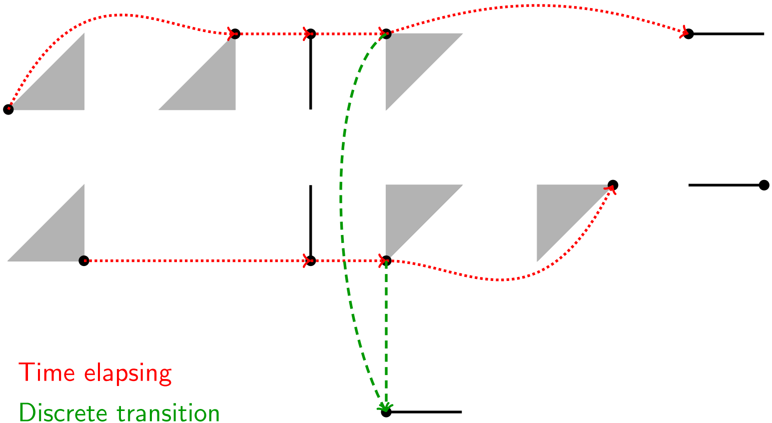
Abstract time successors:



Concrete time successors:



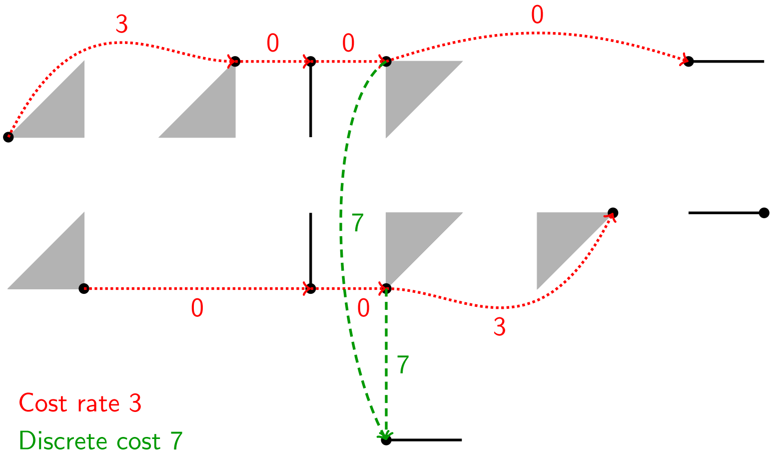
Technical tool: the corner-point abstraction



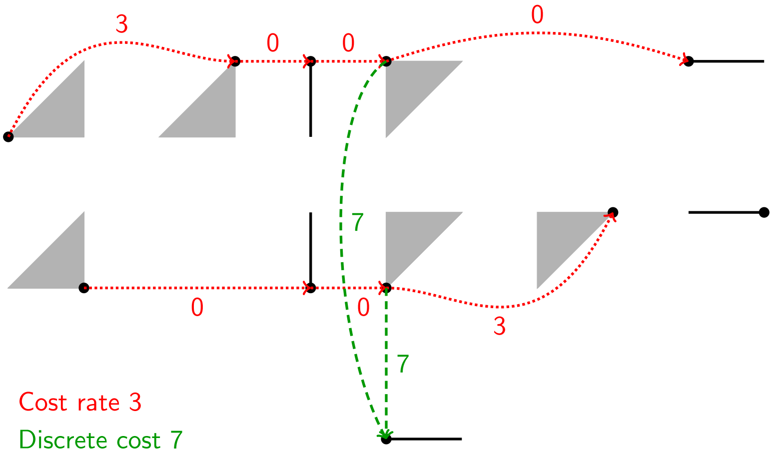
Time elapsing

Discrete transition

Technical tool: the corner-point abstraction



Technical tool: the corner-point abstraction

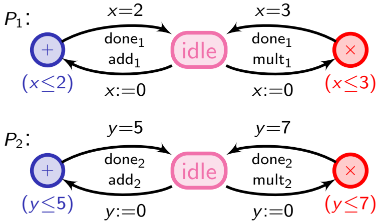


Optimal cost in the weighted graph
= optimal cost in the weighted timed automaton!

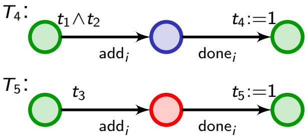
Outline

Modelling the task graph scheduling problem

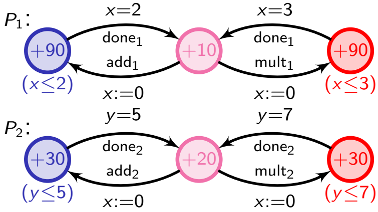
- Processors



- Tasks

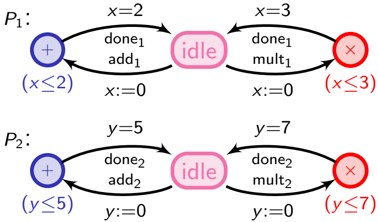


- Modelling energy

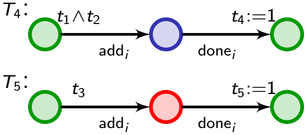


Modelling the task graph scheduling problem

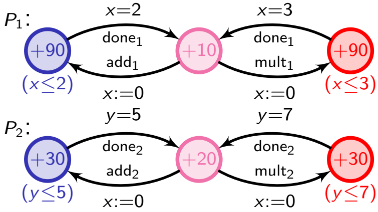
- Processors



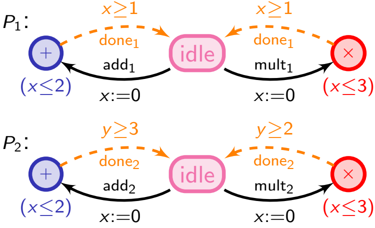
- Tasks



- Modelling energy



- Modelling uncertainty



Modelling the task graph scheduling problem

- Processors

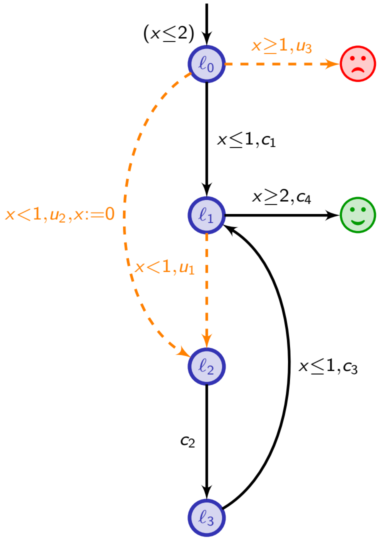
- Tasks

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

- Modelling energy

- Modelling uncertainty

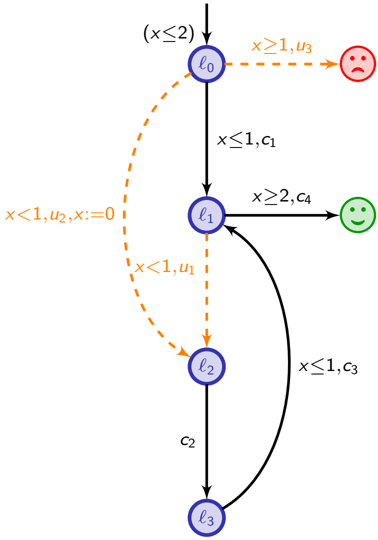
An example of a timed game



Rule of the game

- Aim: avoid ☹ and reach ☺

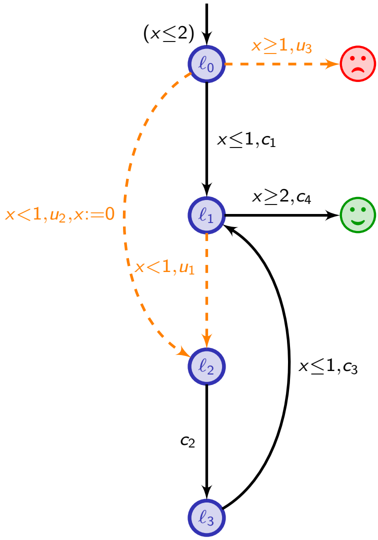
An example of a timed game



Rule of the game

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

An example of a timed game

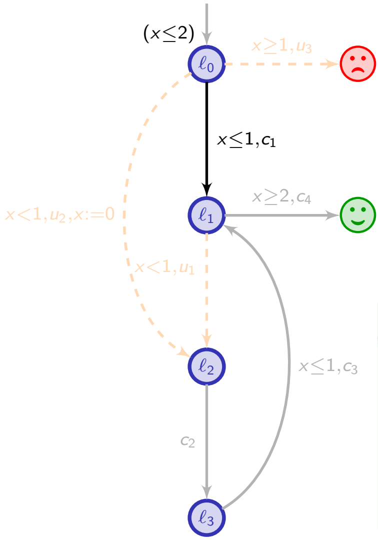


Rule of the game

- 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

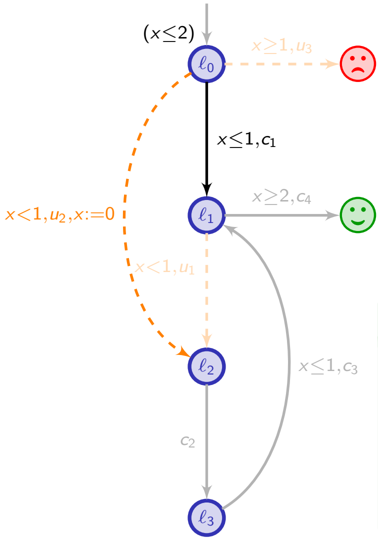
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

A (memoryless) winning strategy

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

An example of a timed game



Rule of the game

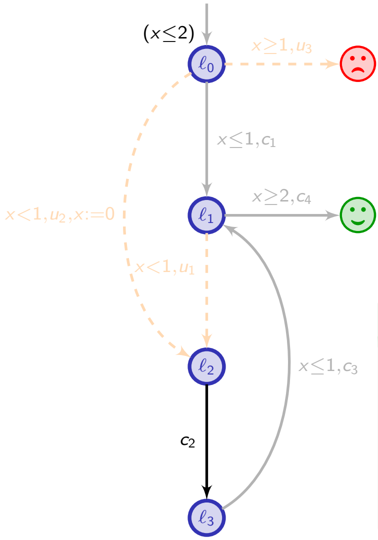
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

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

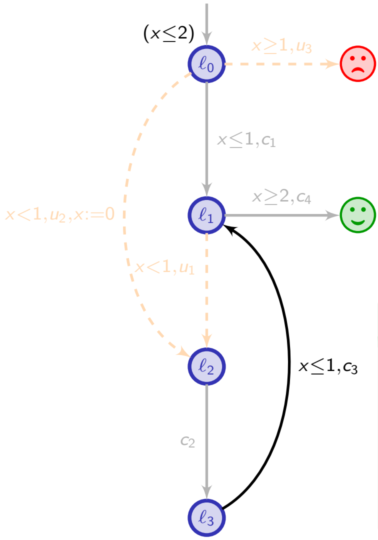
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

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

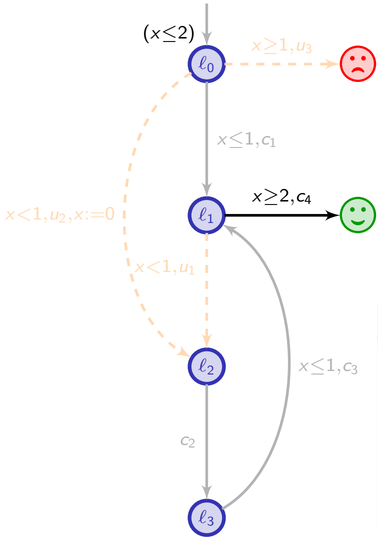
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

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

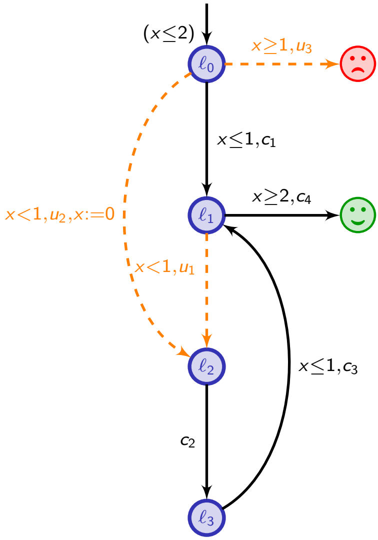
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

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



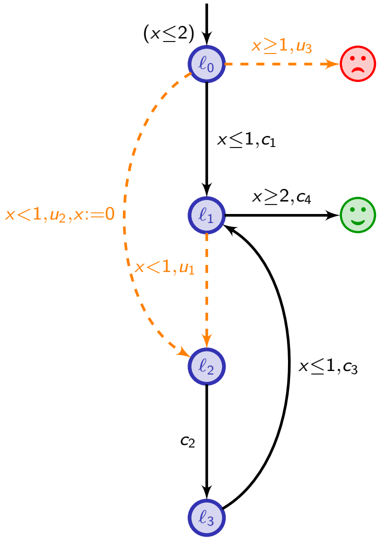
Rule of the game

- 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

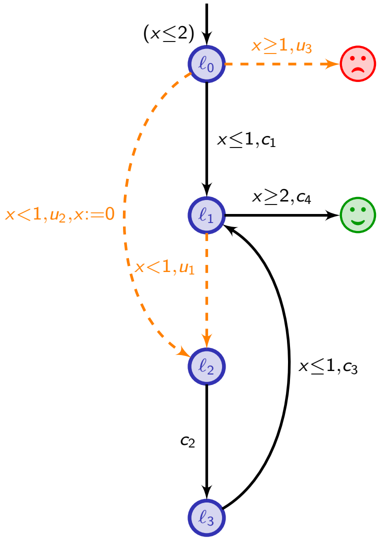
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

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
a region-closed attractor can be computed

[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
a region-closed attractor can be computed

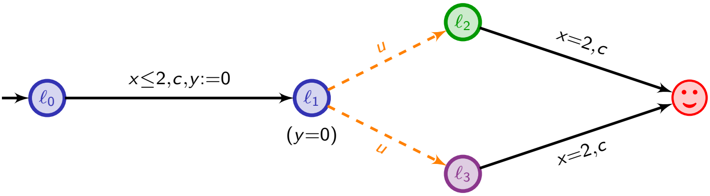
Theorem [AM99,BHPR07,JT07]

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

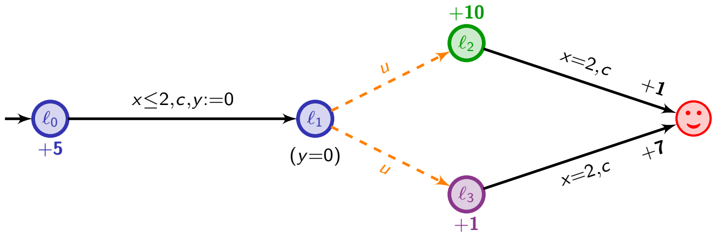
[AM99] Asarin, Maler. As soon as possible: time optimal control for timed automata *(HSCC'99)*.
[BHPR07] Brihaye, Henzinger, Prabhu, Raskin. Minimum-time reachability in timed games *(ICALP'07)*.
[JT07] Jurdziński, Trivedi. Reachability-time games on timed automata *(ICALP'07)*.

Outline

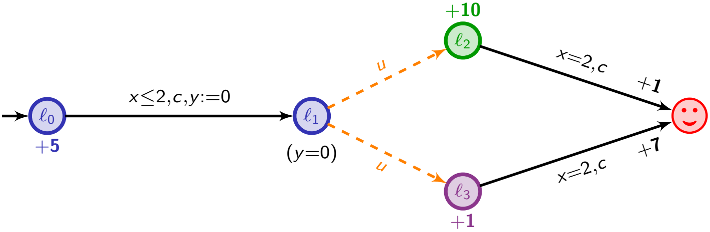
A simple timed game



A simple weighted timed game

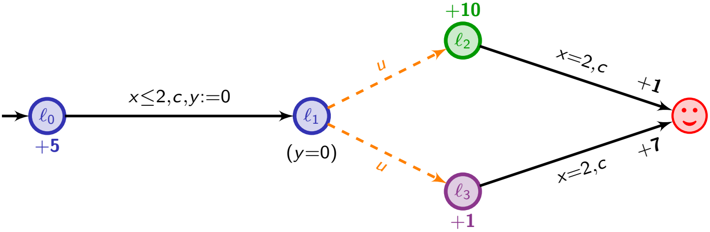


A simple weighted timed game



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

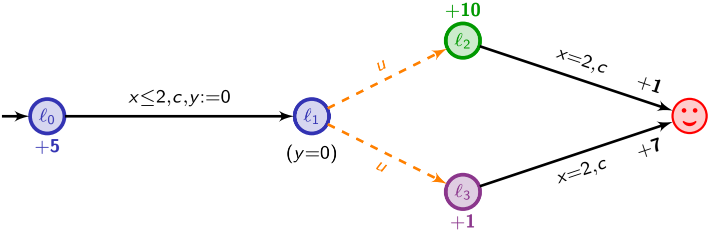
A simple weighted timed game



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

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

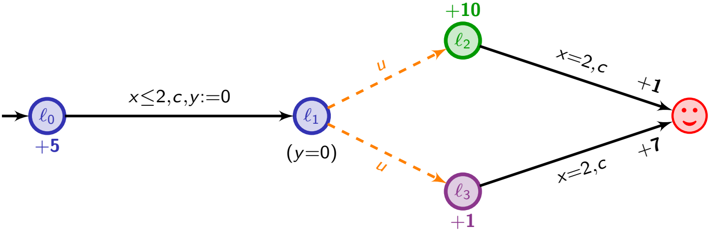
A simple weighted timed game



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

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

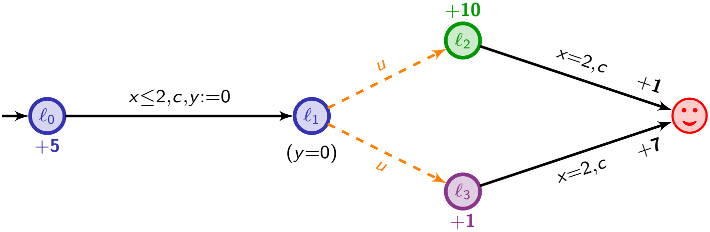
A simple weighted timed game



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

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

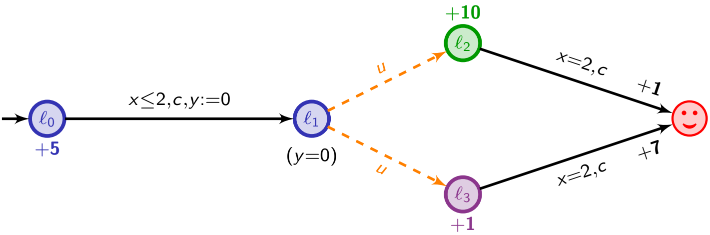
A simple weighted timed game



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 ☺?

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

$\rightsquigarrow$ *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...

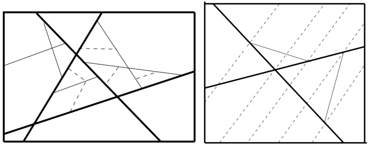
[LMM02,ABM04,BCFL04,BBR05,BBM06,BLMR06,Rut11,HIM13,BGK+14]

[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,BJM15]

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

Optimal reachability in weighted timed games (2)

[BBR05,BBM06,BJM15]

In weighted timed games, the optimal cost (and the value) 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.

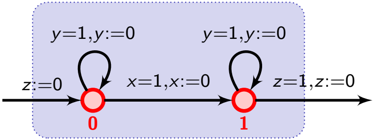
Computing the optimal cost: why is that hard?

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

Computing the optimal cost: why is that hard?

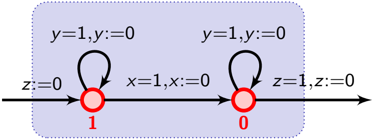
Given two clocks x and y, we can check whether $y = 2x$.

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



The cost is increased by x_0

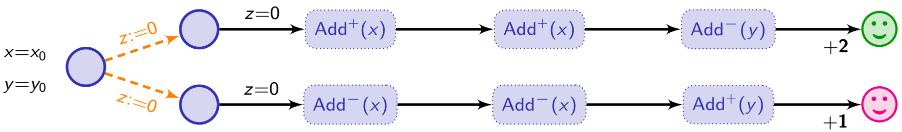
$\text{Add}^-(x)$



The cost is increased by $1-x_0$

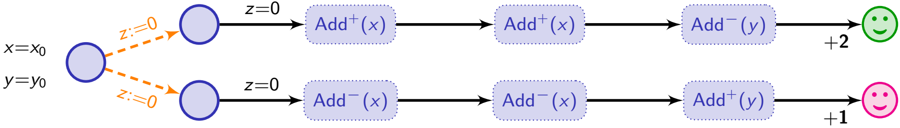
Computing the optimal cost: why is that hard?

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



Computing the optimal cost: why is that hard?

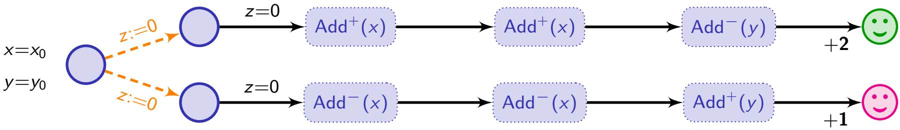
Given two clocks x and y, we can check whether $y = 2x$.



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

Computing the optimal cost: why is that hard?

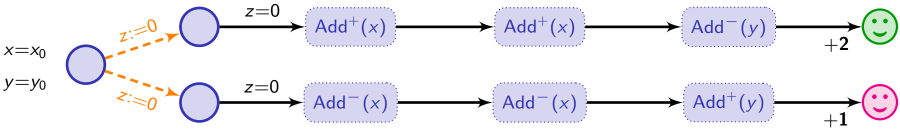
Given two clocks x and y, we can check whether $y = 2x$.



- In , 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 $y = 2x$.

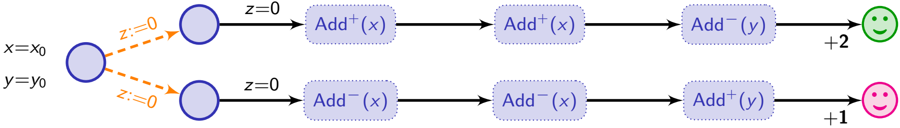


- In , cost $= 2x_0 + (1 - y_0) + 2$

  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?

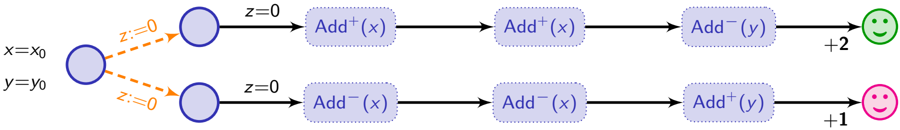
Given two clocks x and y, we can check whether $y = 2x$.



- 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?

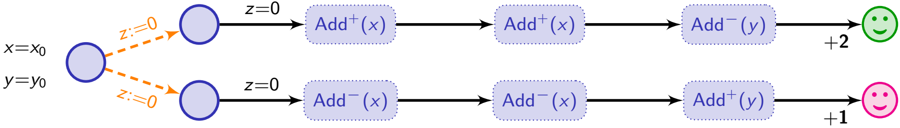
Given two clocks x and y, we can check whether $y = 2x$.



- 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 , 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

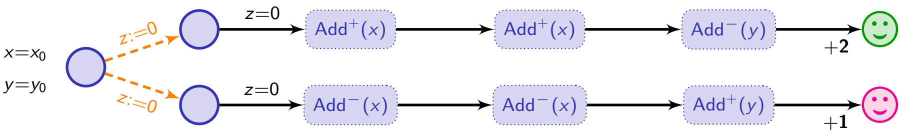
  if $y_0 > 2x_0$, player 2 chooses the second branch: cost > 3

  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?

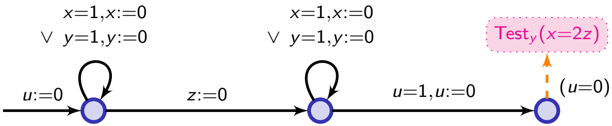
Player 1 will simulate a two-counter machine:

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



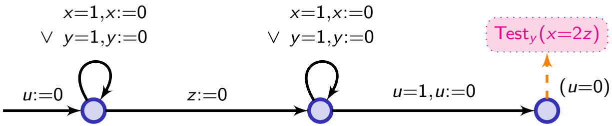
Computing the optimal cost: why is that hard?

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:

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

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



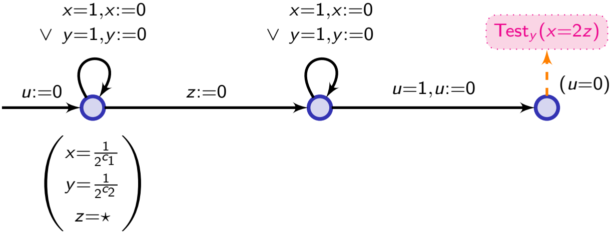
Computing the optimal cost: why is that hard?

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.



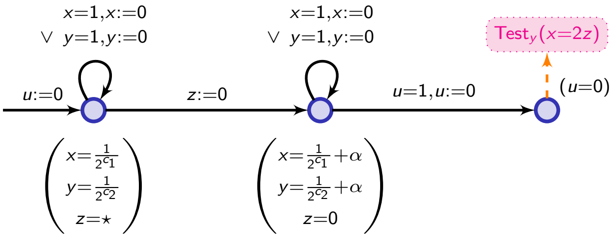
Computing the optimal cost: why is that hard?

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.



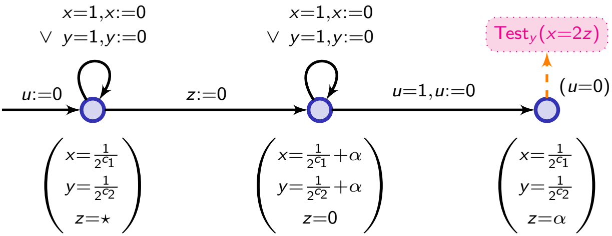
Computing the optimal cost: why is that hard?

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.



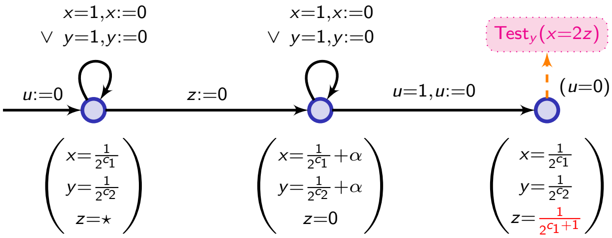
Computing the optimal cost: why is that hard?

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.



Are we done?

Are we done? No!

Are we done? No!

Optimal cost is computable...

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

There is $\kappa > 0$ s.t. 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.

There is $\kappa > 0$ s.t. for every region cycle C, for every real run ϱ read on C,

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

Are we done? No!

Optimal cost is computable...

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

There is $\kappa > 0$ s.t. for every region cycle C, for every real run ϱ read on C,

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

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

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

There is $\kappa > 0$ s.t. for every region cycle C, for every real run ϱ read on C,

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

[BJM15] Bouyer, Jaziri, Markey. On the value problem in weighted timed games *(CONCUR'15)*.

Are we done? No!

Optimal cost is computable...

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

There is $\kappa > 0$ s.t. 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]

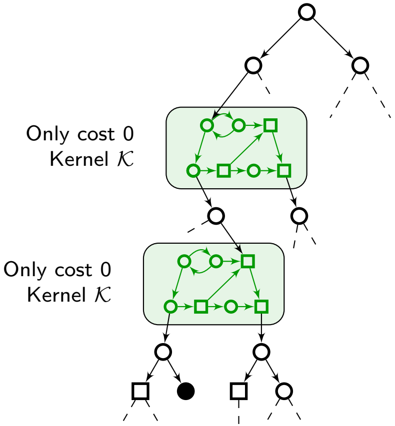
There is $\kappa > 0$ s.t. for every region cycle C, for every real run ϱ read on C,

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

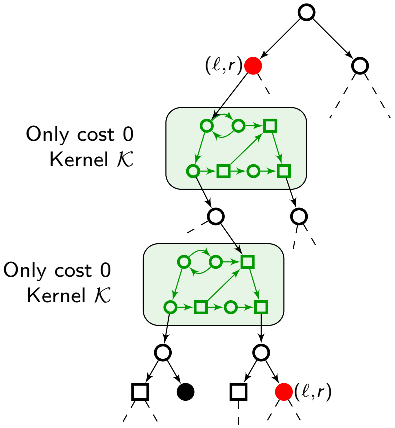
- Almost-optimality in practice should be sufficient
- Even when we know how to compute the value, we are only able to synthesize almost-optimal strategies...

[BJM15] Bouyer, Jaziri, Markey. On the value problem in weighted timed games *(CONCUR'15)*.

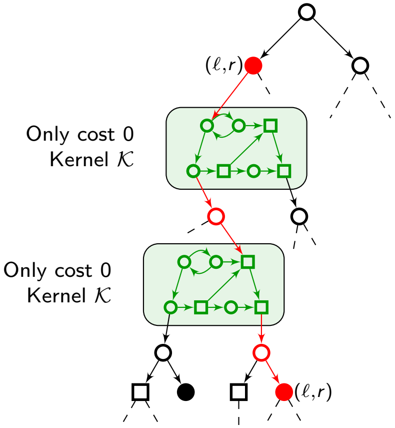
Idea of the proof: Semi-unfolding



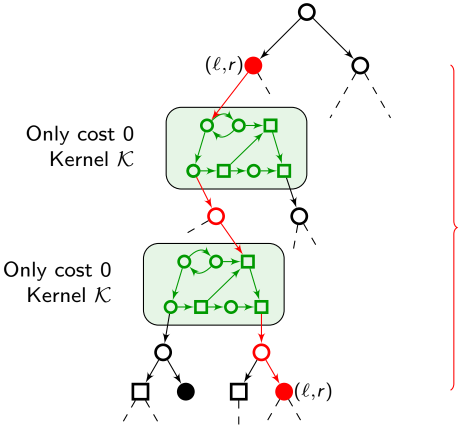
Idea of the proof: Semi-unfolding



Idea of the proof: Semi-unfolding

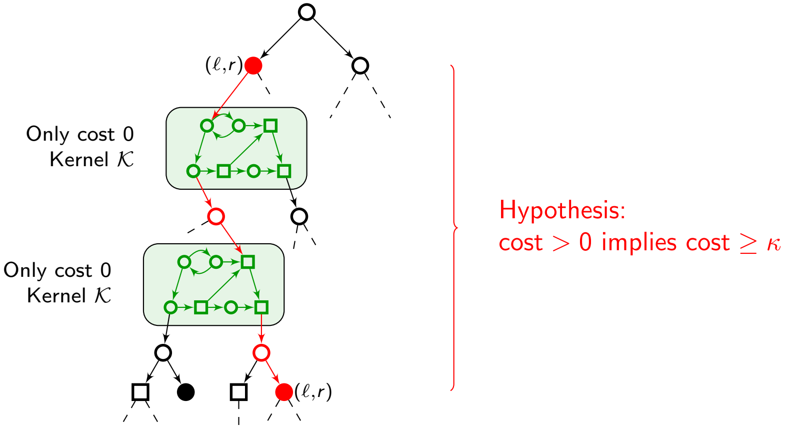


Idea of the proof: Semi-unfolding



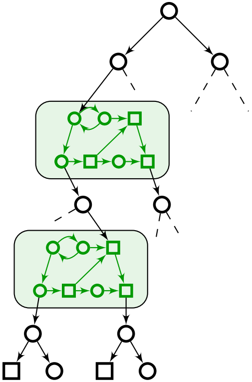
Hypothesis:
cost > 0 implies cost $\geq \kappa$

Idea of the proof: Semi-unfolding

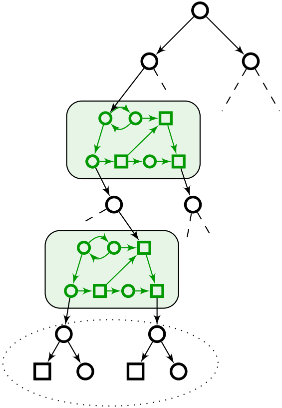


Conclusion: we can stop unfolding the game after finitely many steps

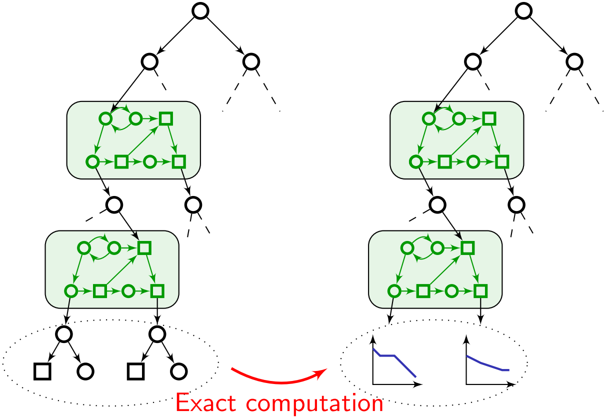
Approximation scheme



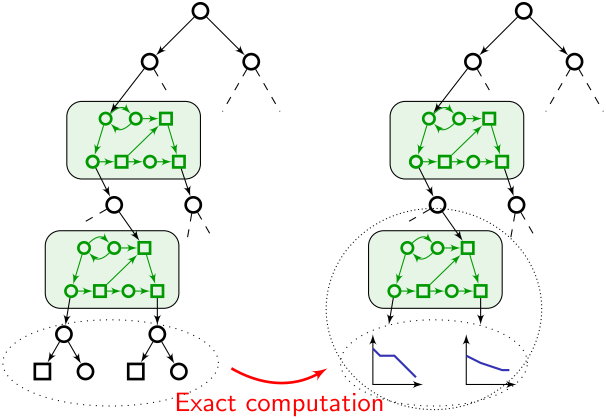
Approximation scheme



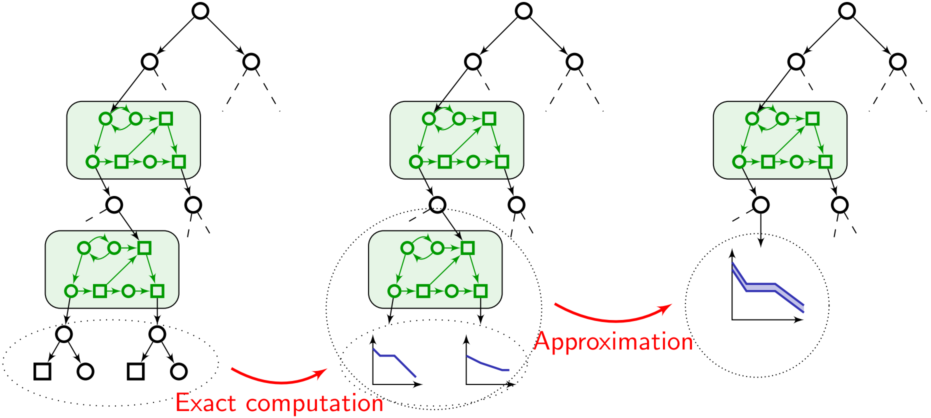
Approximation scheme



Approximation scheme



Approximation scheme



Outline

Tools for (weighted) timed automata and games

- Many tools and prototypes everywhere on earth...

Tools for (weighted) timed automata and games

- Many tools and prototypes everywhere on earth...
- Tool-suite Uppaal, developed in Aalborg (Denmark) and originally Uppsala (Sweden) since 1995
  - Uppaal for timed automata
  - Uppaal-TiGa for timed games
  - Uppaal-Cora for weighted timed automata

Uppaal url: `http://www.uppaal.org`

Tools for (weighted) timed automata and games

- Many tools and prototypes everywhere on earth...
- Tool-suite Uppaal, developed in Aalborg (Denmark) and originally Uppsala (Sweden) since 1995

Uppaal url: http://www.uppaal.org

Tools for (weighted) timed automata and games

- Many tools and prototypes everywhere on earth...
- Tool-suite Uppaal, developed in Aalborg (Denmark) and originally Uppsala (Sweden) since 1995
- Our new tool TiAMo, developed by Maximilien Colange (LSV), using code by Ocan Sankur (IRISA)

TiAMo = Timed Automata Model-checker

Uppaal url: http://www.uppaal.org
TiAMo url: https://git.lsv.fr/colange/tiamo

[BCM16] Bouyer, Colange, Markey. Symbolic optimal reachability in weighted timed automata *(CAV'16)*.

Tools for (weighted) timed automata and games

- Many tools and prototypes everywhere on earth...
- Tool-suite Uppaal, developed in Aalborg (Denmark) and originally Uppsala (Sweden) since 1995
- Our new tool TiAMo, developed by Maximilien Colange (LSV), using code by Ocan Sankur (IRISA)

TiAMo = Timed Automata
Model-checker

- Timed automata:
  (time-optimal) reachability
- Weighted timed automata:
  optimal rechability

Uppaal url: http://www.uppaal.org
TiAMo url: https://git.lsv.fr/colange/tiamo
[BCM16] Bouyer, Colange, Markey. Symbolic optimal reachability in weighted timed automata *(CAV'16)*.

Tools for (weighted) timed automata and games

- Many tools and prototypes everywhere on earth...
- Tool-suite Uppaal, developed in Aalborg (Denmark) and originally Uppsala (Sweden) since 1995
- Our new tool TiAMo, developed by Maximilien Colange (LSV), using code by Ocan Sankur (IRISA)

TiAMo = Timed Automata Model-checker

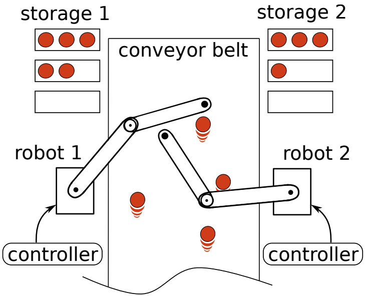
- Timed automata: (time-optimal) reachability
- Weighted timed automata: optimal rechability
- Aims at being a platform for experiments (open source!)
- Aims at asserting and comparing algorithms

Uppaal url: http://www.uppaal.org
TiAMo url: https://git.lsv.fr/colange/tiamo
[BCM16] Bouyer, Colange, Markey. Symbolic optimal reachability in weighted timed automata *(CAV'16)*.

Outline

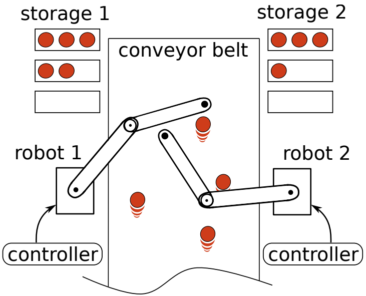
1. Timed automata
2. Weighted timed automata
3. Timed games
4. Weighted timed games
5. Tools
6. Towards applying all this theory to robotic systems
7. Conclusion

Example problem, objective and approach



[BMPS15] Bouyer, Markey, Perrin, Schlehuber-Caissier. Timed-Automata Abstraction of Switched Dynamical Systems Using Control Funnels *(FORMATS'15)*.

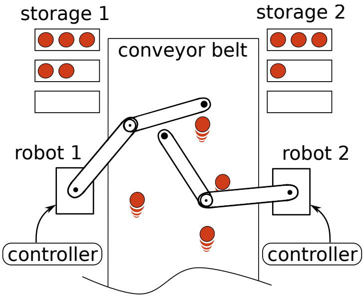
Example problem, objective and approach



- Infinitely many configurations
- Complex behaviour
- Mechanical constraints

[BMPS15] Bouyer, Markey, Perrin, Schlehuber-Caissier. Timed-Automata Abstraction of Switched Dynamical Systems Using Control Funnels *(FORMATS'15)*.

Example problem, objective and approach



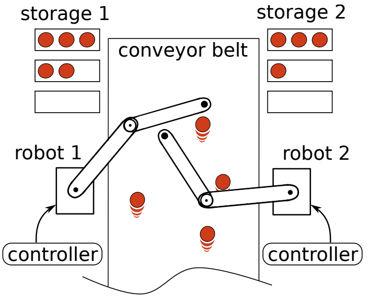
Goal: Synthesize a controller:

- Which robot handles an object
- How to avoid collision
- Don't miss any object

- Infinitely many configurations
- Complex behaviour
- Mechanical constraints

[BMPS15] Bouyer, Markey, Perrin, Schlehuber-Caissier. Timed-Automata Abstraction of Switched Dynamical Systems Using Control Funnels *(FORMATS'15)*.

Example problem, objective and approach



- Infinitely many configurations
- Complex behaviour
- Mechanical constraints

Goal: Synthesize a controller:

- Which robot handles an object
- How to avoid collision
- Don't miss any object

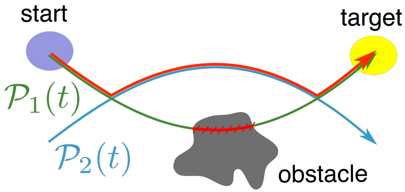
Approach:

- Discretization of the behaviour via a fixed set of continuous controllers
- Create an abstraction and use previous results

[BMPS15] Bouyer, Markey, Perrin, Schlehuber-Caissier. Timed-Automata Abstraction of Switched Dynamical Systems Using Control Funnels *(FORMATS'15)*.

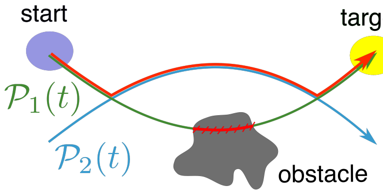
Our approach

Simplistic idea: fixed set of reference trajectories + property

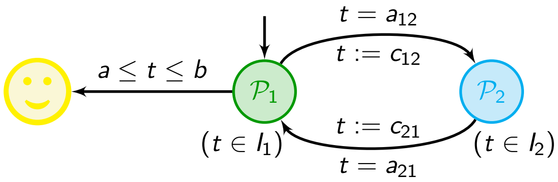


Our approach

Simplistic idea: fixed set of reference trajectories + property

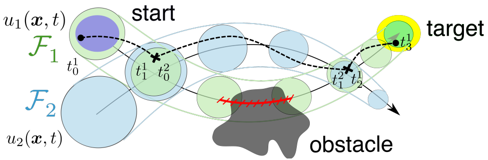


Corresponding timed automaton:



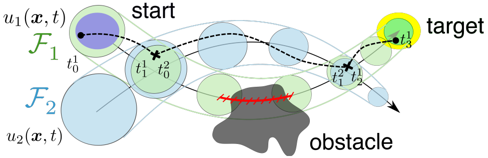
Our approach

More realistic idea: fixed set of funnels for control law + property

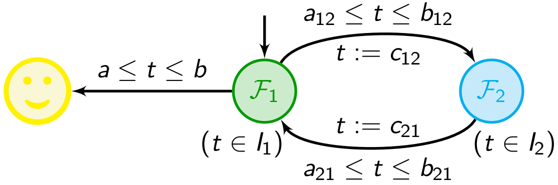


Our approach

More realistic idea: fixed set of funnels for control law + property

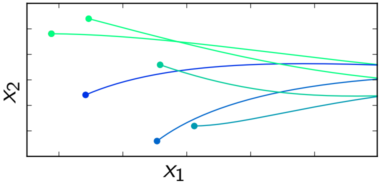


Corresponding timed automaton:



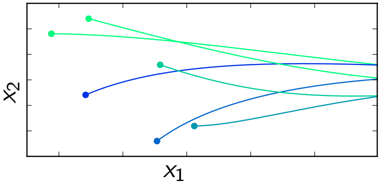
Control funnels

System with continuous dynamics $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, t)$



Control funnels

System with continuous dynamics $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, t)$

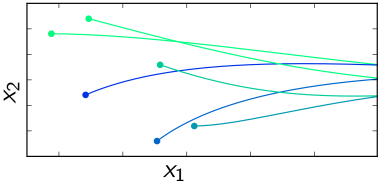


A (control) funnel is a trajectory $\mathcal{F}(t)$ of a set in the state space such that, for any trajectory $\boldsymbol{x}(t)$ of the dynamical system:

$$\forall t_0 \in \mathbb{R},\ \boldsymbol{x}(t_0) \in \mathcal{F}(t_0) \Rightarrow \forall t \geq t_0,\ x(t) \in \mathcal{F}(t)$$

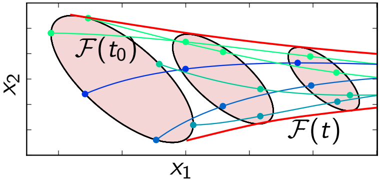
Control funnels

System with continuous dynamics $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, t)$

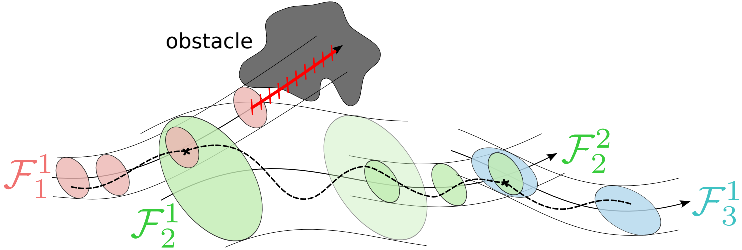


A (control) funnel is a trajectory $\mathcal{F}(t)$ of a set in the state space such that, for any trajectory $\boldsymbol{x}(t)$ of the dynamical system:

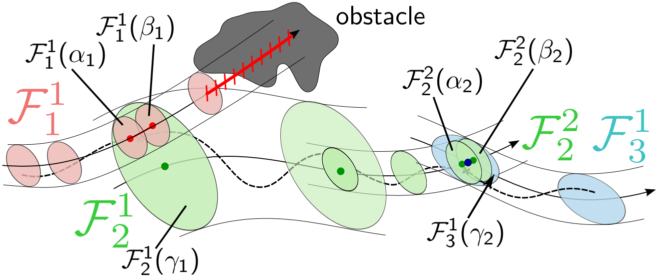
$$\forall t_0 \in \mathbb{R},\ \boldsymbol{x}(t_0) \in \mathcal{F}(t_0) \Rightarrow \forall t \geq t_0,\ x(t) \in \mathcal{F}(t)$$



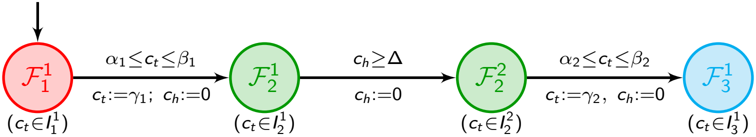
Example



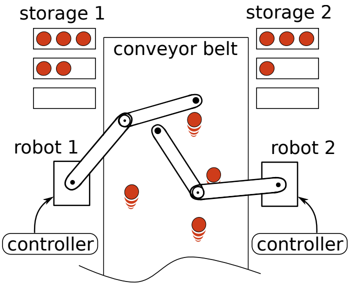
Example



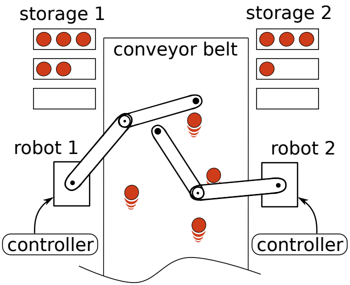
c_t: positional clock; c_h: local clock



Summary

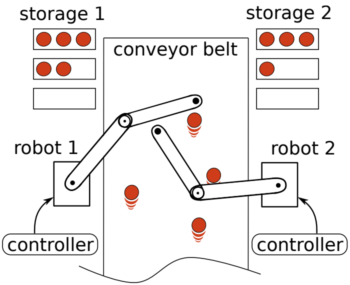

storage 1
storage 2
conveyor belt
robot 1
robot 2
controller
controller

Summary



$\rightsquigarrow$ (huge) timed automata/games (with weights), with few clocks

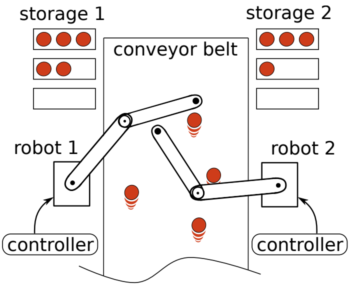
Summary



⇝ (huge) timed automata/games (with weights), with few clocks

← winning (optimal) strategy

Summary

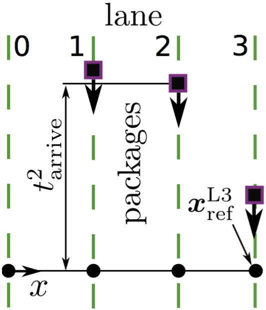


$\rightsquigarrow$ (huge) timed automata/games (with weights), with few clocks

safe (good) controller $\leftarrow$ winning (optimal) strategy

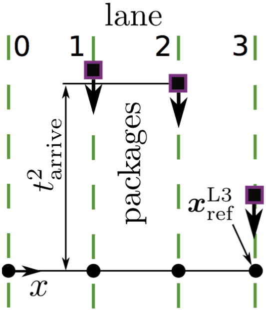
A pick-and-place example

1d point mass

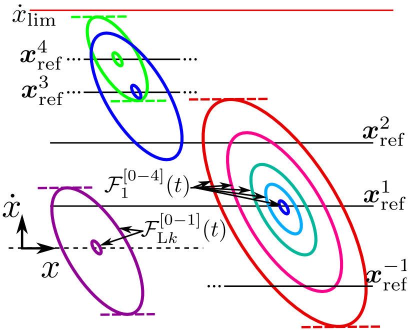


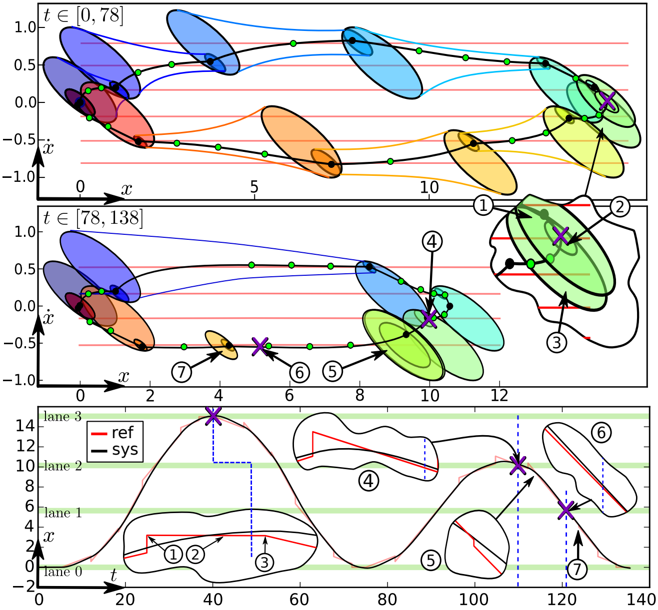
A pick-and-place example

1d point mass



Funnel system





$t \in [0, 78]$
$t \in [78, 138]$
$\dot{x}$
x
t
lane 0
lane 1
lane 2
lane 3
ref
sys
1
2
3
4
5
6
7

Current challenges

For control people

- Handle more non-linear systems (automatically build control funnels)

Current challenges

For control people

- Handle more non-linear systems (automatically build control funnels)

For us

- Does not scale up very well so far (huge timed automata models)
  - Build the model on-demand?
    But, can we give guarantees (optimality) when only part of the model has been built?
  - Develop specific algorithms for the special timed automata we construct?
- Implement efficient approx. algorithm for weighted timed games

Outline

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- Our new tool TiAMo

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- Our new tool TiAMo

Future work

- Various theoretical issues
  - Apply further the idea of approximation
  - Robustness issues

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- Our new tool TiAMo

Future work

- Various theoretical issues
  - Apply further the idea of approximation
  - Robustness issues
- Continue working on TiAMo
  - Implementation of (weighted) timed games (good data structures, abstractions, etc.)
  - More applications with specific challenges (e.g. robotic problems)