On the optimal reachability problem in weighted timed automata and games

Patricia Bouyer-Decitre

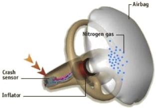
LSV, CNRS & ENS Cachan, France

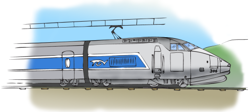
Time-dependent systems

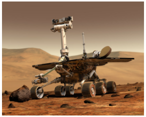
- We are interested in timed systems

Time-dependent systems

- We are interested in timed systems

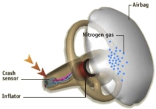


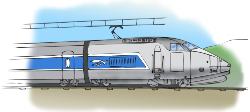


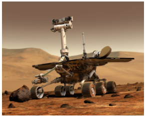


Time-dependent systems

- We are interested in timed systems







- ... and in their analysis and control

An example: The task graph scheduling problem

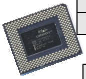
Compute $D \times (C \times (A + B)) + (A + B) + (C \times D)$ using two processors:

P_1 (fast):

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

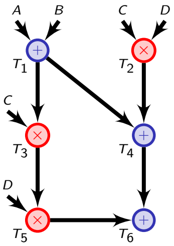
energy	
idle	10 Watt
in use	90 Watts

P_2 (slow):



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

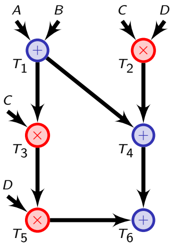
energy	
idle	20 Watts
in use	30 Watts



[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:

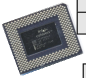


P_1 (fast):

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

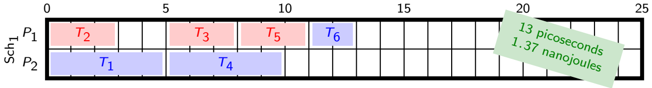
energy	
idle	10 Watt
in use	90 Watts

P_2 (slow):



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

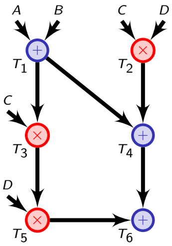
energy	
idle	20 Watts
in use	30 Watts



[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:

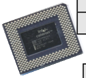


P_1 (fast):

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

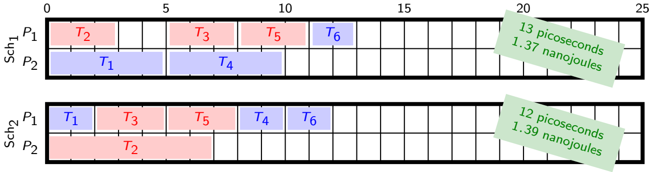
energy	
idle	10 Watt
in use	90 Watts

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

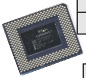
Compute $D\times(C\times(A+B))+(A+B)+(C\times D)$ using two processors:

P_1 (fast):

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

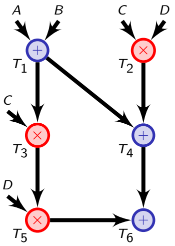
energy	
idle	10 Watt
in use	90 Watts

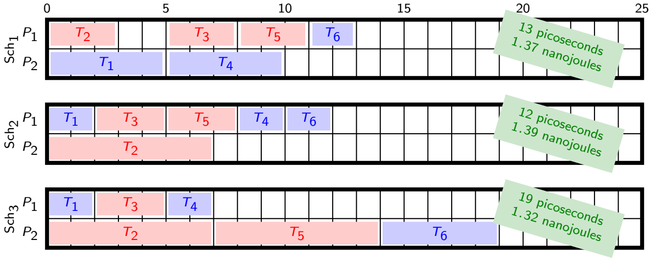
P_2 (slow):



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

energy	
idle	20 Watts
in use	30 Watts

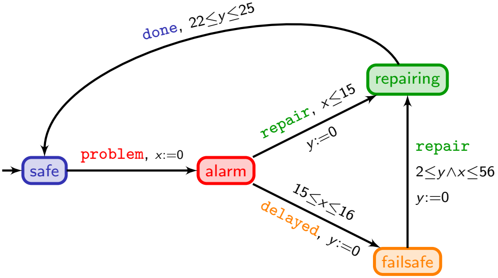




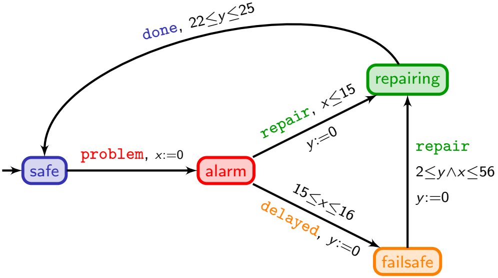
[BFLM10] Bouyer, Fahrenberg, Larsen, Markey. Quantitative Analysis of Real-Time Systems using Priced Timed Automata *(Communication of the ACM)*.

Outline

The model of timed automata



The model of timed automata



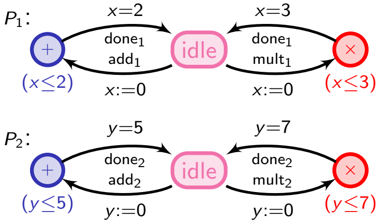
	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	$\cdots$
y	0		23		23		38.6		0	

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

Modelling the task graph scheduling problem

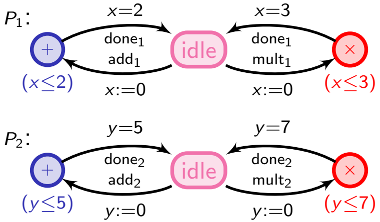
Modelling the task graph scheduling problem

- Processors

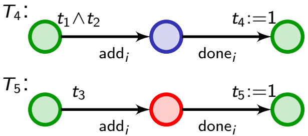


Modelling the task graph scheduling problem

- Processors

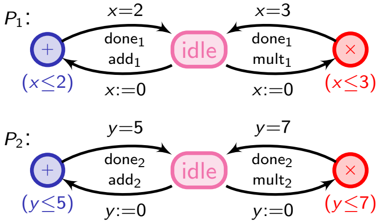


- Tasks

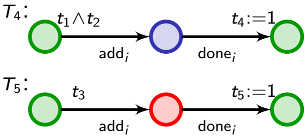


Modelling the task graph scheduling problem

- Processors



- Tasks

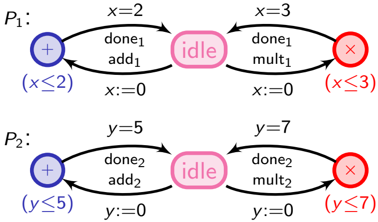


$\rightsquigarrow$ build the synchronized product of all these automata

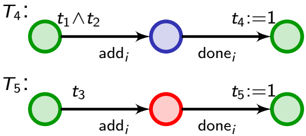
$$(P_1 \parallel P_2) \parallel_s (T_1 \parallel T_2 \parallel \cdots \parallel T_6)$$

Modelling the task graph scheduling problem

- Processors



- Tasks



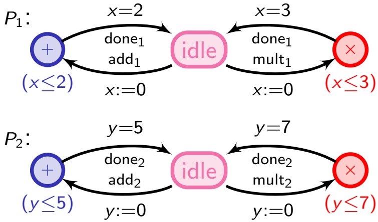
$\rightsquigarrow$ build the synchronized product of all these automata

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

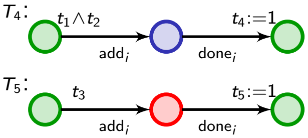
A schedule: a path in the global system which reaches $t_1 \wedge \cdots \wedge t_6$

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?
- ...

Discrete-time semantics

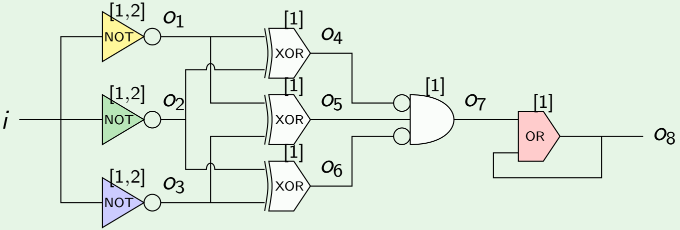
...because computers are digital!

[Alur91] Alur. Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

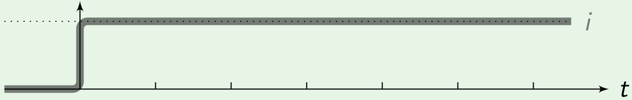
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

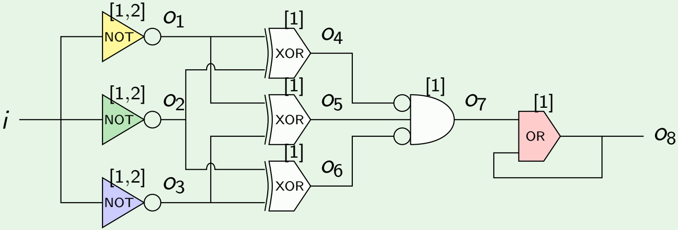


[Alur91] Alur. Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

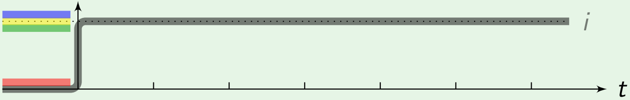
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

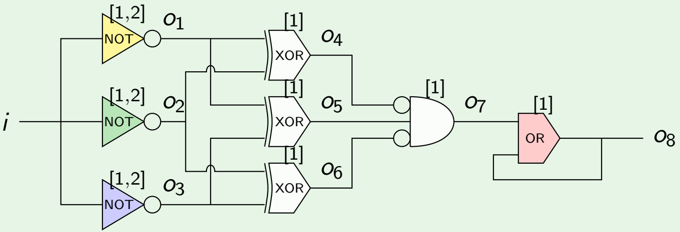


[Alur91] Alur. Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

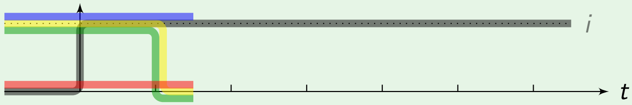
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

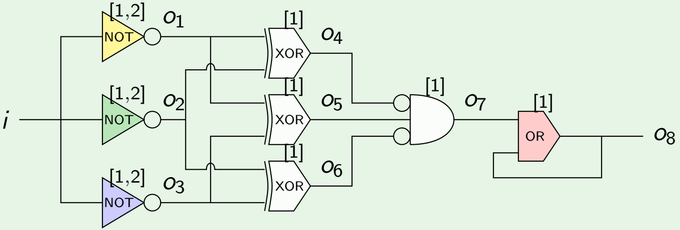


[Alur91] Alur. Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

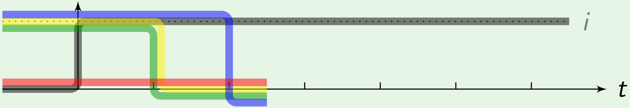
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

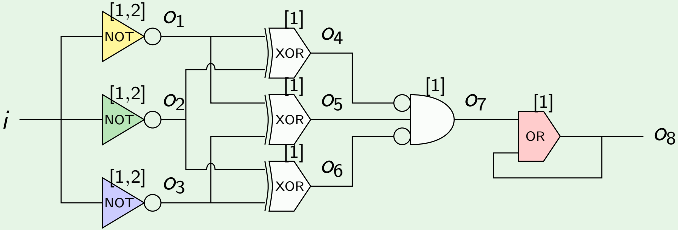


[Alur91] Alur. Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

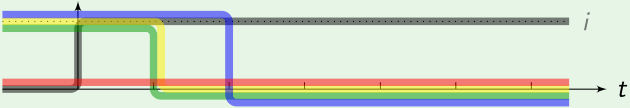
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under discrete-time, the output is always 0:

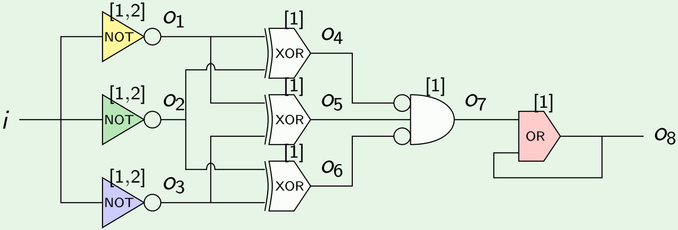


[Alur91] Alur. Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

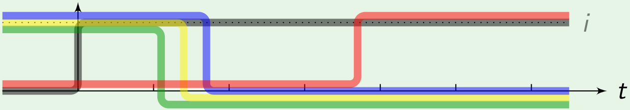
Discrete-time semantics

...because computers are digital!

Example [Alur91]



- under continuous-time, the output can be 1:

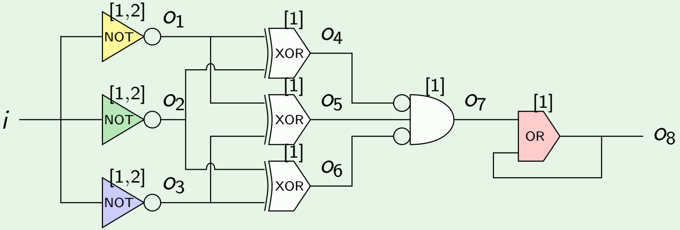


[Alur91] Alur. Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

Discrete-time semantics

...because computers are digital!

Example [Alur91]



Finding the correct granularity (if one exists) is hard!

[Alur91] Alur. Techniques for automatic verification of real-time systems. *PhD thesis*, 1991.

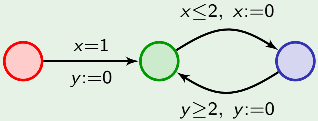
Continuous-time semantics

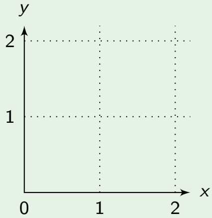
...real-time models for real-time systems!

Continuous-time semantics

...real-time models for real-time systems!

Example

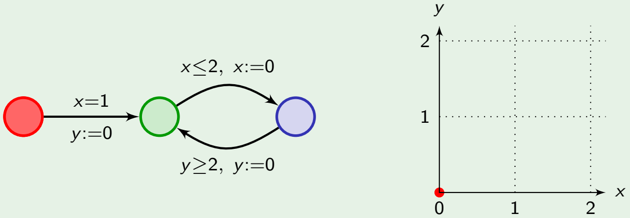




Continuous-time semantics

...real-time models for real-time systems!

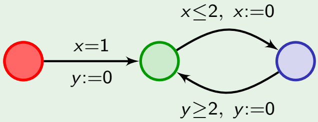
Example

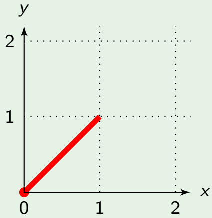


Continuous-time semantics

...real-time models for real-time systems!

Example

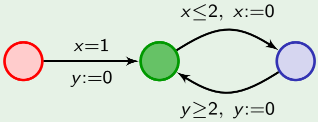


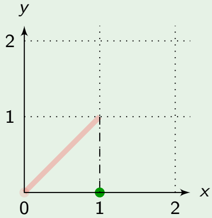


Continuous-time semantics

...real-time models for real-time systems!

Example

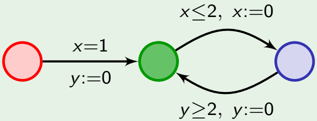


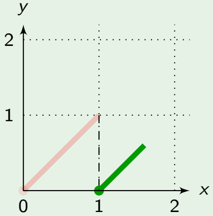


Continuous-time semantics

...real-time models for real-time systems!

Example

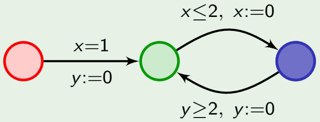


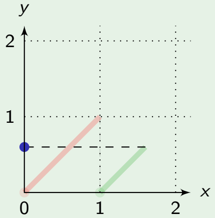


Continuous-time semantics

...real-time models for real-time systems!

Example

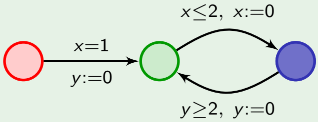


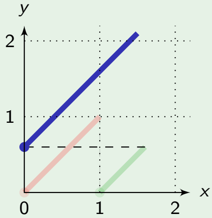


Continuous-time semantics

...real-time models for real-time systems!

Example

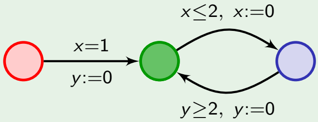


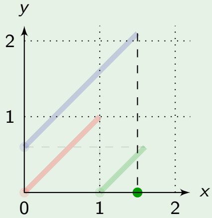


Continuous-time semantics

...real-time models for real-time systems!

Example

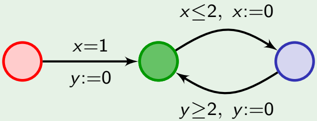


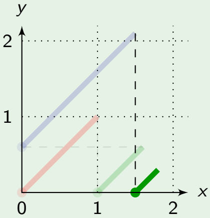


Continuous-time semantics

...real-time models for real-time systems!

Example

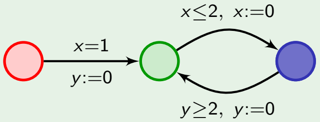


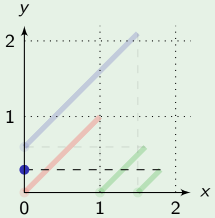


Continuous-time semantics

...real-time models for real-time systems!

Example

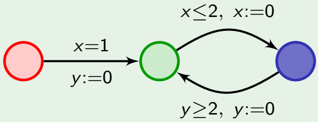


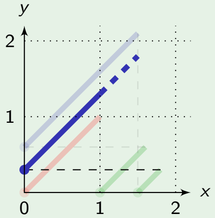


Continuous-time semantics

...real-time models for real-time systems!

Example

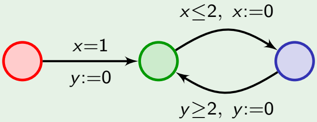


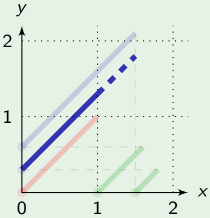


Continuous-time semantics

...real-time models for real-time systems!

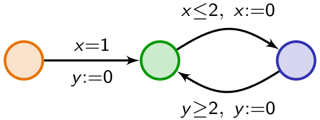
Example

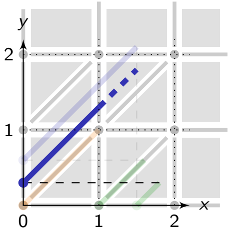




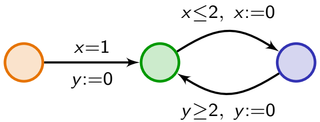
We will focus on the continuous-time semantics

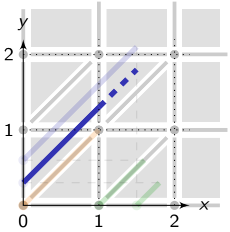
Analyzing timed automata





Analyzing timed automata





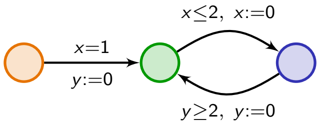
Theorem [AD94]

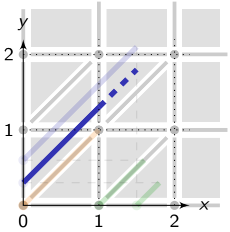
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction

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

Analyzing timed automata





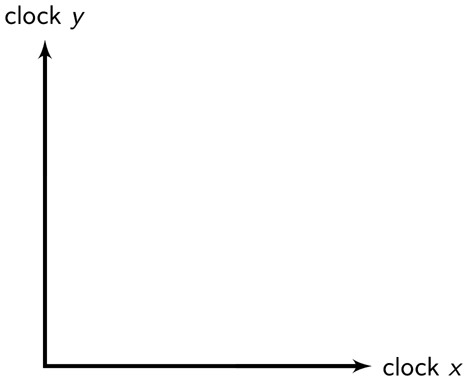
Theorem [AD94]

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

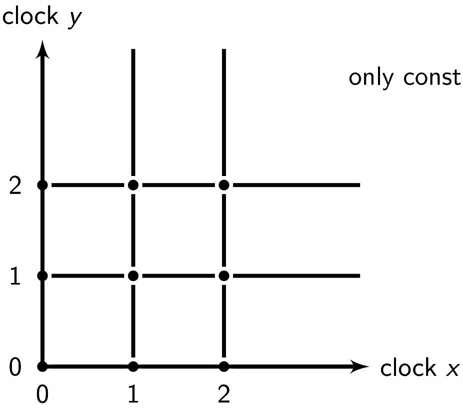
- Technical tool: region abstraction
- Efficient symbolic technics based on zones, implemented in tools

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

Technical tool: Region abstraction



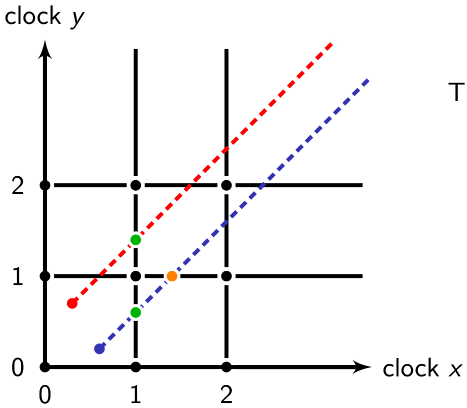
Technical tool: Region abstraction



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

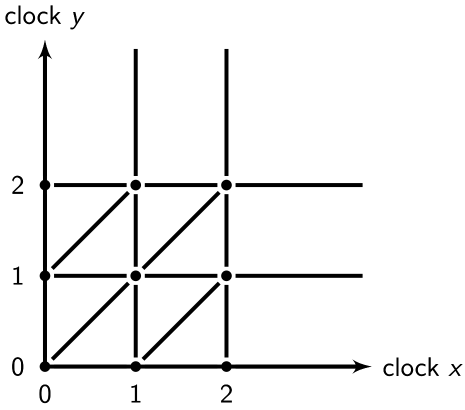


The path 

- can be fired from
- cannot be fired from

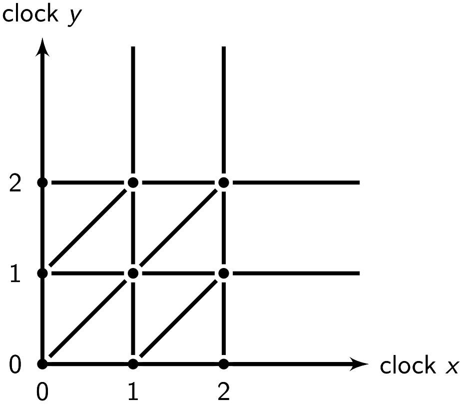
- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

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

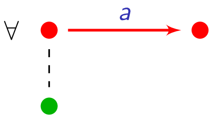
$\rightsquigarrow$ This is a finite time-abstract bisimulation!

Time-abstract bisimulation

This is a relation between • and • such that:

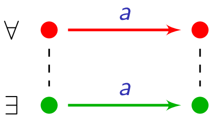
Time-abstract bisimulation

This is a relation between • and • such that:



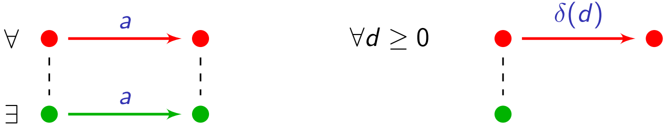
Time-abstract bisimulation

This is a relation between • and • such that:



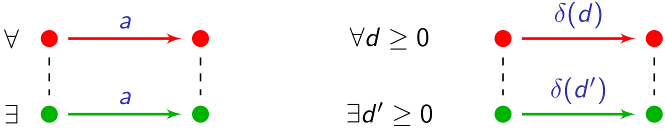
Time-abstract bisimulation

This is a relation between • and • such that:



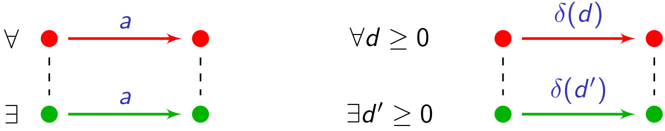
Time-abstract bisimulation

This is a relation between • and • such that:



Time-abstract bisimulation

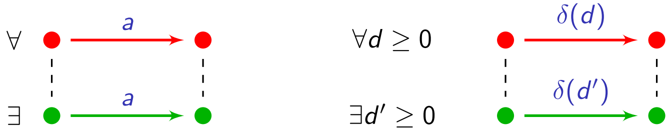
This is a relation between • and • such that:



... and vice-versa (swap • and •).

Time-abstract bisimulation

This is a relation between • and • such that:



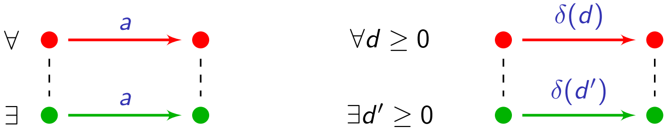
... and vice-versa (swap • and •).

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:



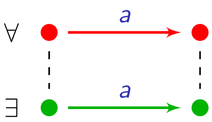
... and vice-versa (swap • and •).

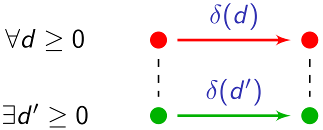
Consequence

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

Time-abstract bisimulation

This is a relation between • and • such that:





... and vice-versa (swap • and •).

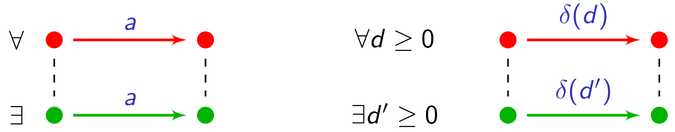
Consequence

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

$\forall v_1' \in R_1$

Time-abstract bisimulation

This is a relation between • and • such that:

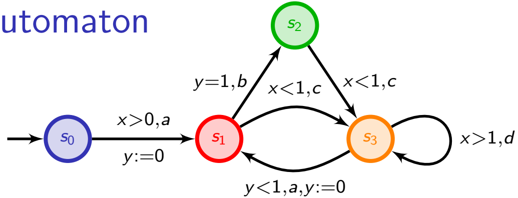


... and vice-versa (swap • and •).

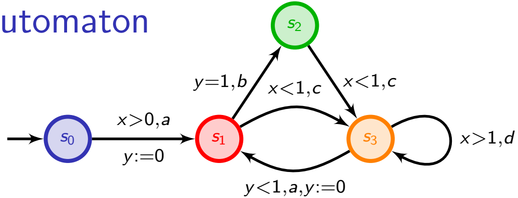
Consequence

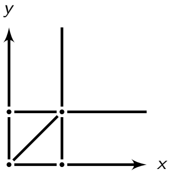
$$
\begin{tikzcd}
\forall & (\ell_1, v_1) \arrow[r, "d_1{,}a_1"] \arrow[d] & (\ell_2, v_2) \arrow[r, "d_2{,}a_2"] \arrow[d] & (\ell_3, v_3) \arrow[r, "d_3{,}a_3"] \arrow[d] & \cdots & \\
 & (\ell_1, R_1) \arrow[r, "a_1"] \arrow[d] & (\ell_2, R_2) \arrow[r, "a_2"] \arrow[d] & (\ell_3, R_3) \arrow[r, "a_3"] \arrow[d] & \cdots & \text{with } v_i \in R_i \\
\forall v'_1 \in R_1 \ \exists & (\ell_1, v'_1) \arrow[r, "d'_1{,}a_1"] & (\ell_2, v'_2) \arrow[r, "d'_2{,}a_2"] & (\ell_3, v'_3) \arrow[r, "d'_3{,}a_3"] & \cdots & \text{with } v'_i \in R_i
\end{tikzcd}
$$

The region automaton

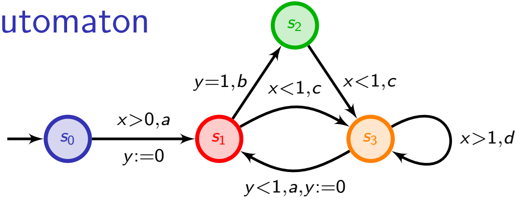


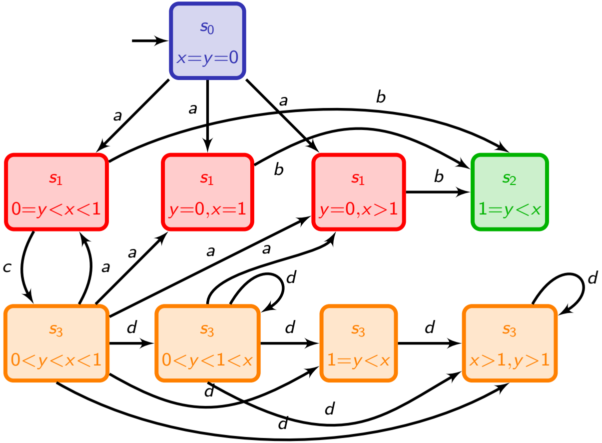
The region automaton

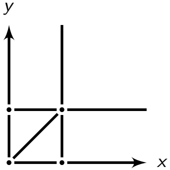




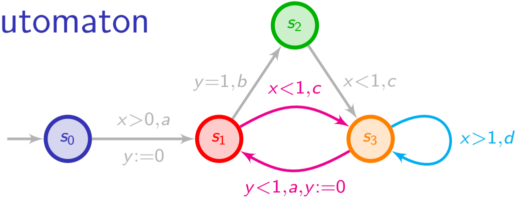
The region automaton

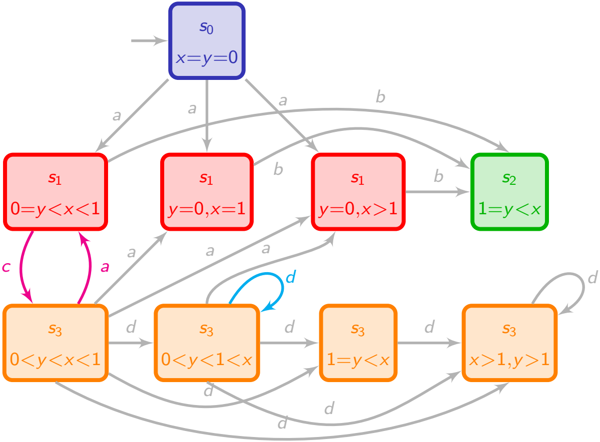


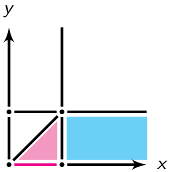




The region automaton

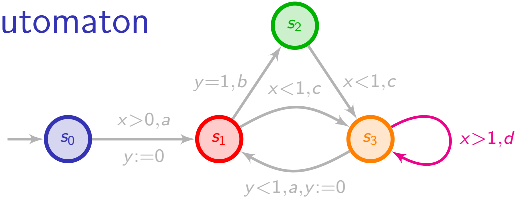


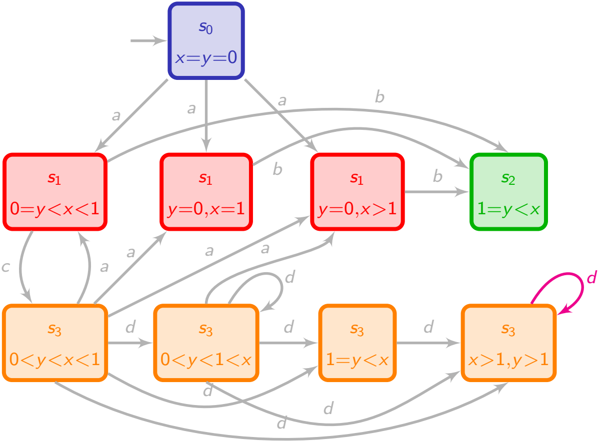


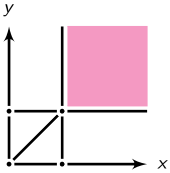


Zeno cycles

The region automaton







Cycles with non-Zeno behaviours

What about the practice?

- the region automaton is never computed
- instead, symbolic computations are performed

What about the practice?

- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:

Finite representation of infinite sets of configurations

What about the practice?

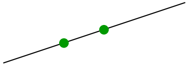
- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:

Finite representation of infinite sets of configurations

- in the plane, a line represented by two points.



What about the practice?

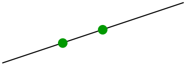
- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:

Finite representation of infinite sets of configurations

  - in the plane, a line
    represented by two points.



  - set of words *aa*, *aaaa*, *aaaaaa*...
    represented by a rational expression $aa(aa)^*$

What about the practice?

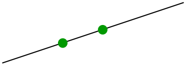
- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

- Need of a symbolic representation:

Finite representation of infinite sets of configurations

  - in the plane, a line
    represented by two points.



  - set of words aa, $aaaa$, $aaaaaa$...
    represented by a rational expression $aa(aa)^*$
  - set of integers, represented using semi-linear sets

What about the practice?

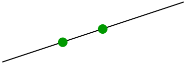
- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

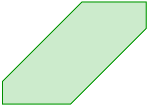
- Need of a symbolic representation:

Finite representation of infinite sets of configurations

  - in the plane, a line
    represented by two points.



  - set of words *aa*, *aaaa*, *aaaaaa*...
    represented by a rational expression $aa(aa)^*$
  - set of integers, represented using semi-linear sets
  - sets of constraints, polyhedra, zones, regions



What about the practice?

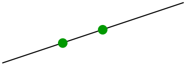
- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

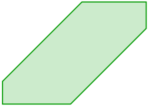
- Need of a symbolic representation:

Finite representation of infinite sets of configurations

  - in the plane, a line
    represented by two points.



  - set of words *aa*, *aaaa*, *aaaaaa*...
    represented by a rational expression $aa(aa)^*$
  - set of integers, represented using semi-linear sets
  - sets of constraints, polyhedra, zones, regions
  - BDDs, DBMs (see later), CDDs, etc...



What about the practice?

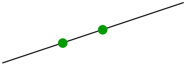
- the region automaton is never computed
- instead, symbolic computations are performed

What do we need?

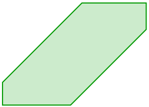
- Need of a symbolic representation:

  Finite representation of infinite sets of configurations

  - in the plane, a line
    represented by two points.



  - set of words *aa*, *aaaa*, *aaaaaa*...
    represented by a rational expression $aa(aa)^*$
  - set of integers, represented using semi-linear sets
  - sets of constraints, polyhedra, zones, regions
  - BDDs, DBMs (see later), CDDs, etc...

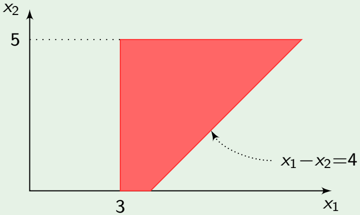


- Need of abstractions, heuristics, etc...

Zones: A symbolic representation for timed systems

Example of a zone and its DBM representation

$$Z \;=\; (x_1 \geq 3) \;\wedge\; (x_2 \leq 5) \;\wedge\; (x_1 - x_2 \leq 4)$$



$$\begin{array}{c c} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} \infty & -3 & \infty \\ \infty & \infty & 4 \\ 5 & \infty & \infty \end{pmatrix} \end{array}$$

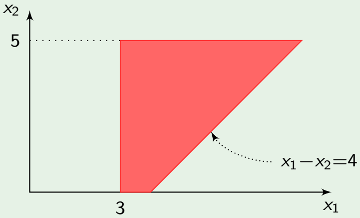
DBM: Difference Bound Matrice [BM83,Dill89]

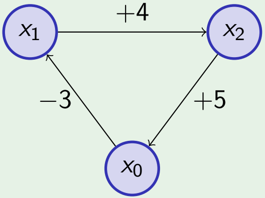
[BM83] Berthomieu, Menasche. An enumerative approach for analyzing time Petri nets *World Comupter Congress*.
[Dill89] Dill. Timing assumptions and verification of finite-state concurrent systems *(Automatic Verification Methods for Finite State Systems)*.

Zones: A symbolic representation for timed systems

Example of a zone and its DBM representation

$$Z \;=\; (x_1 \geq 3) \;\wedge\; (x_2 \leq 5) \;\wedge\; (x_1 - x_2 \leq 4)$$





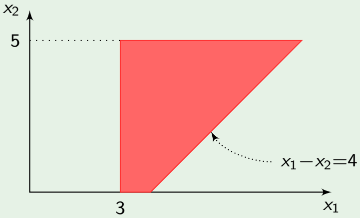
DBM: Difference Bound Matrice [BM83,Dill89]

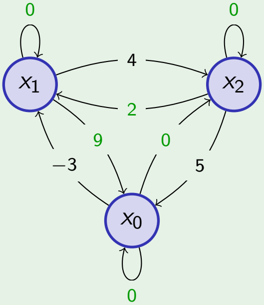
[BM83] Berthomieu, Menasche. An enumerative approach for analyzing time Petri nets *World Comupter Congress*.
[Dill89] Dill. Timing assumptions and verification of finite-state concurrent systems *(Automatic Verification Methods for Finite State Systems)*.

Zones: A symbolic representation for timed systems

Example of a zone and its DBM representation

$$Z = (x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$





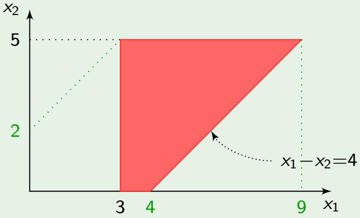
DBM: Difference Bound Matrice [BM83,Dill89]

[BM83] Berthomieu, Menasche. An enumerative approach for analyzing time Petri nets *World Comupter Congress*.
[Dill89] Dill. Timing assumptions and verification of finite-state concurrent systems *(Automatic Verification Methods for Finite State Systems)*.

Zones: A symbolic representation for timed systems

Example of a zone and its DBM representation

$$Z = (x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$



$$\begin{array}{c|ccc} & x_0 & x_1 & x_2 \\ \hline x_0 & 0 & -3 & 0 \\ x_1 & 9 & 0 & 4 \\ x_2 & 5 & 2 & 0 \end{array}$$

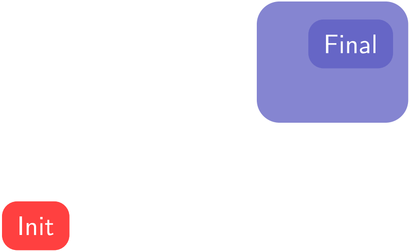
"normal form"

DBM: Difference Bound Matrice [BM83,Dill89]

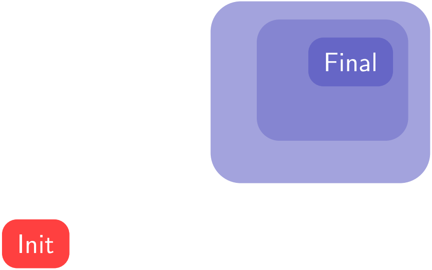
[BM83] Berthomieu, Menasche. An enumerative approach for analyzing time Petri nets *World Comupter Congress*.
[Dill89] Dill. Timing assumptions and verification of finite-state concurrent systems *(Automatic Verification Methods for Finite State Systems)*.

Backward computation

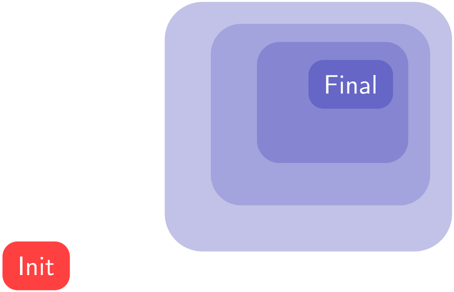
Backward computation



Backward computation

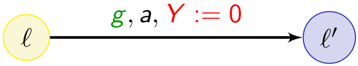


Backward computation



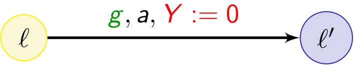
Backward computation

Note on the backward analysis of timed automata

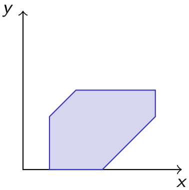


$\overleftarrow{[Y \leftarrow 0]^{-1}(Z \cap (Y = 0)) \cap g}$ $\qquad\qquad Z$

Note on the backward analysis of timed automata

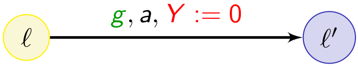


$\overleftarrow{[Y \leftarrow 0]^{-1}(Z \cap (Y = 0)) \cap g}$ Z

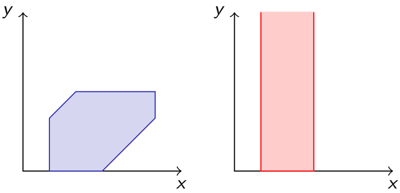


Z

Note on the backward analysis of timed automata



$$\overleftarrow{[Y \leftarrow 0]^{-1}(Z \cap (Y = 0)) \cap g} \qquad Z$$

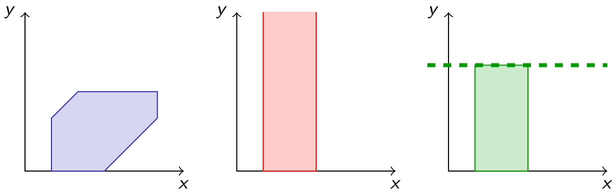


Z $\qquad$ $[y{\leftarrow}0]^{-1}(Z{\cap}(y{=}0))$

Note on the backward analysis of timed automata

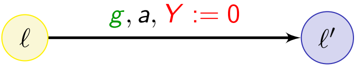
$$\ell \xrightarrow{g,\, a,\, Y := 0} \ell'$$

$$\overleftarrow{[Y \leftarrow 0]^{-1}(Z \cap (Y = 0)) \cap g} \qquad\qquad Z$$

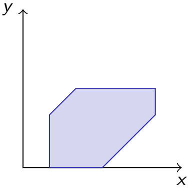


Z $\qquad$ $[y \leftarrow 0]^{-1}(Z \cap (y = 0))$

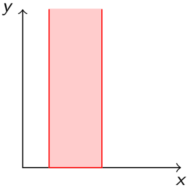
Note on the backward analysis of timed automata



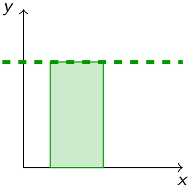
$\overleftarrow{[Y \leftarrow 0]^{-1}(Z \cap (Y = 0)) \cap g}$ $\quad$ Z

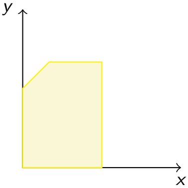


Z



$[y \leftarrow 0]^{-1}(Z \cap (y=0))$





$\overleftarrow{[y \leftarrow 0]^{-1}(Z \cap (y=0)) \cap g}$

Note on the backward analysis (cont.)

☺ All previous operations can be computed using DBMs!

Note on the backward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints

Note on the backward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- inverse reset w.r.t y: relax constraints on y (on a DBM on normal form)

Note on the backward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- inverse reset w.r.t y: relax constraints on y (on a DBM on normal form)
- past: relax lower bounds (on a DBM on normal form)

Note on the backward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- inverse reset w.r.t y: relax constraints on y (on a DBM on normal form)
- past: relax lower bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

Note on the backward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- inverse reset w.r.t y: relax constraints on y (on a DBM on normal form)
- past: relax lower bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

The backward computation terminates

Because of the bisimulation property of the region abstraction:

"Every set of valuations which is computed along the backward computation is a finite union of regions"

Note on the backward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- inverse reset w.r.t y: relax constraints on y (on a DBM on normal form)
- past: relax lower bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

The backward computation terminates

Because of the bisimulation property of the region abstraction:

"Every set of valuations which is computed along the backward computation is a finite union of regions"

Let R be a region. Assume:

- $v \in \overleftarrow{R}$ (for ex. $v + t \in R$)
- $v' \equiv_{\text{reg.}} v$

There exists t' s.t. $v' + t' \equiv_{\text{reg.}} v + t$, which implies that $v' + t' \in R$ and thus $v' \in \overleftarrow{R}$.

Note on the backward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- inverse reset w.r.t y: relax constraints on y (on a DBM on normal form)
- past: relax lower bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

The backward computation terminates

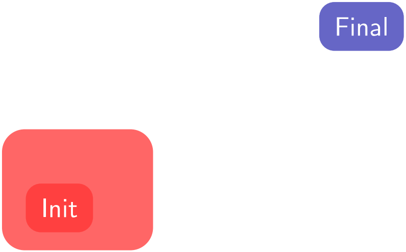
Because of the bisimulation property of the region abstraction:

"Every set of valuations which is computed along the backward computation is a finite union of regions"

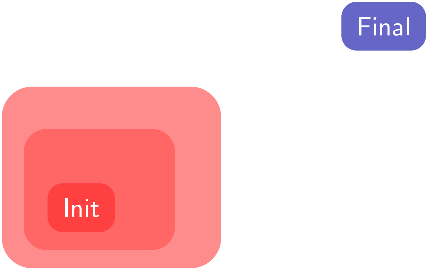
However the backward computation is not appropriate to manipulate other variables (think for instance of assignment $i := j.k + l$)

Forward computation

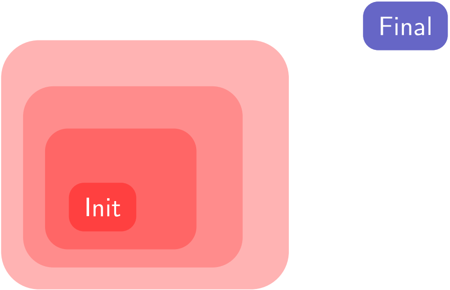
Forward computation



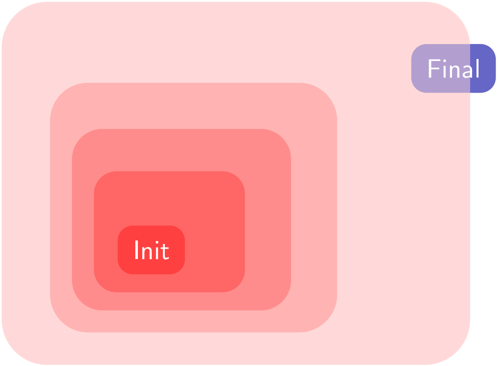
Forward computation



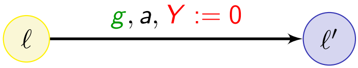
Forward computation



Forward computation

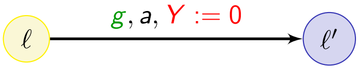


Forward analysis of timed automata

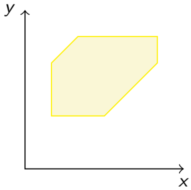


Z $\qquad\qquad$ $[Y \leftarrow 0](\overrightarrow{Z} \cap g)$

Forward analysis of timed automata



Z $\qquad\qquad$ $[Y \leftarrow 0](\overrightarrow{Z} \cap g)$

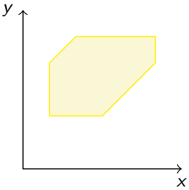


Z

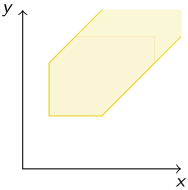
Forward analysis of timed automata

$$\ell \xrightarrow{g,\, a,\, Y := 0} \ell'$$

Z $\qquad$ $[Y \leftarrow 0](\overrightarrow{Z} \cap g)$

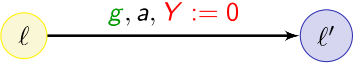


Z

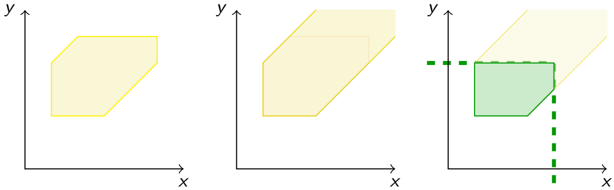


$\overrightarrow{Z}$

Forward analysis of timed automata

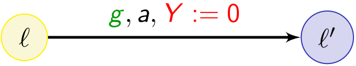


Z $\qquad$ $[Y \leftarrow 0](\overrightarrow{Z} \cap g)$

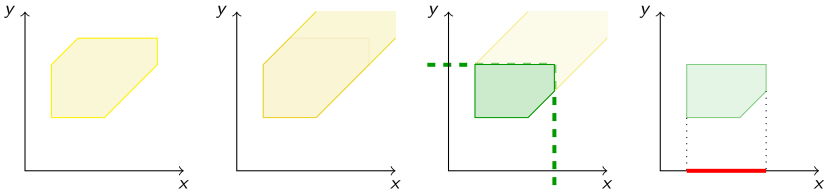


Z $\qquad$ $\overrightarrow{Z}$ $\qquad$ $\overrightarrow{Z} \cap g$

Forward analysis of timed automata



Z $\quad\quad\quad\quad$ $[Y \leftarrow 0](\overrightarrow{Z} \cap g)$



Z $\quad$ $\overrightarrow{Z}$ $\quad$ $\overrightarrow{Z} \cap g$ $\quad$ $[y \leftarrow 0](\overrightarrow{Z} \cap g)$

Note on the forward analysis (cont.)

☺ All previous operations can be computed using DBMs!

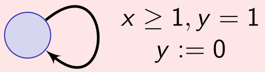
- intersection: take the minimum of the two constraints
- reset w.r.t y: set constraint if y to 0 (on a DBM on normal form)
- future: relax upper bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

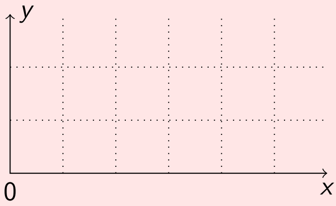
Note on the forward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- reset w.r.t y: set constraint if y to 0 (on a DBM on normal form)
- future: relax upper bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

☹ The forward computation may not terminate...



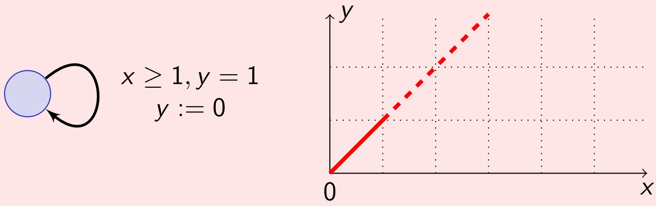


Note on the forward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- reset w.r.t y: set constraint if y to 0 (on a DBM on normal form)
- future: relax upper bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

☹ The forward computation may not terminate...

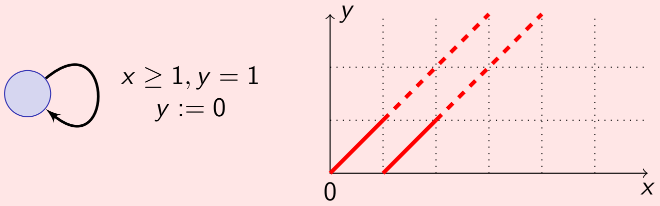


Note on the forward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- reset w.r.t y: set constraint if y to 0 (on a DBM on normal form)
- future: relax upper bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

☹ The forward computation may not terminate...

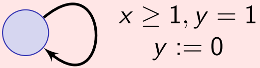


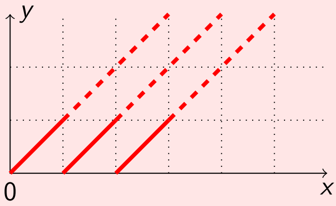
Note on the forward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- reset w.r.t y: set constraint if y to 0 (on a DBM on normal form)
- future: relax upper bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

☹ The forward computation may not terminate...



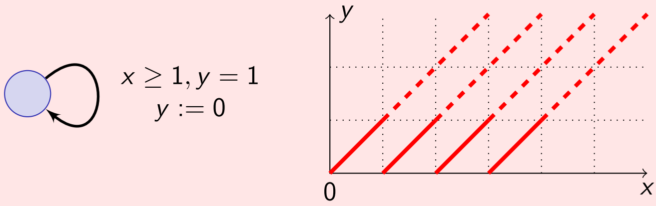


Note on the forward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- reset w.r.t y: set constraint if y to 0 (on a DBM on normal form)
- future: relax upper bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

☹ The forward computation may not terminate...

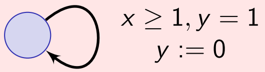


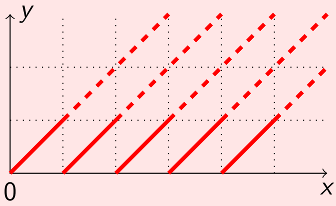
Note on the forward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- reset w.r.t y: set constraint if y to 0 (on a DBM on normal form)
- future: relax upper bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

☹ The forward computation may not terminate...



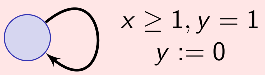


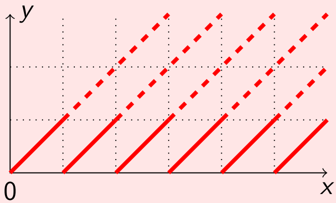
Note on the forward analysis (cont.)

☺ All previous operations can be computed using DBMs!

- intersection: take the minimum of the two constraints
- reset w.r.t y: set constraint if y to 0 (on a DBM on normal form)
- future: relax upper bounds (on a DBM on normal form)
- emptiness: check whether there is a negative cycle

☹ The forward computation may not terminate...





$\rightsquigarrow$ an infinite number of steps...

Forward reachability algorithm

Parameters: Abstraction abs and inclusion test $\preceq$

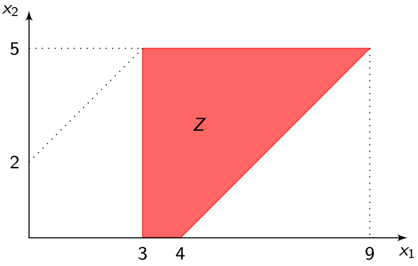
Forward reachability algorithm

Parameters: Abstraction abs and inclusion test $\preceq$

- $\texttt{Passed} \leftarrow \emptyset$ and $\texttt{Waiting} \leftarrow \{(\ell_0, Z_0)\}$
- While $\texttt{Waiting} \neq \emptyset$
    - select (ℓ, Z) from Waiting
    - If ℓ is final, then return "Reachable!"
    - If forall $(\ell, Z') \in \texttt{Passed}$, $Z \not\preceq Z'$, then add $\mathsf{abs}(\ell, Z)$ to Passed and add $\mathsf{Post}(\mathsf{abs}(\ell, Z))$ to Waiting
- Return "Not reachable!"

Standard solution: the extrapolation operator

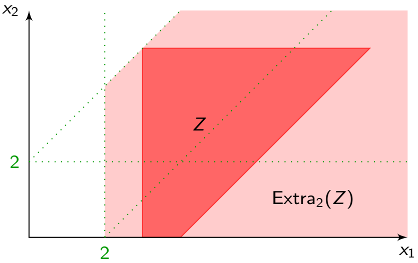
$\mathsf{Extra}_2(Z)$: "the smallest zone containing Z that is defined only with constants no more than 2"



$$\begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix}$$

Standard solution: the extrapolation operator

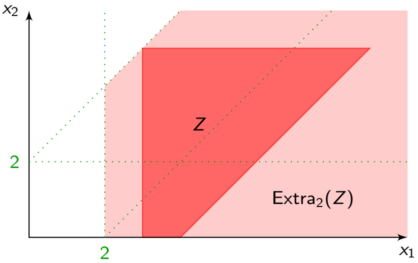
$\mathsf{Extra}_2(Z)$: "the smallest zone containing Z that is defined only with constants no more than 2"



$$\begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix} \xRightarrow{\mathsf{Extra}_2} \begin{pmatrix} 0 & -2 & 0 \\ \infty & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix}$$

Standard solution: the extrapolation operator

$\mathsf{Extra}_2(Z)$: "the smallest zone containing Z that is defined only with constants no more than 2"



$$\begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix} \xRightarrow{\mathsf{Extra}_2} \begin{pmatrix} 0 & -2 & 0 \\ \infty & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix}$$

$\rightsquigarrow$ The extrapolation operator ensures termination of the computation!

The extrapolation: correctness

Theorem [Bou04]

The forward algorithm with $\mathsf{abs} = \mathsf{Extra}_M$ and $\preceq \, = \, \subseteq$ is correct for timed automata.

[Bou04] Bouyer. Forward analysis of updatable timed automata *(Formal Methods in System Design)*.

The extrapolation: correctness

Theorem [Bou04]

The forward algorithm with $\mathsf{abs} = \mathsf{Extra}_M$ and $\preceq\, = \,\subseteq$ is correct for timed automata.

- the extrapolation operator can be made coarser:
  - use local extrapolation constants [BBFL03];
  - distinguish between lower- and upper-bounded contraints [BBLP04,BBLP06]
  - use non-convex (but optimal!) abstractions [HSW12]
  - compute constants dynamically [HSW13]

[Bou04] Bouyer. Forward analysis of updatable timed automata *(Formal Methods in System Design)*.
[BBFL03] Behrmann, Bouyer, Fleury, Larsen. Static Guard Analysis in Timed Automata Verification *(TACAS'03)*.
[BBLP04] Behrmann, Bouyer, Larsen, Pelánek. Lower and Upper Bounds in Zone Based Abstractions of Timed Automata *(TACAS'04)*.
[BBLP06] Behrmann, Bouyer, Larsen, Pelánek. Lower and Upper Bounds in Zone-Based Abstractions of Timed Automata *(International Journal on Software Tools for Technology Transfer)*.
[HSW12] Herbreteau, Srivathsan, Walukiewicz. Better abstractions for timed automata *(LICS'12)*.
[HSW13] Herbreteau, Srivathsan, Walukiewicz. Lazy abstractions for timed automata *(CAV'13)*.

Alternative: Improved inclusion test

Develop an inclusion test $\sqsubseteq_{\mathsf{abs}}$ such that:

$$Z \sqsubseteq_{\mathsf{abs}} Z' \quad \text{iff} \quad Z \subseteq \mathsf{abs}(Z')$$

[HSW12] Herbreteau, Srivathsan, Walukiewicz. Better abstractions for timed automata *(LICS'12)*.

Alternative: Improved inclusion test

Develop an inclusion test $\sqsubseteq_{\mathsf{abs}}$ such that:

$$Z \sqsubseteq_{\mathsf{abs}} Z' \quad \text{iff} \quad Z \subseteq \mathsf{abs}(Z')$$

- Advantage: only store zones (without abstractions)!

[HSW12] Herbreteau, Srivathsan, Walukiewicz. Better abstractions for timed automata *(LICS'12)*.

Alternative: Improved inclusion test

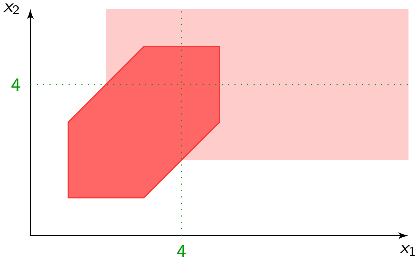
Develop an inclusion test $\sqsubseteq_{\text{abs}}$ such that:

$$Z \sqsubseteq_{\text{abs}} Z' \quad \text{iff} \quad Z \subseteq \text{abs}(Z')$$

- Advantage: only store zones (without abstractions)!
- abs_{LU}: (non-convex) abstraction w.r.t. LU-constraints

Simple case:

$M = L = U = 4$



[HSW12] Herbreteau, Srivathsan, Walukiewicz. Better abstractions for timed automata *(LICS'12)*.

Alternative: Improved inclusion test

Develop an inclusion test $\sqsubseteq_{\mathsf{abs}}$ such that:

$$Z \sqsubseteq_{\mathsf{abs}} Z' \quad \text{iff} \quad Z \subseteq \mathsf{abs}(Z')$$

- Advantage: only store zones (without abstractions)!
- abs_{LU}: (non-convex) abstraction w.r.t. LU-constraints
- Note: Optimal abstraction for LU-constrained automata!

[HSW12] Herbreteau, Srivathsan, Walukiewicz. Better abstractions for timed automata *(LICS'12)*.

Alternative: Improved inclusion test

Develop an inclusion test $\sqsubseteq_{\mathsf{abs}}$ such that:

$$Z \sqsubseteq_{\mathsf{abs}} Z' \quad \text{iff} \quad Z \subseteq \mathsf{abs}(Z')$$

- Advantage: only store zones (without abstractions)!
- abs_{LU}: (non-convex) abstraction w.r.t. LU-constraints
- Note: Optimal abstraction for LU-constrained automata!
- Result: $Z \sqsubseteq_{\mathsf{abs}_{LU}} Z'$ can be computed efficiently

[HSW12] Herbreteau, Srivathsan, Walukiewicz. Better abstractions for timed automata *(LICS'12)*.

Alternative: Improved inclusion test

Develop an inclusion test $\sqsubseteq_{\mathsf{abs}}$ such that:

$$Z \sqsubseteq_{\mathsf{abs}} Z' \quad \text{iff} \quad Z \subseteq \mathsf{abs}(Z')$$

- Advantage: only store zones (without abstractions)!
- abs_{LU}: (non-convex) abstraction w.r.t. LU-constraints
- Note: Optimal abstraction for LU-constrained automata!
- Result: $Z \sqsubseteq_{\mathsf{abs}_{LU}} Z'$ can be computed efficiently

Theorem

The forward algorithm with $\mathsf{abs} = \mathsf{Id}$ and $\preceq = \sqsubseteq_{\mathsf{abs}_{LU}}$ is correct for timed automata.

[HSW12] Herbreteau, Srivathsan, Walukiewicz. Better abstractions for timed automata *(LICS'12)*.

Tools for timed automata

- Uppaal, developed in Aalborg (Denmark) and Uppsala (Sweden) since 1995

  http://www.uppaal.org

Tools for timed automata

- Uppaal, developed in Aalborg (Denmark) and Uppsala (Sweden) since 1995

  `http://www.uppaal.org`

- Kronos, developed in Grenoble (France), no more maintained

  `http://www-verimag.imag.fr/DIST-TOOLS/TEMPO/kronos/`

Tools for timed automata

- Uppaal, developed in Aalborg (Denmark) and Uppsala (Sweden) since 1995

  `http://www.uppaal.org`

- Kronos, developed in Grenoble (France), no more maintained

  `http://www-verimag.imag.fr/DIST-TOOLS/TEMPO/kronos/`

- Many other prototypes everywhere on earth...

Tools for timed automata

- Uppaal, developed in Aalborg (Denmark) and Uppsala (Sweden) since 1995

  `http://www.uppaal.org`

- Kronos, developed in Grenoble (France), no more maintained

  `http://www-verimag.imag.fr/DIST-TOOLS/TEMPO/kronos/`

- Many other prototypes everywhere on earth...
- Our new tool TiAMo, developed by Maximilien Colange (LSV)

  `https://git.lsv.fr/colange/tiamo`

Tools for timed automata

- Uppaal, developed in Aalborg (Denmark) and Uppsala (Sweden) since 1995

  `http://www.uppaal.org`

- Kronos, developed in Grenoble (France), no more maintained

  `http://www-verimag.imag.fr/DIST-TOOLS/TEMPO/kronos/`

- Many other prototypes everywhere on earth...
- Our new tool TiAMo, developed by Maximilien Colange (LSV)

  `https://git.lsv.fr/colange/tiamo`

$\rightsquigarrow$ see description and demo later

Outline

Modelling resources in timed systems

- System resources might be relevant and even crucial information

Modelling resources in timed systems

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

Modelling resources in timed systems

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

$\rightsquigarrow$ timed automata are not powerful enough!

Modelling resources in timed systems

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

$\rightsquigarrow$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

$$\begin{array}{ll} & \text{a discrete control (the mode of the system)} \\ + & \text{continuous evolution of the variables within a mode} \end{array}$$

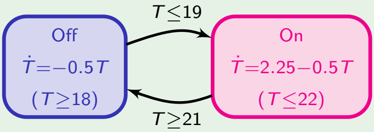
Modelling resources in timed systems

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

$\rightsquigarrow$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

The thermostat example



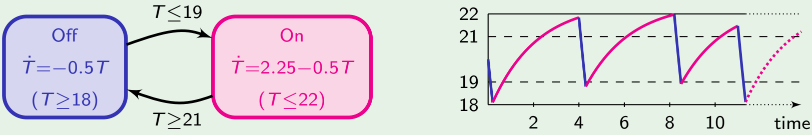
Modelling resources in timed systems

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

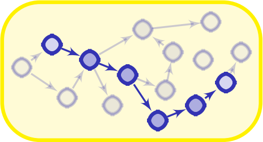
$\rightsquigarrow$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

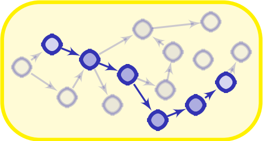
The thermostat example



Ok...

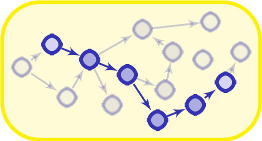


Ok...

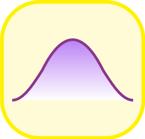


Easy...

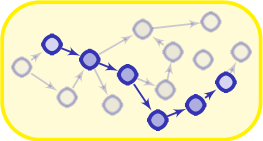
Ok...



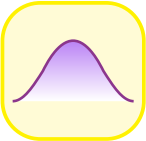
Easy...



Ok...

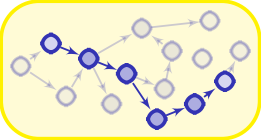


Easy...

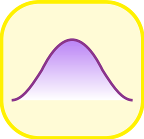


Easy...

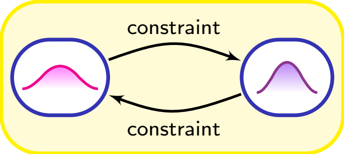
Ok... but?



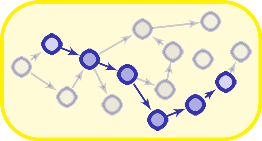
Easy...



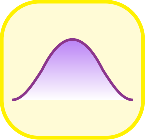
Easy...



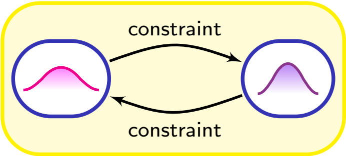
Ok... but?



Easy...



Easy...



Hard!

Modelling resources in timed systems

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

  $\rightsquigarrow$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

Theorem [HKPV95]

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

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

Modelling resources in timed systems

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

  $\rightsquigarrow$ timed automata are not powerful enough!

- A possible solution: use hybrid automata

Theorem [HKPV95]

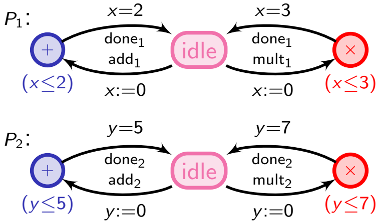
The reachability problem is undecidable in hybrid automata. Even for the simplest, the so-called stopwatch automata (clocks can be stopped).

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

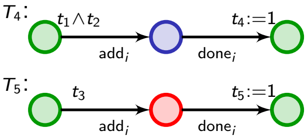
[HKPV95] Henzinger, Kopke, Puri, Varaiya. What's decidable wbout hybrid automata? *(SToC'95)*.
[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata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

Modelling the task graph scheduling problem

- Processors

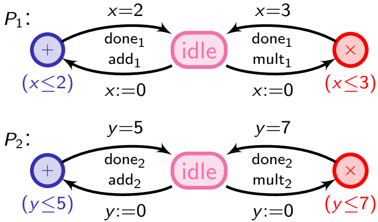


- Tasks

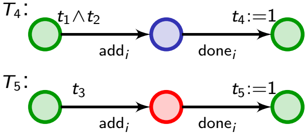


Modelling the task graph scheduling problem

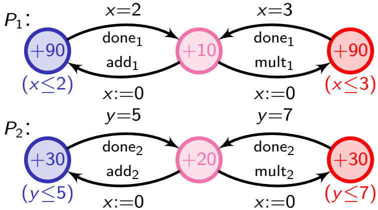
- Processors



- Tasks

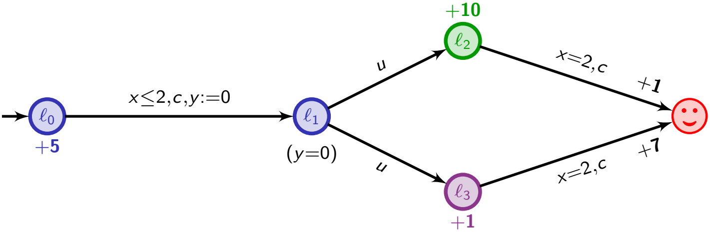


- Modelling energy



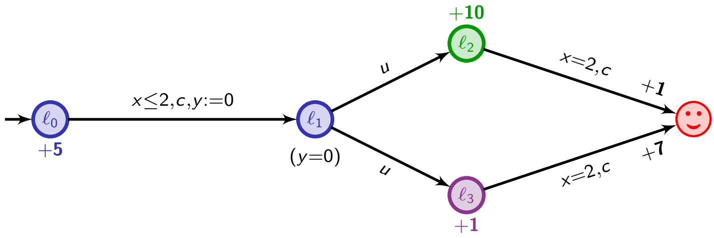
A good schedule is a path in the product automaton with a low cost

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



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

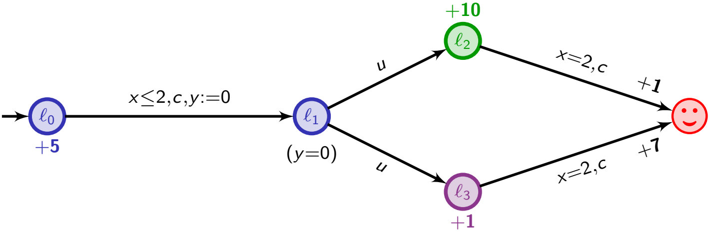
Weighted/priced timed automata [ALP01,BFH+01]



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

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

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

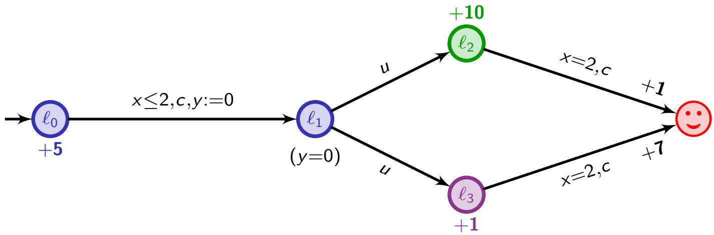


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

cost :

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

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

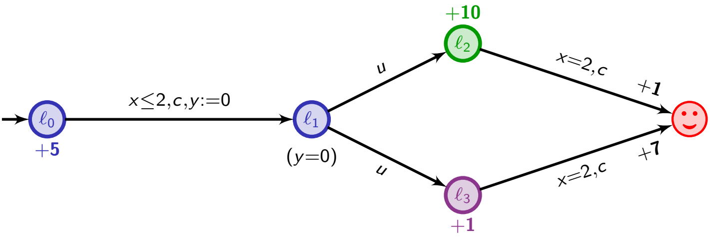


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

cost : 6.5

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

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

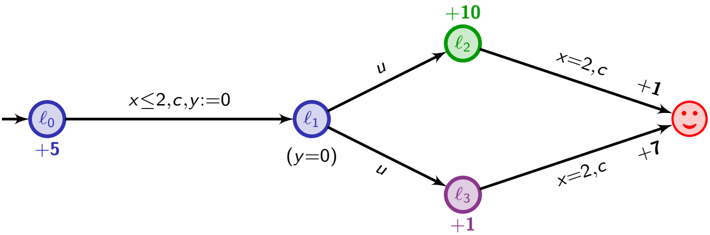


$$
\begin{array}{ccccccccccccc}
 & \ell_0 & \xrightarrow{1.3} & \ell_0 & \xrightarrow{c} & \ell_1 & \xrightarrow{u} & \ell_3 & \xrightarrow{0.7} & \ell_3 & \xrightarrow{c} & ☺ \\
x & 0 & & 1.3 & & 1.3 & & 1.3 & & 2 & & \\
y & 0 & & 1.3 & & 0 & & 0 & & 0.7 & &
\end{array}
$$

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]

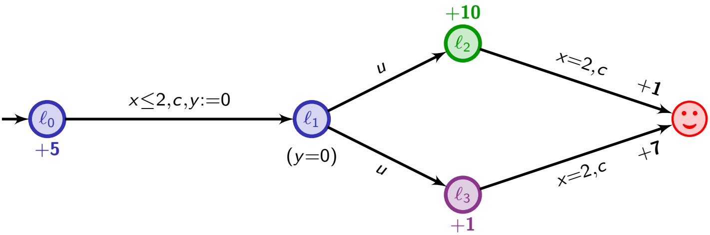


$$
\begin{array}{cccccccccccc}
 & \ell_0 & \xrightarrow{1.3} & \ell_0 & \xrightarrow{c} & \ell_1 & \xrightarrow{u} & \ell_3 & \xrightarrow{0.7} & \ell_3 & \xrightarrow{c} & ☺ \\
x & 0 & & 1.3 & & 1.3 & & 1.3 & & 2 & & \\
y & 0 & & 1.3 & & 0 & & 0 & & 0.7 & &
\end{array}
$$

cost : 6.5 + 0 + 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]

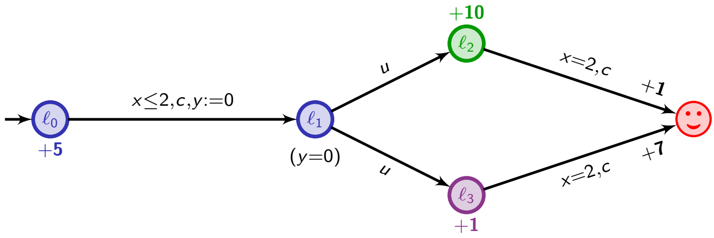


$$\begin{array}{ccccccccccccc} & \ell_0 & \xrightarrow{1.3} & \ell_0 & \xrightarrow{c} & \ell_1 & \xrightarrow{u} & \ell_3 & \xrightarrow{0.7} & \ell_3 & \xrightarrow{c} & ☺ \\ x & 0 & & 1.3 & & 1.3 & & 1.3 & & 2 & & \\ y & 0 & & 1.3 & & 0 & & 0 & & 0.7 & & \end{array}$$

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]

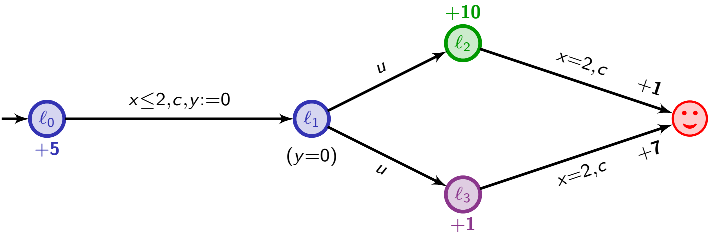


	ℓ_0	$\xrightarrow{1.3}$	ℓ_0	$\xrightarrow{c}$	ℓ_1	$\xrightarrow{u}$	ℓ_3	$\xrightarrow{0.7}$	ℓ_3	$\xrightarrow{c}$	☺
x	0		1.3		1.3		1.3		2		
y	0		1.3		0		0		0.7		

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]

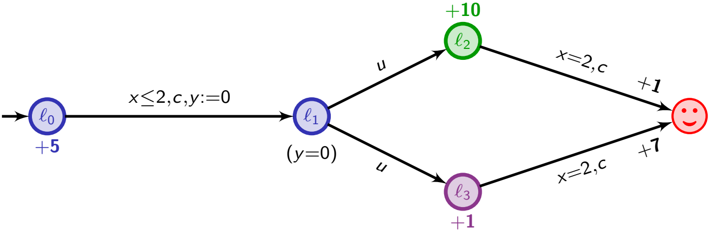


$$
\begin{array}{ccccccccccc}
 & \ell_0 & \xrightarrow{1.3} & \ell_0 & \xrightarrow{c} & \ell_1 & \xrightarrow{u} & \ell_3 & \xrightarrow{0.7} & \ell_3 & \xrightarrow{c} & ☺ \\
x & 0 & & 1.3 & & 1.3 & & 1.3 & & 2 & & \\
y & 0 & & 1.3 & & 0 & & 0 & & 0.7 & &
\end{array}
$$

cost : 6.5 + 0 + 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)*.

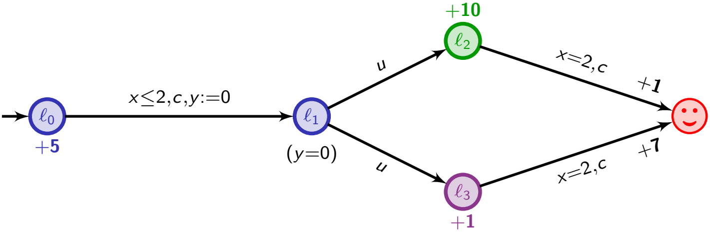
Weighted/priced timed automata [ALP01,BFH+01]



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]

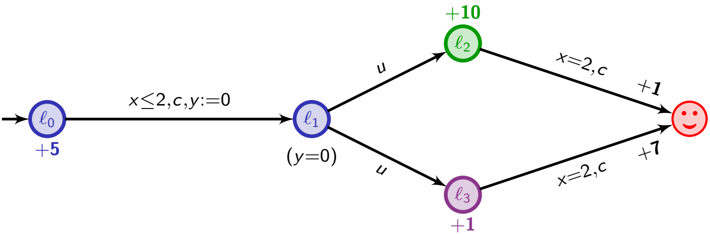


Question: what is the optimal cost for reaching 🙂?

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

[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]

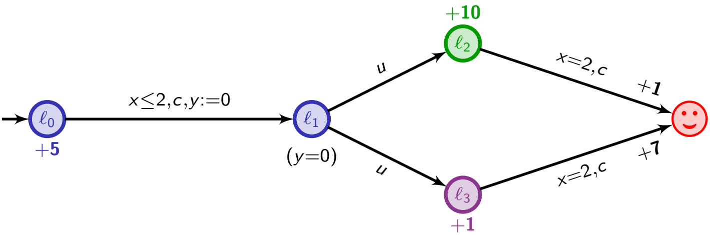


Question: what is the optimal cost for reaching 🙂?

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

[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]

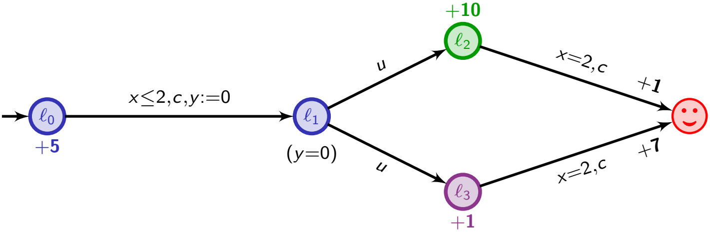


Question: what is the optimal cost for reaching ☺ ?

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

[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]

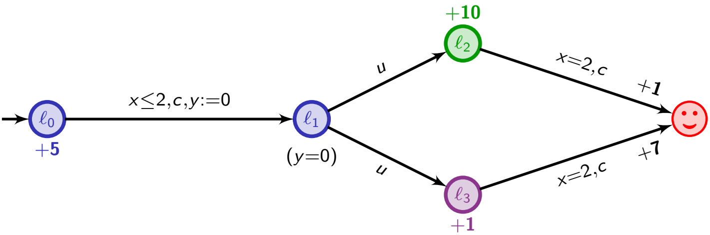


Question: what is the optimal cost for reaching ☺?

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

[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.

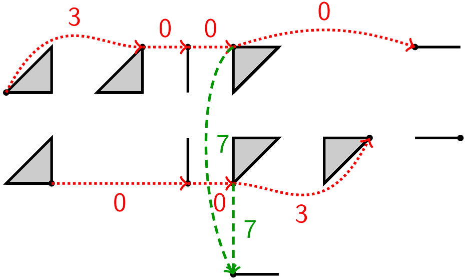
[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.

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



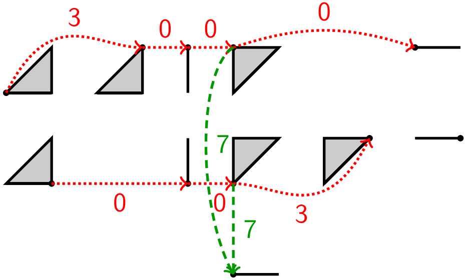
[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata *(HSCC'01)*.
[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata *(HSCC'01)*.
[BBBR07] Bouyer, Brihaye, Bruyère, Raskin. On the optimal reachability problem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

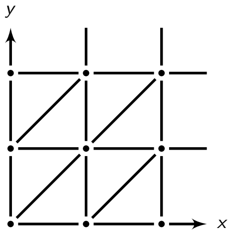
- Technical tool: a refinement of the regions, the corner-point abstraction



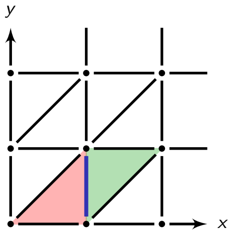
- Symbolic technics based on priced zones

[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)*.

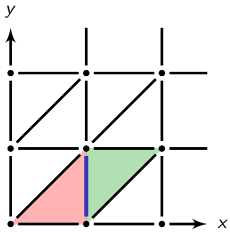
Technical tool: the corner-point abstraction



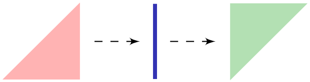
Technical tool: the corner-point abstraction



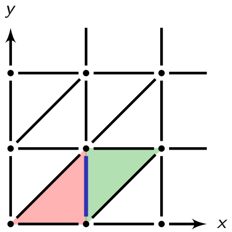
Technical tool: the corner-point abstraction



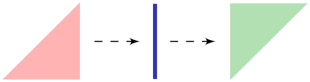
Abstract time successors:



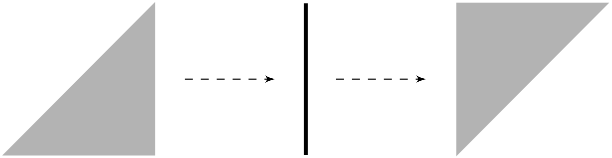
Technical tool: the corner-point abstraction



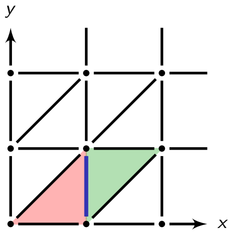
Abstract time successors:



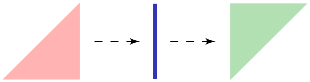
Concrete time successors:



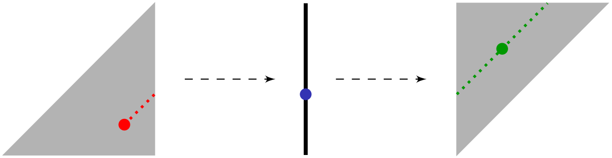
Technical tool: the corner-point abstraction



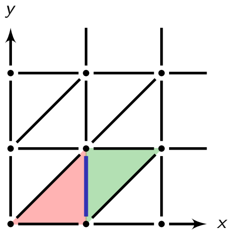
Abstract time successors:



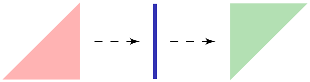
Concrete time successors:



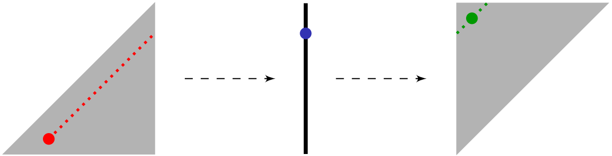
Technical tool: the corner-point abstraction



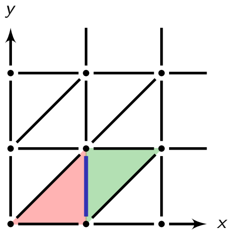
Abstract time successors:



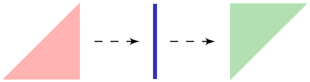
Concrete time successors:



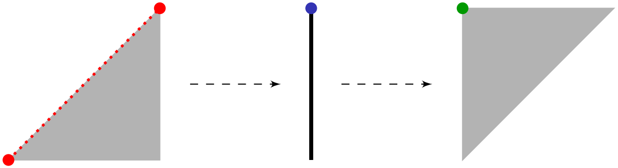
Technical tool: the corner-point abstraction



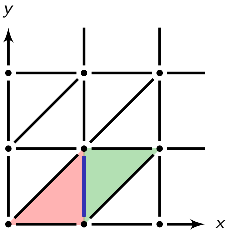
Abstract time successors:



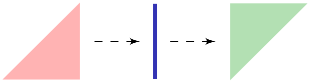
Concrete time successors:



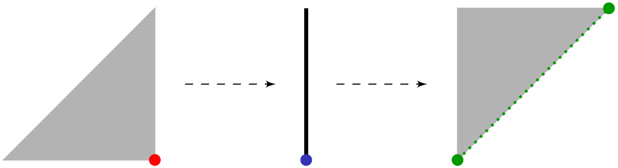
Technical tool: the corner-point abstraction



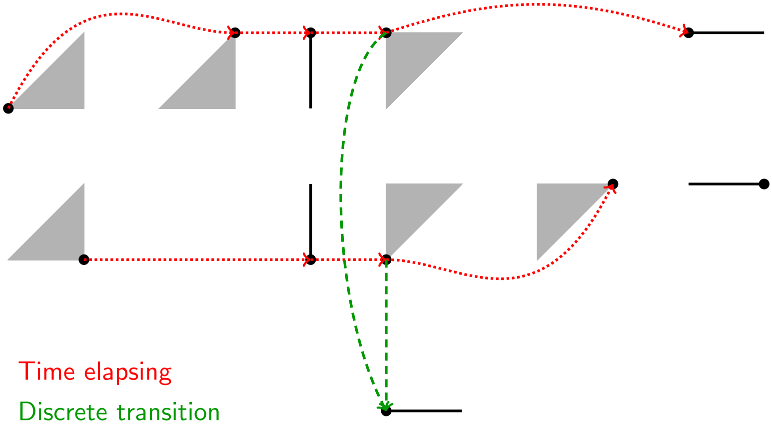
Abstract time successors:



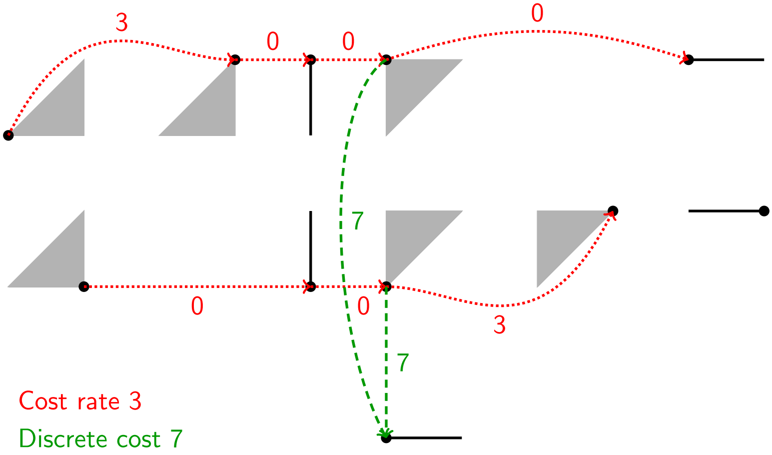
Concrete time successors:



Technical tool: the corner-point abstraction



Technical tool: the corner-point abstraction



From timed to discrete behaviours

Optimal reachability as a linear programming problem

From timed to discrete behaviours

Optimal reachability as a linear programming problem

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

From timed to discrete behaviours

Optimal reachability as a linear programming problem

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

From timed to discrete behaviours

Optimal reachability as a linear programming problem

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

From timed to discrete behaviours

Optimal reachability as a linear programming problem

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

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

From timed to discrete behaviours

Optimal reachability as a linear programming problem

$T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5$

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

Lemma

Let Z be a bounded zone and f be a function

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

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

From timed to discrete behaviours

Optimal reachability as a linear programming problem

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

$T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5$

Lemma

Let Z be a bounded zone and f be a function

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

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

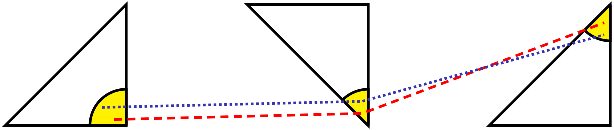
$\leadsto$ for every finite path π in $\mathcal{A}$, there exists a path Π in $\mathcal{A}_{\mathrm{cp}}$ such that

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

[Π is a “corner-point projection” of π]

From discrete to timed behaviours

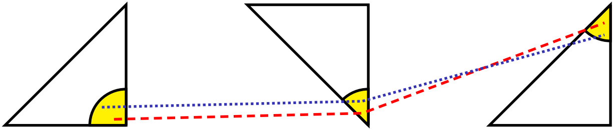
Approximation of abstract paths:



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

From discrete to timed behaviours

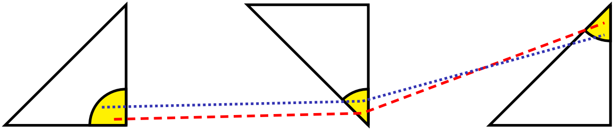
Approximation of abstract paths:



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

From discrete to timed behaviours

Approximation of abstract paths:

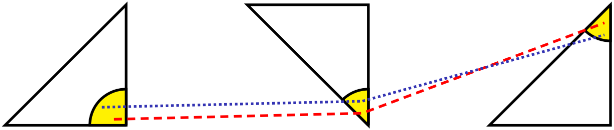


For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$, there exists a path π_ε of $\mathcal{A}$ s.t.

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

From discrete to timed behaviours

Approximation of abstract paths:



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

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

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

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

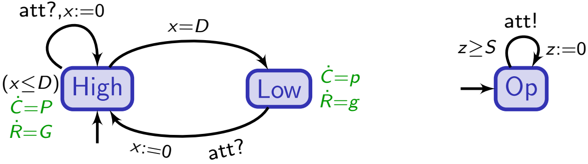
Use of the corner-point abstraction

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

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

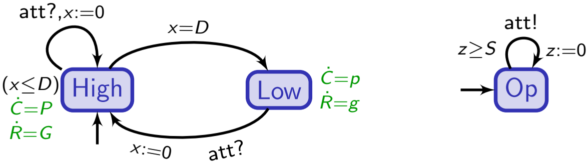
[BBL04] Bouyer, Brinksma, Larsen. Staying Alive As Cheaply As Possible *(HSCC'04)*.
[BBL08] Bouyer, Brinksma, Larsen. Optimal infinite scheduling for multi-priced timed automata *(Formal Methods in System Designs)*.
[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata *(INFINITY'08)*.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

Going further 1: mean-cost optimization



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

Going further 1: mean-cost optimization

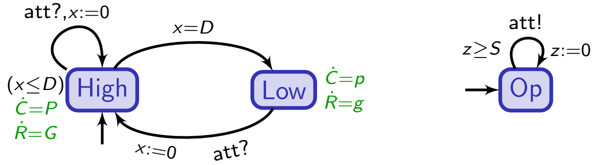


$\rightsquigarrow$ compute optimal infinite schedules that minimize

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

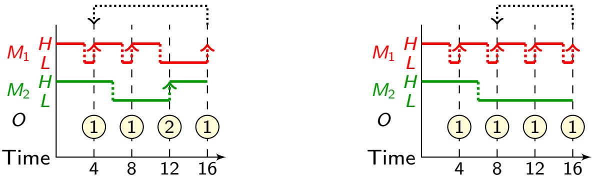
[BBL08] Bouyer, Brinksma, Larsen. Optimal infinite scheduling for multi-priced timed automata *(Formal Methods in System Designs)*.

Going further 1: mean-cost optimization



$\rightsquigarrow$ compute optimal infinite schedules that minimize

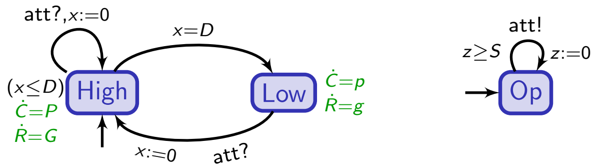
$$\text{mean-cost}(\pi) = \limsup_{n\to+\infty} \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)}$$



Schedule with ratio ≈ 1.455 Schedule with ratio ≈ 1.478

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

Going further 1: mean-cost optimization



$\rightsquigarrow$ compute optimal infinite schedules that minimize

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

Theorem [BBL08]

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

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

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

From timed to discrete behaviours

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

Lemma

Let Z be a bounded zone and f be a function

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

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

From timed to discrete behaviours

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

Lemma

Let Z be a bounded zone and f be a function

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

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

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

From timed to discrete behaviours

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

Lemma

Let Z be a bounded zone and f be a function

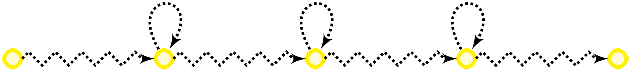
$$f : (t_1, \dots, t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

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

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

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

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



The (acyclic) linear part will be negligible!

From timed to discrete behaviours

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

Lemma

Let Z be a bounded zone and f be a function

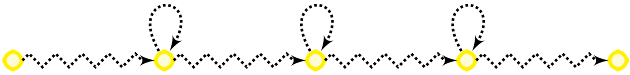
$$f : (t_1, \ldots, t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

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

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

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

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

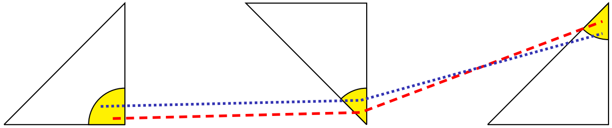


The (acyclic) linear part will be negligible!

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

From discrete to timed behaviours

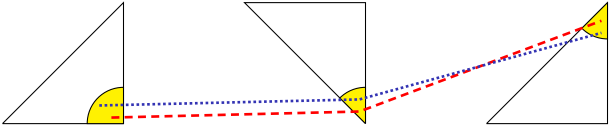
Approximation of abstract paths:



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

From discrete to timed behaviours

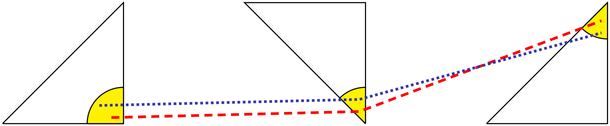
Approximation of abstract paths:



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

From discrete to timed behaviours

Approximation of abstract paths:

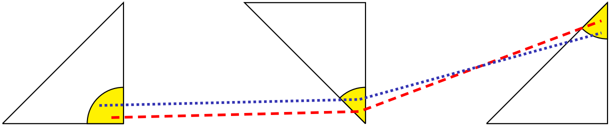


For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$, there exists a path π_ε of $\mathcal{A}$ s.t.

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

From discrete to timed behaviours

Approximation of abstract paths:



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

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

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

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

Going further 2: concavely-priced cost functions

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

Theorem [JT08]

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

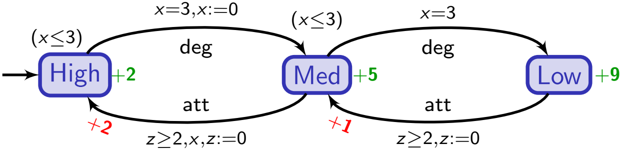
- optimal-time and optimal-cost reachability;
- optimal discrete discounted cost;
- optimal mean-cost.

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

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

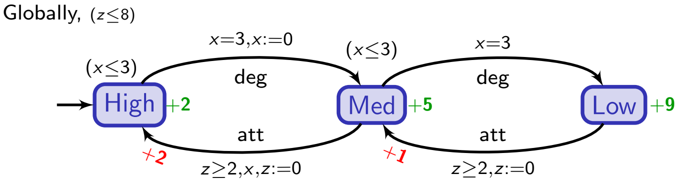
Going further 3: discounted-time cost optimization

Globally, $(z \le 8)$



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

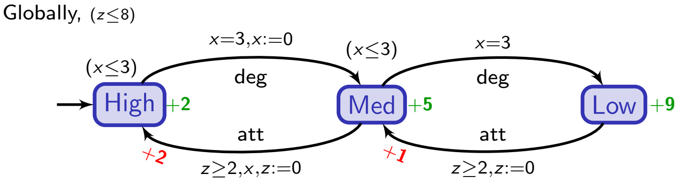
Going further 3: discounted-time cost optimization



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

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

Going further 3: discounted-time cost optimization



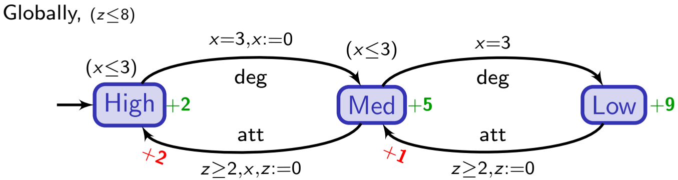
$\rightsquigarrow$ compute optimal infinite schedules that minimize

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

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

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

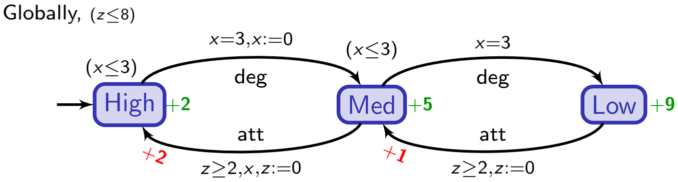
Going further 3: discounted-time cost optimization



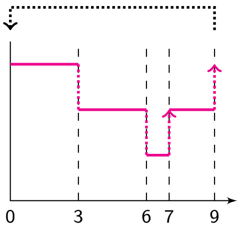
$\rightsquigarrow$ compute optimal infinite schedules that minimize
discounted cost over time

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

Going further 3: discounted-time cost optimization



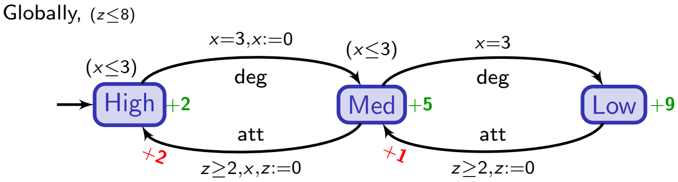
$\rightsquigarrow$ compute optimal infinite schedules that minimize
discounted cost over time



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

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

Going further 3: discounted-time cost optimization



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

Theorem [FL08]

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

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

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

And symbolically?

- Only for optimal reachability

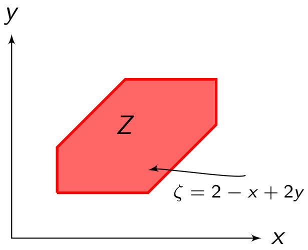
[LBB+01] Larsen, Behrmann, Brinksma, Fehnker, Hune, Pettersson, Romijn. As cheap as possible: Efficient cost- optimal reachability for priced timed automata *(CAV'01)*.

And symbolically?

- Only for optimal reachability

Priced zones

priced zone = zone + affine cost function



[LBB+01] Larsen, Behrmann, Brinksma, Fehnker, Hune, Pettersson, Romijn. As cheap as possible: Efficient cost- optimal reachability for priced timed automata *(CAV'01)*.

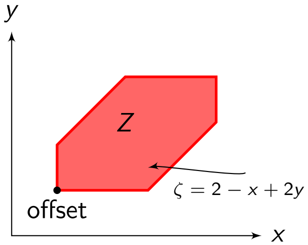
And symbolically?

- Only for optimal reachability

Priced zones

priced zone = zone + affine cost function

☺ efficient representation: DBM + offset cost + affine coefficient for each clock



Represented by: zone Z
offset cost: $+4$
rate for x: -1
rate for y: $+2$

[LBB+01] Larsen, Behrmann, Brinksma, Fehnker, Hune, Pettersson, Romijn. As cheap as possible: Efficient cost- optimal reachability for priced timed automata *(CAV'01)*.

And symbolically?

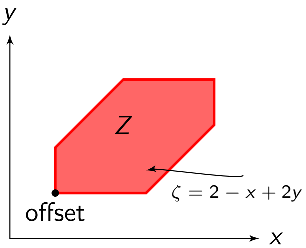
- Only for optimal reachability

Priced zones

priced zone = zone + affine cost function

☺ efficient representation: DBM + offset cost + affine coefficient for each clock

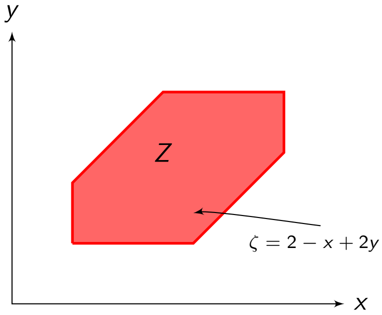
☹ the successor of a priced zone is a *union* of priced zones



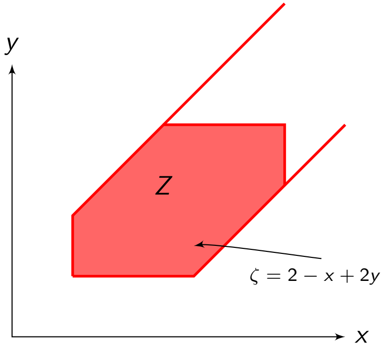
Represented by: zone Z
offset cost: $+4$
rate for x: -1
rate for y: $+2$

[LBB+01] Larsen, Behrmann, Brinksma, Fehnker, Hune, Pettersson, Romijn. As cheap as possible: Efficient cost- optimal reachability for priced timed automata *(CAV'01)*.

Computing the (time) successor of a priced zone



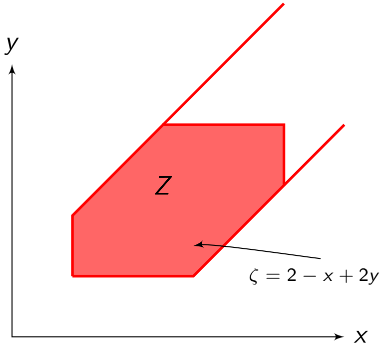
Computing the (time) successor of a priced zone



Cost rate in current location: $+3$

We want (Z', ζ') with
$\zeta'(v') = \min_{v'-\delta \in Z} \zeta(v' - \delta) + 3\delta$

Computing the (time) successor of a priced zone

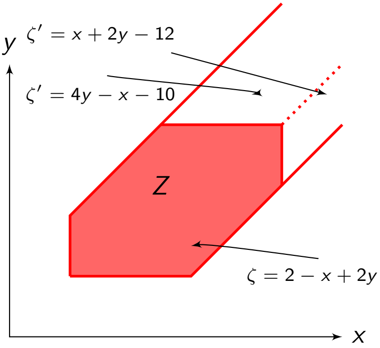


Cost rate in current location: $+3$

We want (Z', ζ') with
$\zeta'(v') = \min_{v'-\delta \in Z} \zeta(v' - \delta) + 3\delta$

- if $v' \in Z$, $\zeta'(v') = \zeta(v)$

Computing the (time) successor of a priced zone



Cost rate in current location: $+3$

We want (Z', ζ') with
$\zeta'(v') = \min_{v'-\delta \in Z} \zeta(v' - \delta) + 3\delta$

- if $v' \in Z$, $\zeta'(v') = \zeta(v)$
- otherwise, depends on the facet

Forward optimal reachability algorithm

Parameters: Abstraction abs and inclusion test $\preceq$

Forward optimal reachability algorithm

Parameters: Abstraction abs and inclusion test $\preceq$

- cost $\leftarrow +\infty$, $\texttt{Passed} \leftarrow \emptyset$ and $\texttt{Waiting} \leftarrow \{(\ell_0, \mathcal{Z}_0)\}$
- While $\texttt{Waiting} \neq \emptyset$
  - select $(\ell, \mathcal{Z})$ from Waiting
  - If ℓ is final and $\mathit{minCost}(\mathcal{Z}) <$ cost, then set $\mathit{minCost}(\mathcal{Z})$ to cost
  - If forall $(\ell, \mathcal{Z}') \in \texttt{Passed}$, $\mathcal{Z} \not\preceq \mathcal{Z}'$, then add abs$(\ell, \mathcal{Z})$ to Passed and add Post(abs$(\ell, \mathcal{Z})$) to Waiting
- Return cost

Results

Theorem [LBB+01,RLS06]

The forward algorithm with abs $=$ Id and $\preceq = \subseteq$ is correct and terminates for **bounded** timed automata with non-negative costs.

Termination: well-quasi-order on priced zones

[LBB+01] Larsen, Behrmann, Brinksma, Fehnker, Hune, Pettersson, Romijn. As cheap as possible: Efficient cost- optimal reachability for priced timed automata *(CAV'01)*.

[RLS06] Rasmussen, Larsen, Subramani. On using priced timed automata to achieve optimal scheduling *(Formal Methods in System Design)*.

Results

Theorem [LBB+01,RLS06]

The forward algorithm with $\mathsf{abs} = \mathsf{Id}$ and $\preceq\, = \,\subseteq$ is correct and terminates for **bounded** timed automata with non-negative costs.

Termination: well-quasi-order on priced zones

- Development of an (abstract) inclusion test $\sqsubseteq_M$ on priced zones

[LBB+01] Larsen, Behrmann, Brinksma, Fehnker, Hune, Pettersson, Romijn. As cheap as possible: Efficient cost- optimal reachability for priced timed automata *(CAV'01)*.
[RLS06] Rasmussen, Larsen, Subramani. On using priced timed automata to achieve optimal scheduling *(Formal Methods in System Design)*.
[BCM16] Bouyer, Colange, Markey. Symbolic Optimal Reachability in Weighted Timed Automata *(CAV'16)*.

Results

Theorem [LBB+01,RLS06]

The forward algorithm with abs = Id and $\preceq = \subseteq$ is correct and terminates for **bounded** timed automata with non-negative costs.

Termination: well-quasi-order on priced zones

- Development of an (abstract) inclusion test $\sqsubseteq_M$ on priced zones
- $\mathcal{Z} \sqsubseteq_M \mathcal{Z}'$ reduces to several bilevel linear optimization problems

[LBB+01] Larsen, Behrmann, Brinksma, Fehnker, Hune, Pettersson, Romijn. As cheap as possible: Efficient cost- optimal reachability for priced timed automata *(CAV'01)*.
[RLS06] Rasmussen, Larsen, Subramani. On using priced timed automata to achieve optimal scheduling *(Formal Methods in System Design)*.
[BCM16] Bouyer, Colange, Markey. Symbolic Optimal Reachability in Weighted Timed Automata *(CAV'16)*.

Results

Theorem [LBB+01,RLS06]

The forward algorithm with abs = Id and $\preceq = \subseteq$ is correct and terminates for **bounded** timed automata with non-negative costs.

Termination: well-quasi-order on priced zones

- Development of an (abstract) inclusion test $\sqsubseteq_M$ on priced zones
- $\mathcal{Z} \sqsubseteq_M \mathcal{Z}'$ reduces to several bilevel linear optimization problems

Theorem [BCM16]

The forward algorithm with abs = Id and $\preceq = \sqsubseteq_M$ is correct and terminates for timed automata with some conditions on the cost. It is always better than $\preceq = \subseteq$ for bounded timed automata.

[LBB+01] Larsen, Behrmann, Brinksma, Fehnker, Hune, Pettersson, Romijn. As cheap as possible: Efficient cost- optimal reachability for priced timed automata *(CAV'01)*.
[RLS06] Rasmussen, Larsen, Subramani. On using priced timed automata to achieve optimal scheduling *(Formal Methods in System Design)*.
[BCM16] Bouyer, Colange, Markey. Symbolic Optimal Reachability in Weighted Timed Automata *(CAV'16)*.

Tools

- Uppaal-Cora, developed in Aalborg (Denmark) between 2001-2005

  `http://people.cs.aau.dk/~adavid/cora/`

$\rightsquigarrow$ no more maintained

Tools

- Uppaal-Cora, developed in Aalborg (Denmark) between 2001-2005

  `http://people.cs.aau.dk/~adavid/cora/`

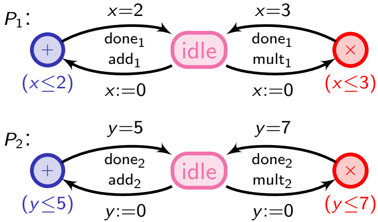
  $\leadsto$ no more maintained

- Our new tool TiAMo

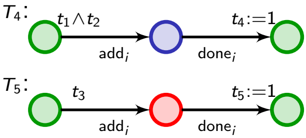
Outline

Modelling the task graph scheduling problem

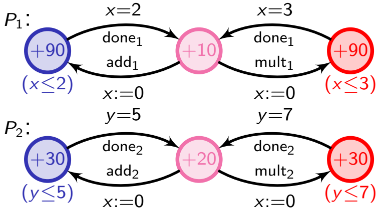
- Processors



- Tasks

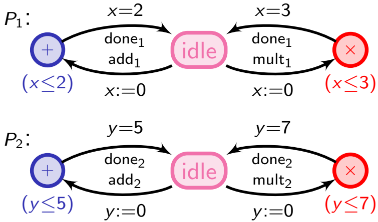


- Modelling energy

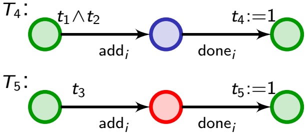


Modelling the task graph scheduling problem

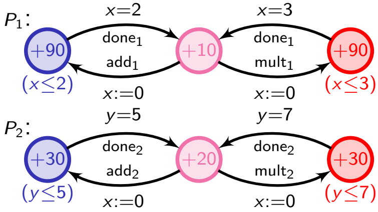
- Processors



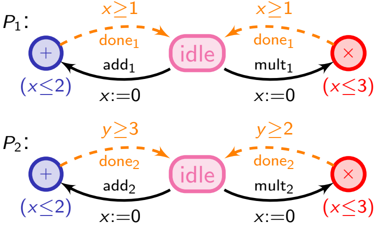
- Tasks



- Modelling energy



- Modelling uncertainty



Modelling the task graph scheduling problem

- Processors

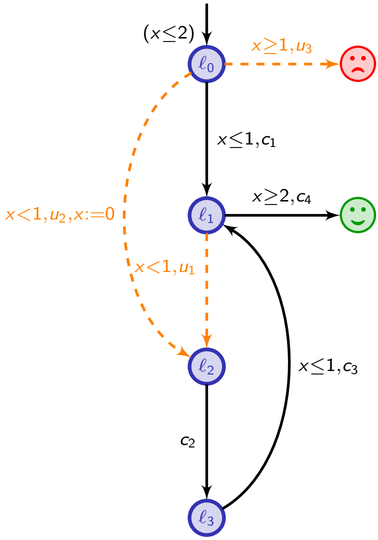
- Tasks

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

- Modelling energy

- Modelling uncertainty

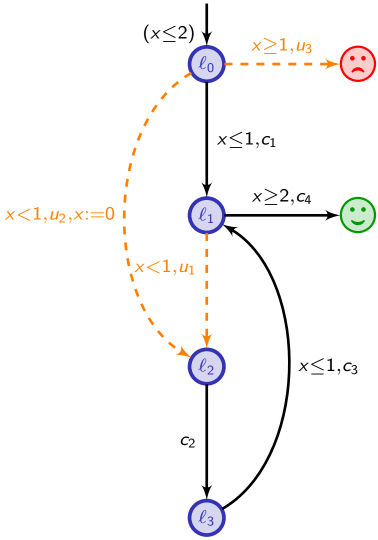
An example of a timed game



Rule of the game

- Aim: avoid ☹ and reach ☺

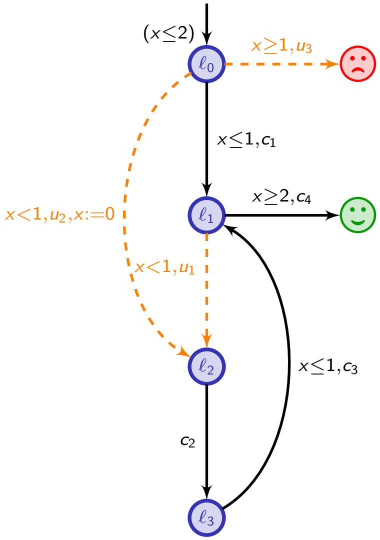
An example of a timed game



Rule of the game

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

An example of a timed game

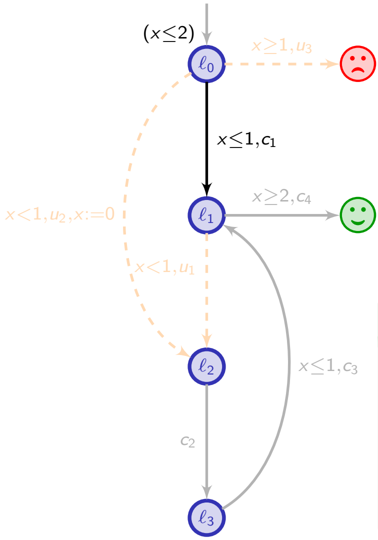


Rule of the game

- 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

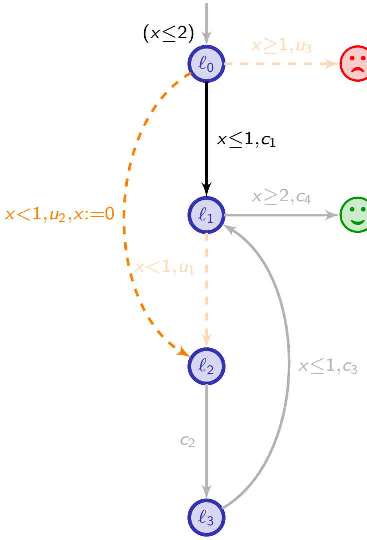
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

A (memoryless) winning strategy

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

An example of a timed game



Rule of the game

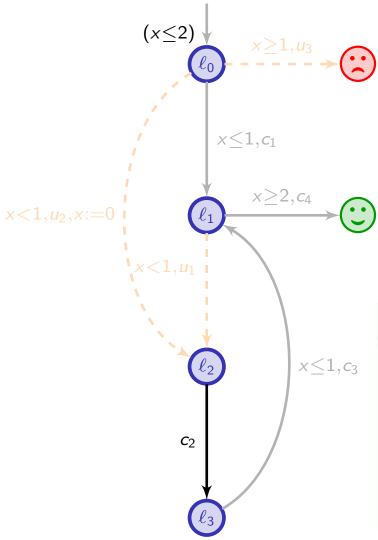
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

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

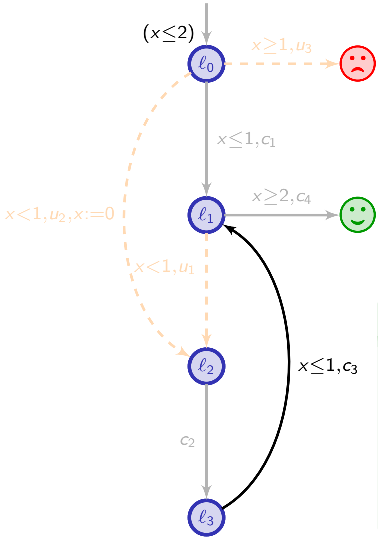
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

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

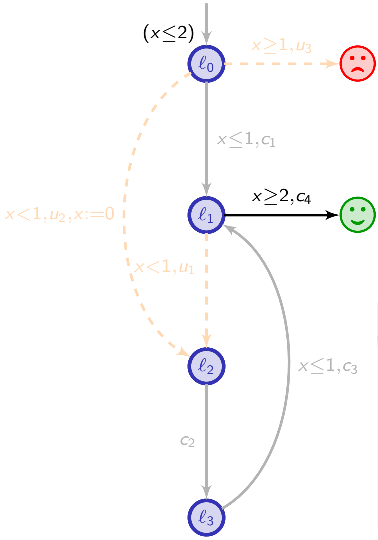
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

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

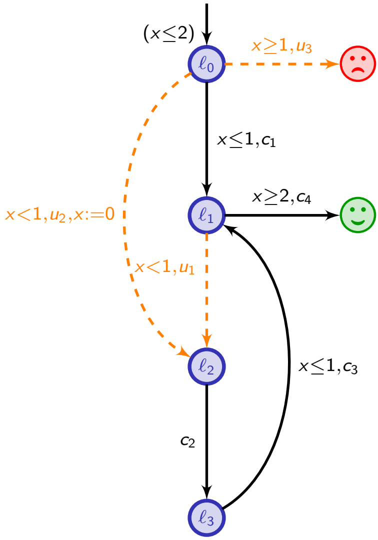
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

A (memoryless) winning strategy

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

An example of a timed game



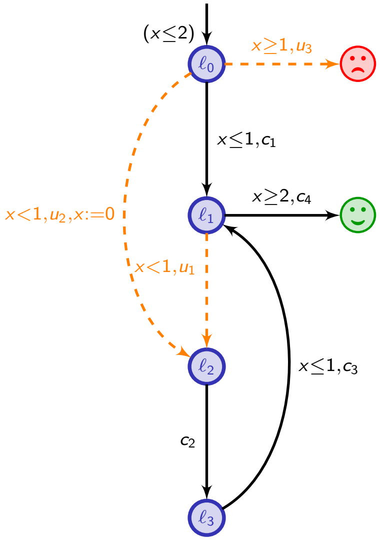
Rule of the game

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

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

Problems to be considered

An example of a timed game



Rule of the game

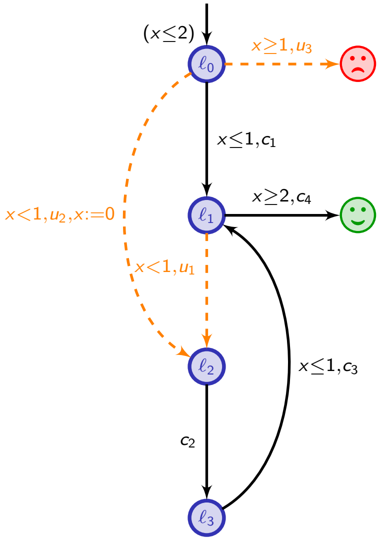
- Aim: avoid ☹ and reach ☺
- How do we play? According to a strategy:

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

Problems to be considered

- Does there exist a winning strategy?

An example of a timed game



Rule of the game

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

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

Problems to be considered

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

Decidability of timed games

Theorem [AMPS98,HK99]

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

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

Decidability of timed games

Theorem [AMPS98,HK99]

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

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

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

Decidability of timed games

Theorem [AMPS98,HK99]

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

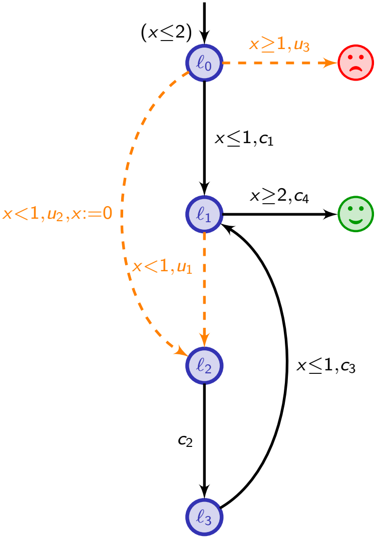
$\rightsquigarrow$ classical regions are sufficient for solving such problems

Theorem [AM99,BHPR07,JT07]

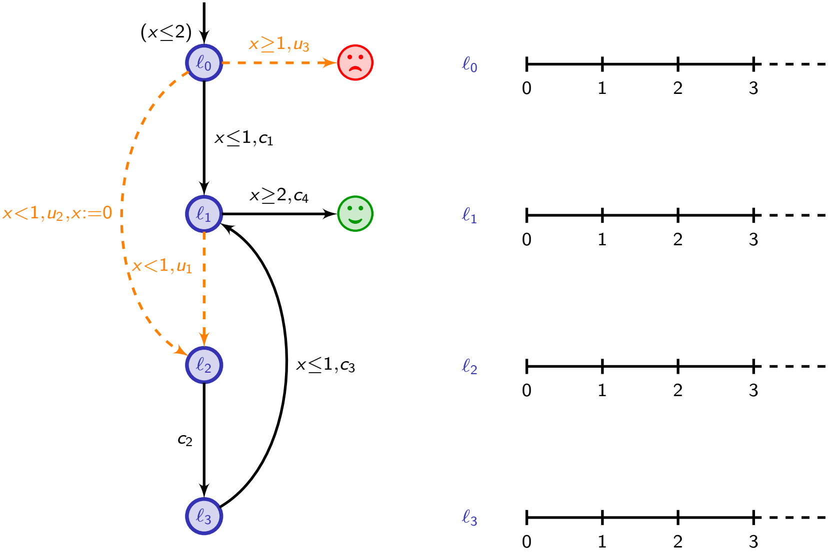
Optimal-time reachability timed games are decidable and EXPTIME-complete.

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

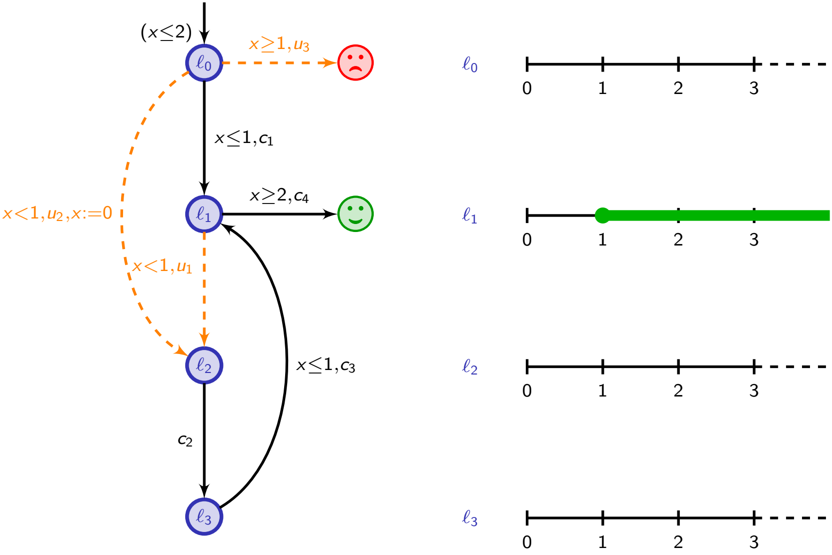
Back to the example: computing winning states



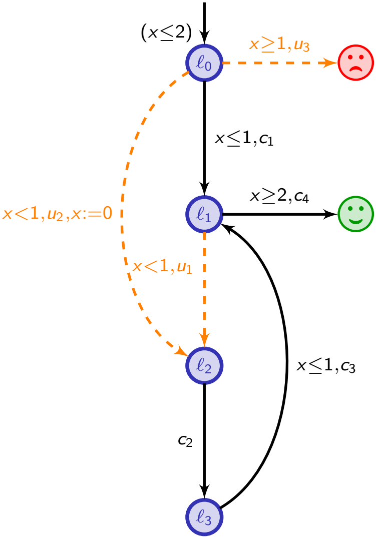
Back to the example: computing winning states

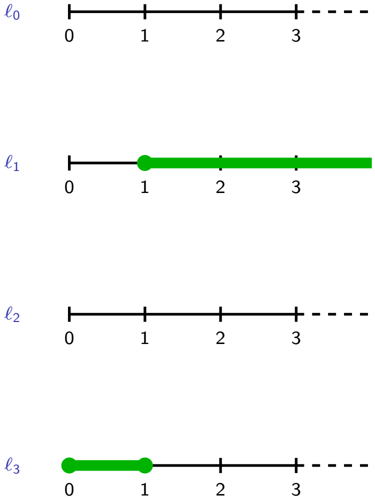


Back to the example: computing winning states

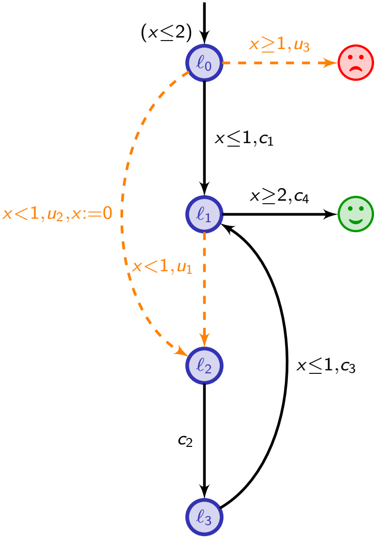


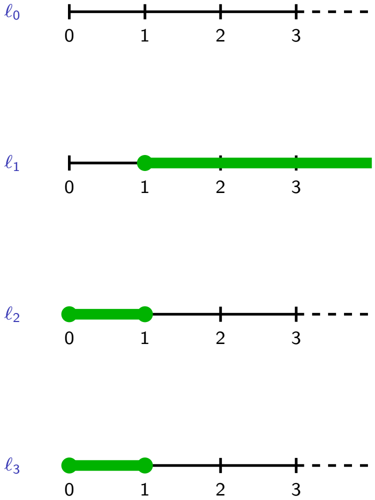
Back to the example: computing winning states



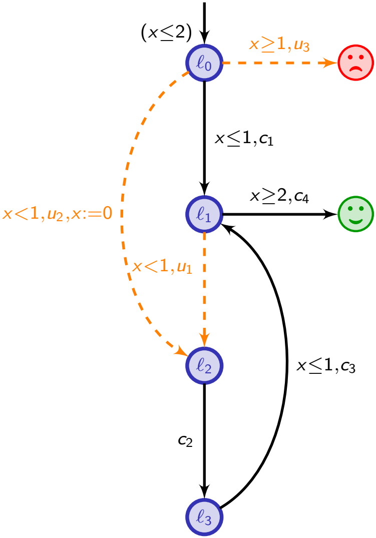


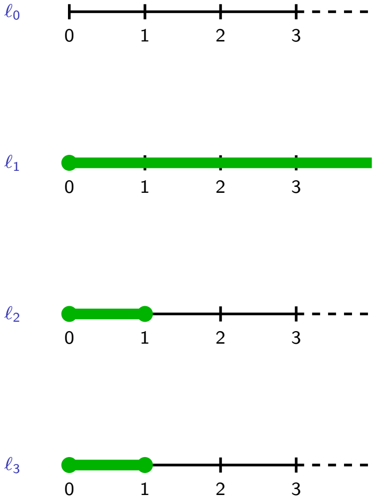
Back to the example: computing winning states



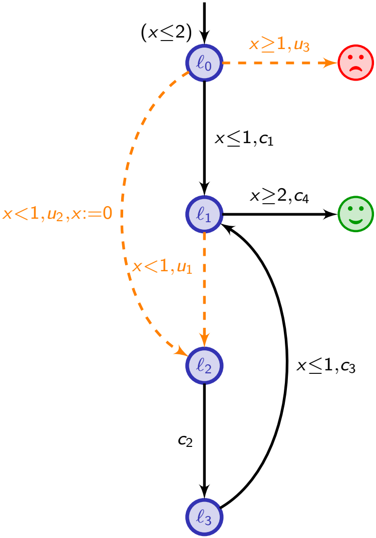


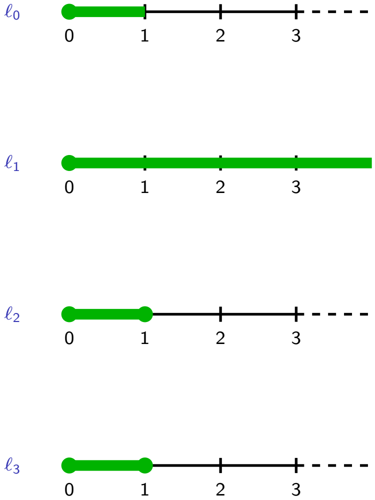
Back to the example: computing winning states



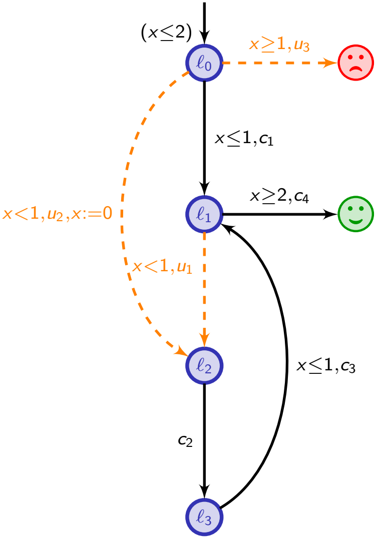


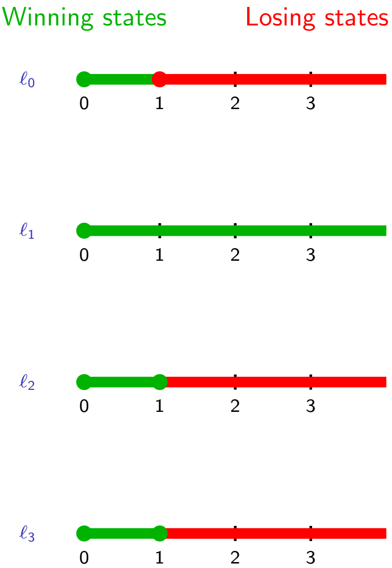
Back to the example: computing winning states





Back to the example: computing winning states





Decidability *via* attractors

Skip attractors

Decidability *via* attractors

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

Decidability *via* attractors

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

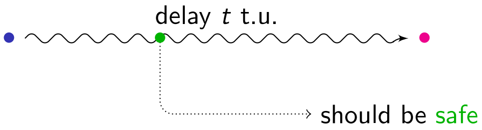
$$\mathrm{cPred}(X) = \bigcup_{a \text{ cont.}} \mathrm{Pred}^a(X) \qquad \mathrm{uPred}(X) = \bigcup_{a \text{ uncont.}} \mathrm{Pred}^a(X)$$

Decidability *via* attractors

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

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

- time controllable predecessors:

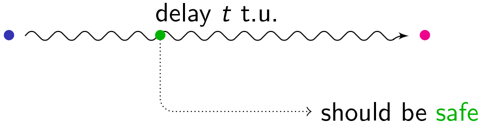


Decidability *via* attractors

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

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

- time controllable predecessors:



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

Timed games with a reachability objective

We write:

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

Timed games with a reachability objective

We write:

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

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

$$\mathsf{Attr}_1(☺) = \pi(☺)$$

Timed games with a reachability objective

We write:

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

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

$$\mathsf{Attr}_1(☺) = \pi(☺)$$

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

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

Timed games with a reachability objective

We write:

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

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

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

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

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

- ...

Timed games with a reachability objective

We write:

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

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

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

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

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

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

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

Timed games with a reachability objective

We write:

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

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

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

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

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

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

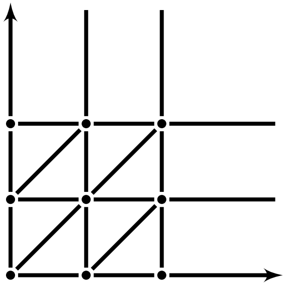
$$\begin{aligned}\mathsf{Attr}_n(\smiley) &= \pi(\mathsf{Attr}_{n-1}(\smiley)) \\ &= \pi^n(\smiley)\end{aligned}$$

Stability w.r.t. regions

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

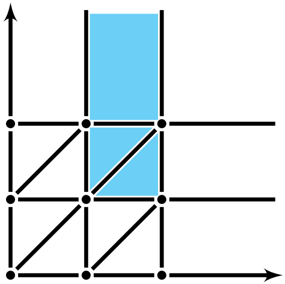
Stability w.r.t. regions

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



Stability w.r.t. regions

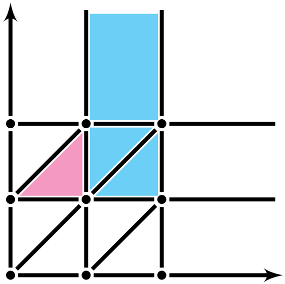
- if X is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does π also preserve unions of regions?



cPred(X)

Stability w.r.t. regions

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

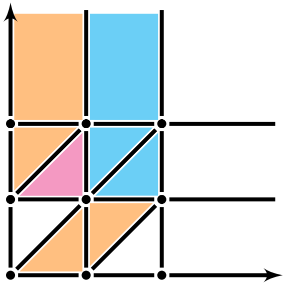


cPred(X)

uPred($\neg X$)

Stability w.r.t. regions

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



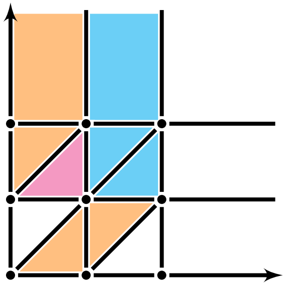
$\mathrm{cPred}(X)$

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

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

Stability w.r.t. regions

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



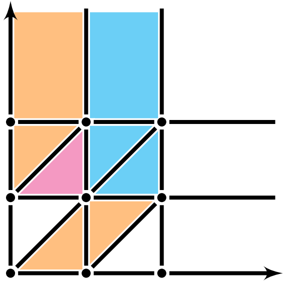
$\mathrm{cPred}(X)$

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

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

Stability w.r.t. regions

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



$\mathrm{cPred}(X)$

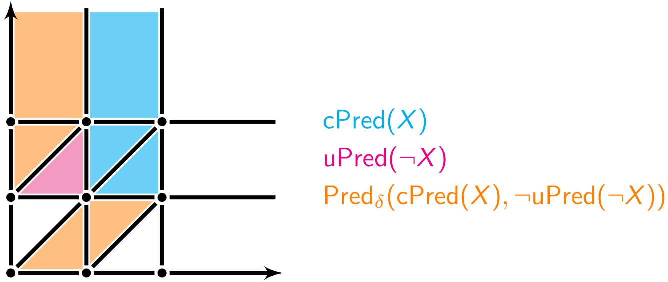
$\mathrm{uPred}(\neg X)$

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

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

Stability w.r.t. regions

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

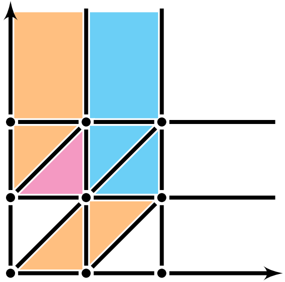


(but it generates non-convex unions of regions...)

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

Stability w.r.t. regions

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



$\text{cPred}(X)$

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

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

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

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

... and is correct

And in practice?

- A zone-based forward algorithm with backtracking [CDF+05,BCD+07]

[CDF+05] Cassez, David, Fleury, Larsen, Lime. Efficient On-the-Fly Algorithms for the Analysis of Timed Games *(CONCUR'05)*.
[BCD+07] Behrmann, Cougnard, David, Fleury, Larsen, Lime. UPPAAL-Tiga: Time for Playing Games! *(CAV'07)*.

And in practice?

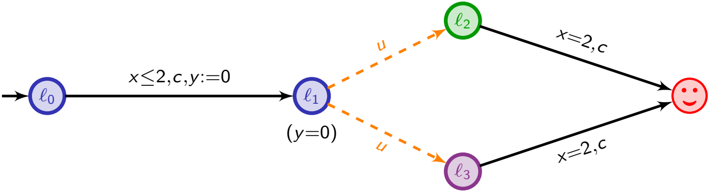
- A zone-based forward algorithm with backtracking [CDF+05,BCD+07]
- A tool: Uppaal-TiGa, developed in Aalborg (Denmark) since 2005

  `http://people.cs.aau.dk/~adavid/tiga/`

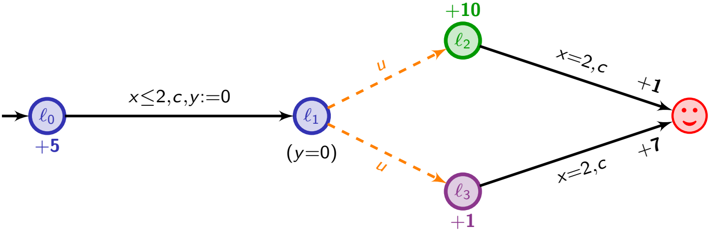
[CDF+05] Cassez, David, Fleury, Larsen, Lime. Efficient On-the-Fly Algorithms for the Analysis of Timed Games *(CONCUR'05)*.
[BCD+07] Behrmann, Cougnard, David, Fleury, Larsen, Lime. UPPAAL-Tiga: Time for Playing Games! *(CAV'07)*.

Outline

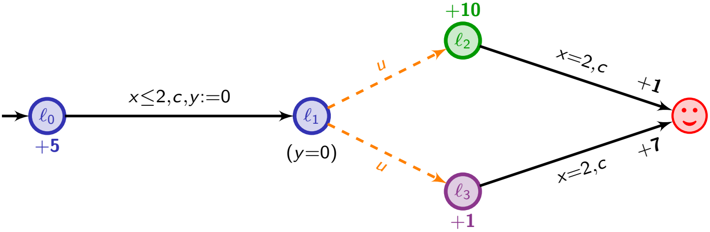
A simple timed game



A simple weighted timed game

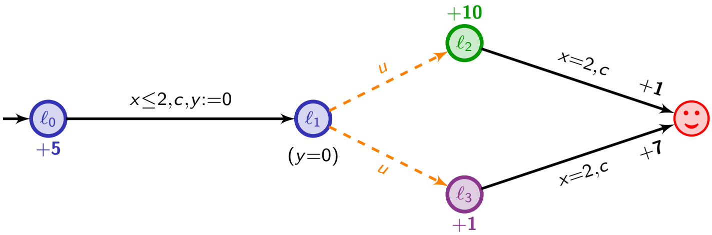


A simple weighted timed game



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

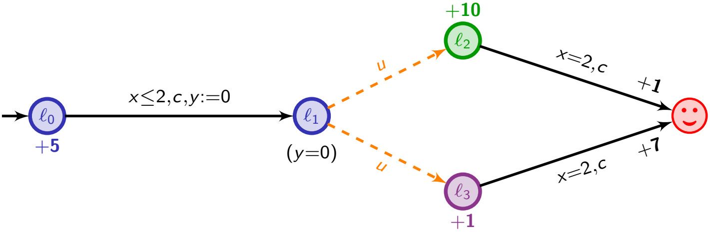
A simple weighted timed game



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

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

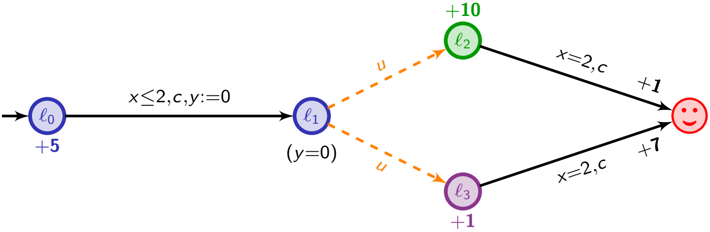
A simple weighted timed game



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

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

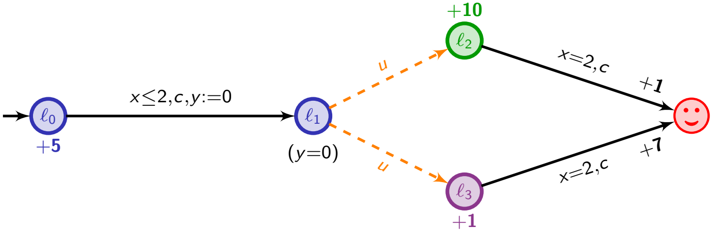
A simple weighted timed game



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

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

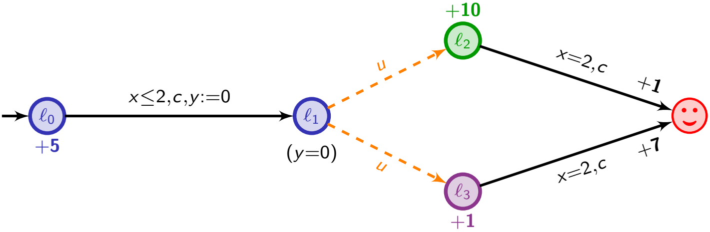
A simple weighted timed game



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

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

A simple weighted timed game



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

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

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

Optimal reachability in weighted timed games (1)

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

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

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

Optimal reachability in weighted timed games (1)

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

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

[LMM02]

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

Optimal reachability in weighted timed games (1)

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

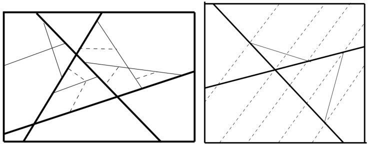
[LMM02,ABM04,BCFL04,BBR05,BBM06,BLMR06,Rut11,HIM13,BGK+14]

[LMM02]

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

[ABM04,BCFL04]

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



Optimal reachability in weighted timed games (2)

[BBR05,BBM06,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]

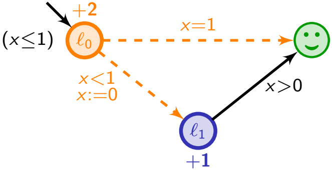
In weighted timed games, the optimal cost (and the value) cannot be computed, as soon as games have three clocks or more.

[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.

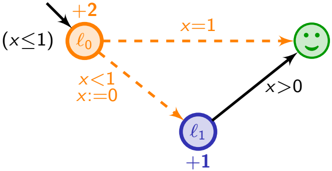
What is easier with a single clock?

- Memoryless strategies can be non-optimal...



What is easier with a single clock?

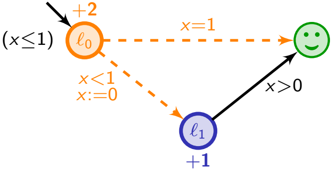
- Memoryless strategies can be non-optimal...



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

What is easier with a single clock?

- Memoryless strategies can be non-optimal...

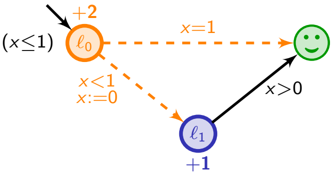


... but memoryless almost-optimal strategies will be sufficient.

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

What is easier with a single clock?

- Memoryless strategies can be non-optimal...

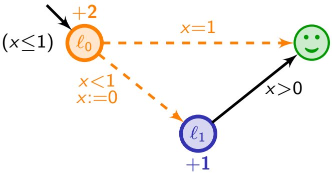


... but memoryless almost-optimal strategies will be sufficient.

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

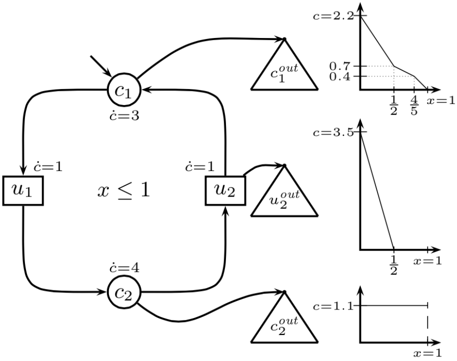
What is easier with a single clock?

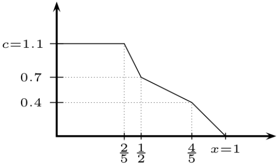
- Memoryless strategies can be non-optimal...



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

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





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

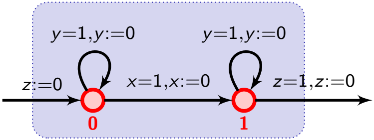
Computing the optimal cost: why is that hard?

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

Computing the optimal cost: why is that hard?

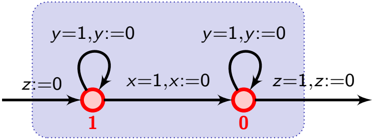
Given two clocks x and y, we can check whether $y = 2x$.

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



The cost is increased by x_0

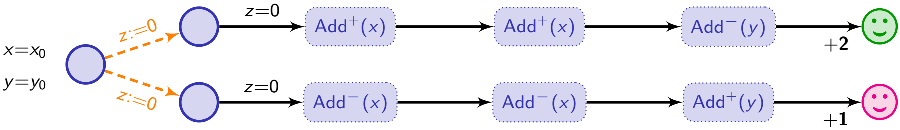
$\mathsf{Add}^-(x)$



The cost is increased by $1-x_0$

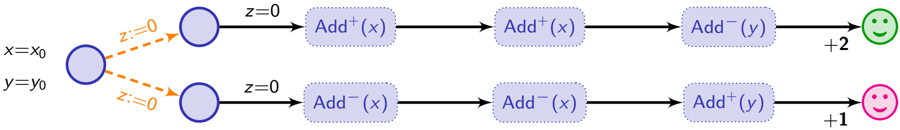
Computing the optimal cost: why is that hard?

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



Computing the optimal cost: why is that hard?

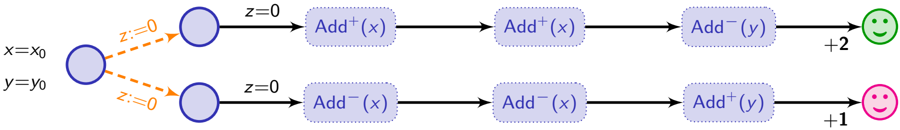
Given two clocks x and y, we can check whether $y = 2x$.



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

Computing the optimal cost: why is that hard?

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

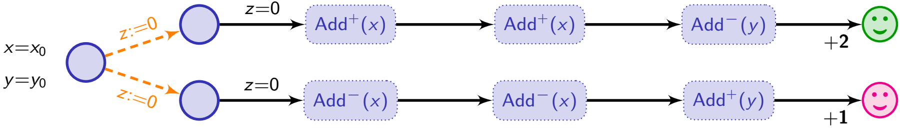


- In , cost $= 2x_0 + (1 - y_0) + 2$

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

Computing the optimal cost: why is that hard?

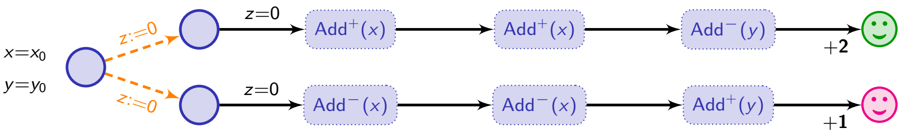
Given two clocks x and y, we can check whether $y = 2x$.



- 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?

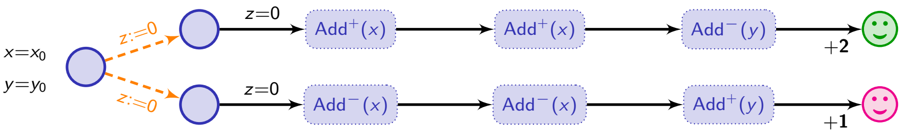
Given two clocks x and y, we can check whether $y = 2x$.



- In 🙂 (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

Computing the optimal cost: why is that hard?

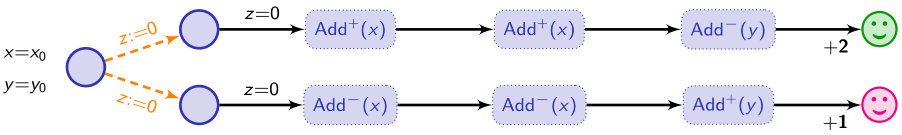
Given two clocks x and y, we can check whether $y = 2x$.



- 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 $y = 2x$.

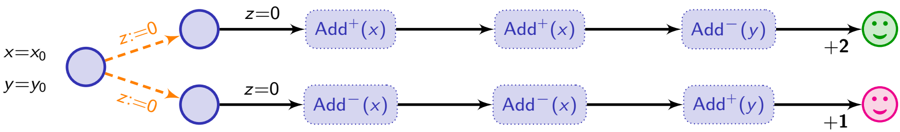


- In , cost $= 2x_0 + (1 - y_0) + 2$
  In , cost $= 2(1 - x_0) + y_0 + 1$
- if $y_0 < 2x_0$, player 2 chooses the first branch: cost > 3
  if $y_0 > 2x_0$, player 2 chooses the second branch: cost > 3
  if $y_0 = 2x_0$, in both branches, cost $= 3$

  $\rightsquigarrow$ 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:

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

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:

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

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?

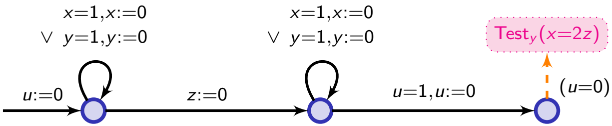
Player 1 will simulate a two-counter machine:

- 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.

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



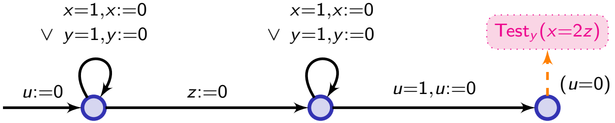
Computing the optimal cost: why is that hard?

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.



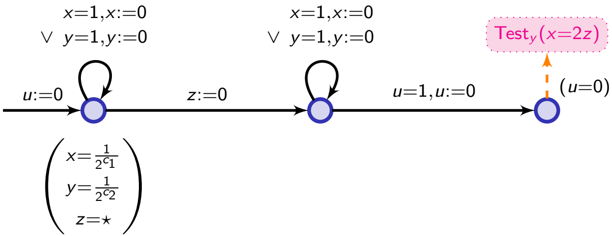
Computing the optimal cost: why is that hard?

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:

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

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



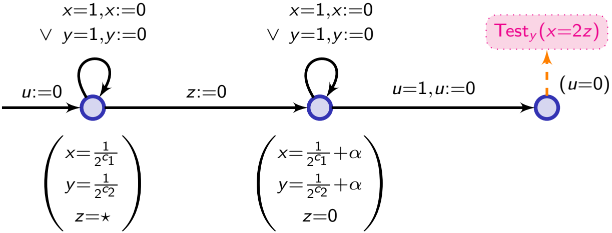
Computing the optimal cost: why is that hard?

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:

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

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



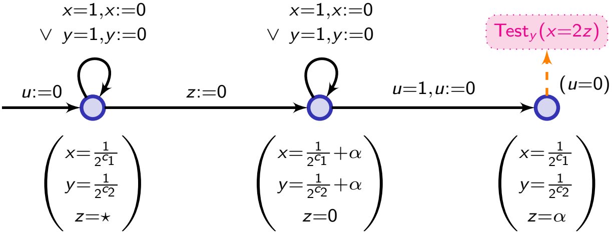
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

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

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

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



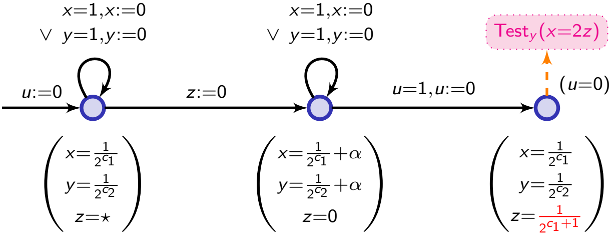
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

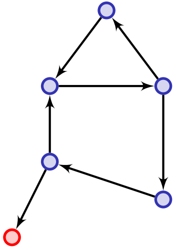
- each instruction is encoded as a module;
- the counter values c_1 and c_2 are encoded by two clocks:

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

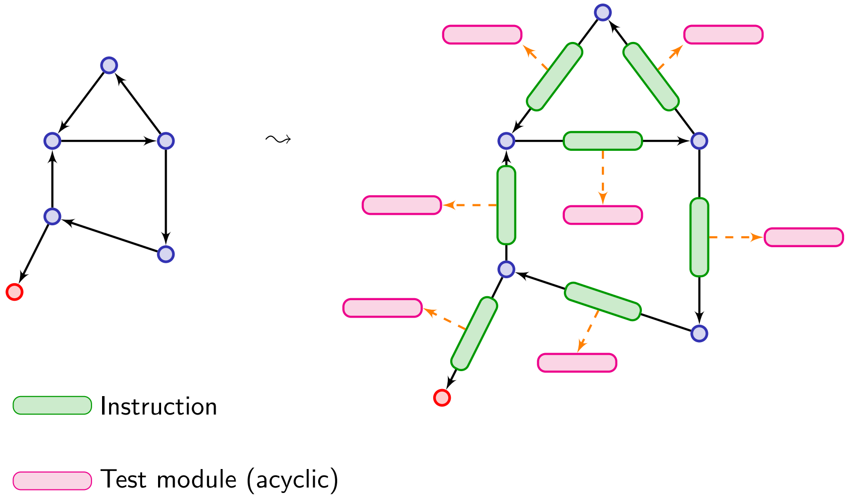
The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.



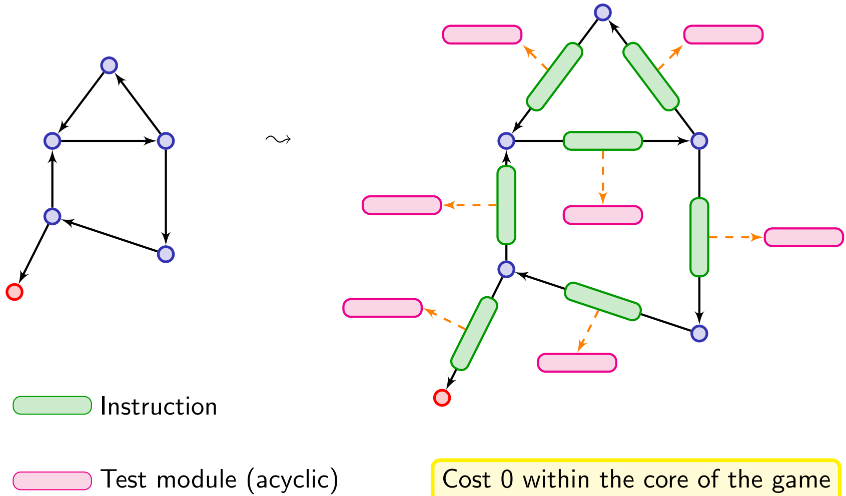
Shape of the reduction



Shape of the reduction



Shape of the reduction



Cost 0 within the core of the game

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,

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

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,

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

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,

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

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,

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

[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,

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

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,

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

- 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)*.

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,

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

- 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...

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

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

Approximation of the optimal cost

Theorem

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

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

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

Approximation of the optimal cost

Theorem

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

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

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

- one strategy σ_ϵ such that

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

[it is an ϵ-optimal winning strategy]

Approximation of the optimal cost

Theorem

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

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

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

- one strategy σ_ϵ such that

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

[it is an ϵ-optimal winning strategy]

Skip approximation scheme

Approximation of the optimal cost

Theorem

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

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

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

- one strategy σ_ϵ such that

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

[it is an ϵ-optimal winning strategy]

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

Approximation of the optimal cost

Theorem

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

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

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

- one strategy σ_ϵ such that

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

[it is an ϵ-optimal winning strategy]

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

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

Idea for approximation

Idea

Only partially unfold the game:

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

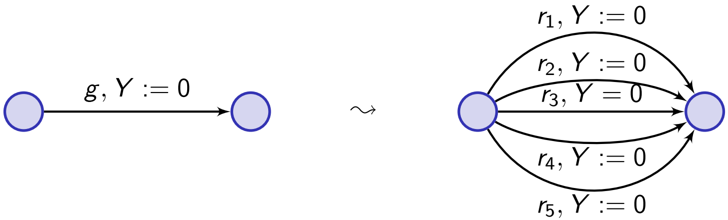
Idea for approximation

Idea

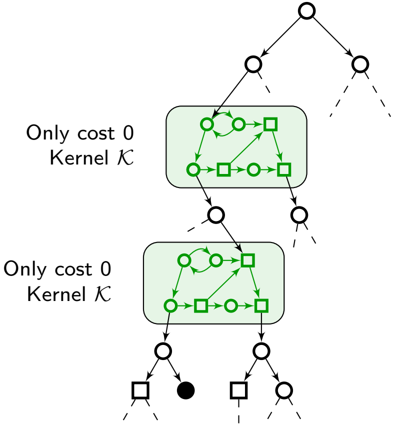
Only partially unfold the game:

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

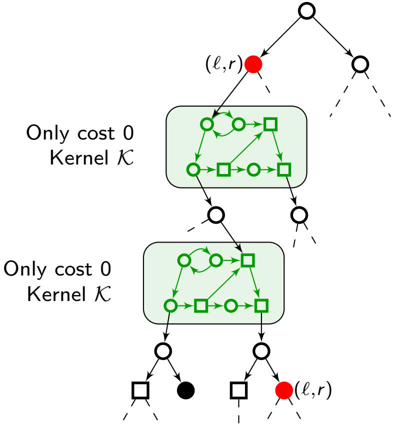
First: split the game along regions!



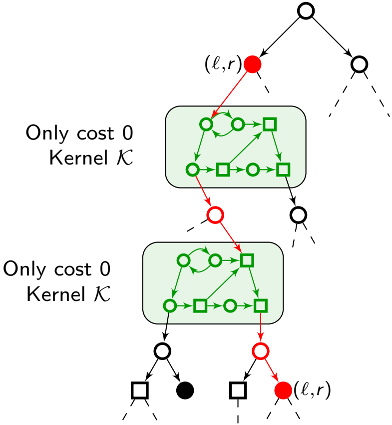
Semi-unfolding



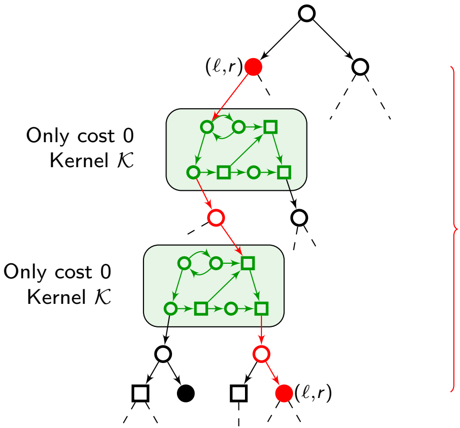
Semi-unfolding



Semi-unfolding

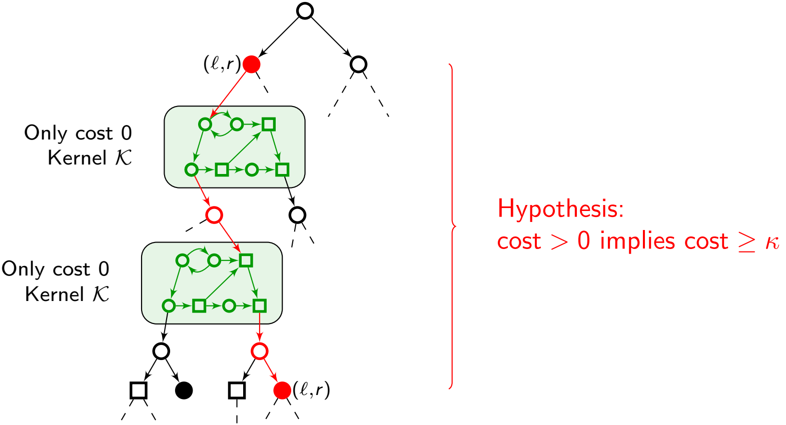


Semi-unfolding



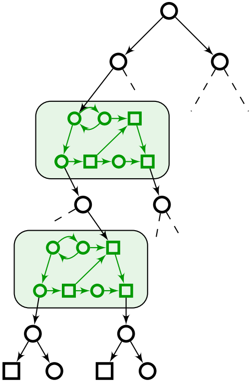
Hypothesis:
cost > 0 implies cost $\geq \kappa$

Semi-unfolding

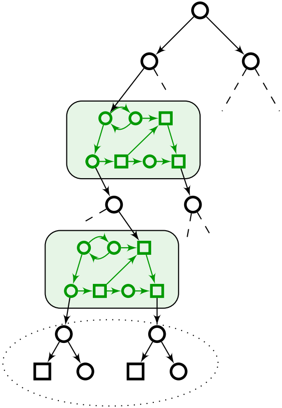


Conclusion: we can stop unfolding the game after N steps
(e.g. $N = (M+2) \cdot |\mathcal{R}(\mathcal{A})|$, where M is a pre-computed bound on $\mathrm{optcost}_{\mathcal{G}}$)

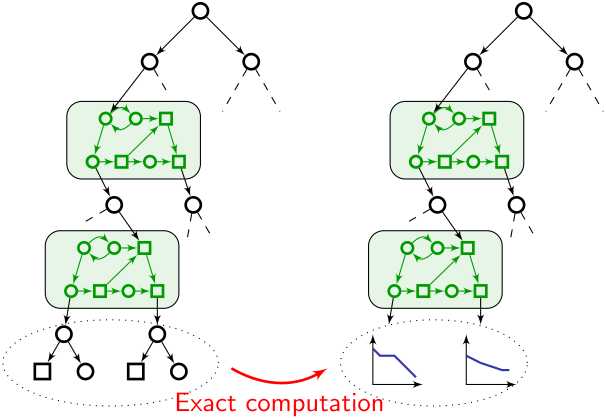
Approximation scheme



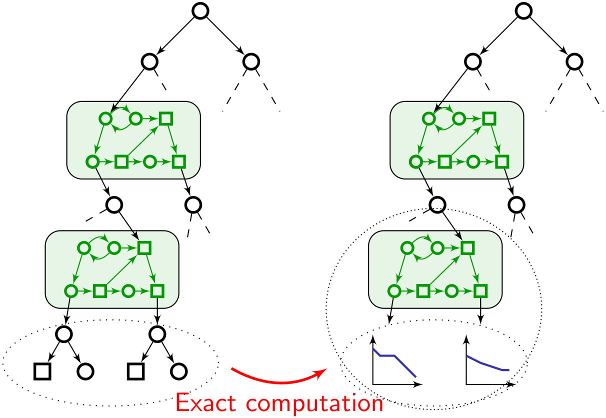
Approximation scheme



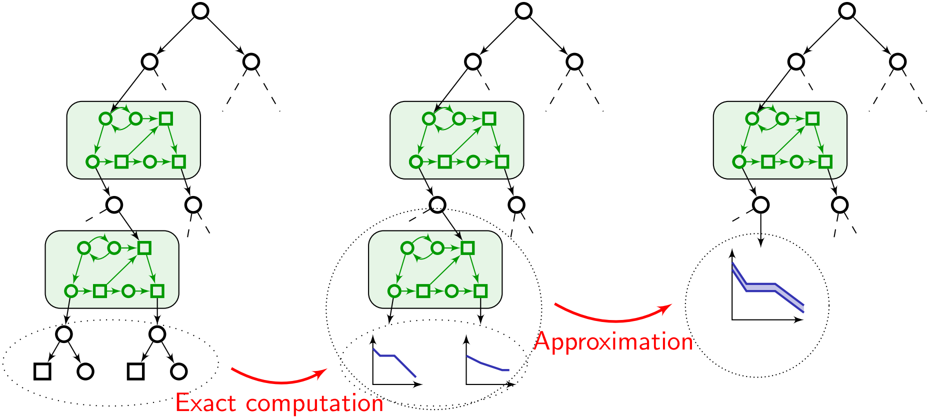
Approximation scheme



Approximation scheme



Approximation scheme



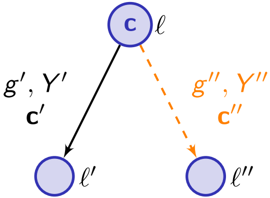
First step: Tree-like parts

$\rightsquigarrow$ 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

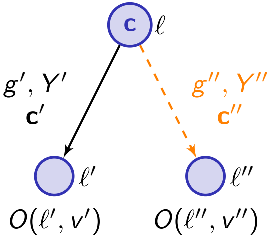
$\rightsquigarrow$ Goes back to [LMM02]



[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]

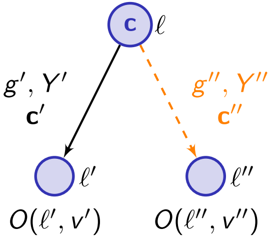


$O(\ell, v) =$

[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]

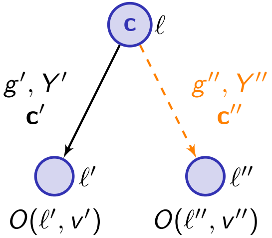


$$O(\ell, v) = \inf_{t' \mid v+t' \models g'}$$

[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]

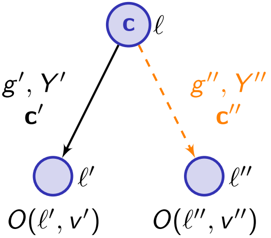


$$O(\ell, v) = \inf_{t' \mid v+t' \models g'} \max(\quad , \quad)$$

[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]



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

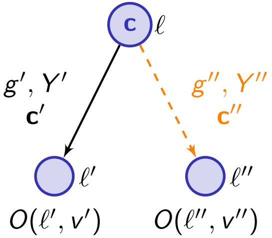
$$(\alpha) = t'\mathbf{c} + \mathbf{c}' + O(\ell', v')$$

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

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

First step: Tree-like parts

$\rightsquigarrow$ Goes back to [LMM02]



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

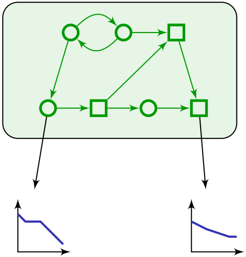
$$(\alpha) = t'\mathbf{c} + \mathbf{c}' + O(\ell', v')$$

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

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

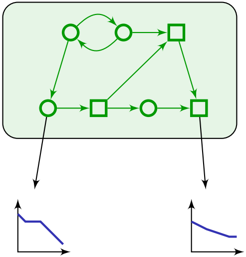
[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata *(TCS@02)*.

Second step: Kernels



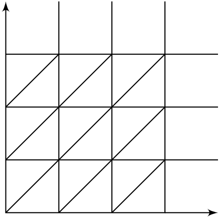
Output cost functions f

Second step: Kernels

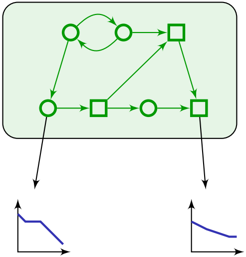


Output cost functions f

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

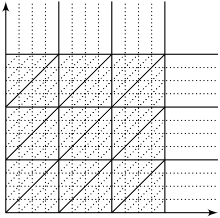


Second step: Kernels

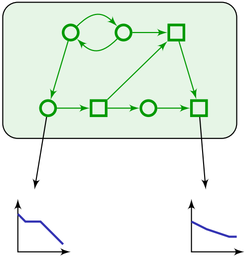


Output cost functions f

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

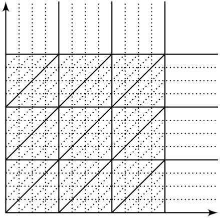


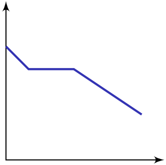
Second step: Kernels



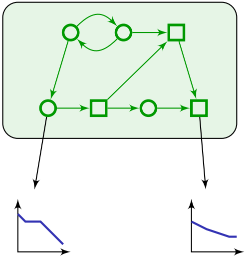
Output cost functions f

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



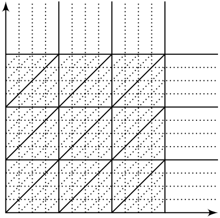


Second step: Kernels

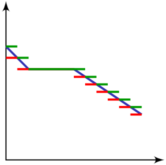


Output cost functions f

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

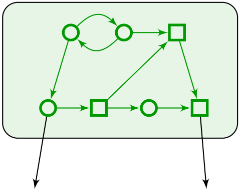


2. Under- and over-approximate by piecewise constant functions f_ϵ^- and f_ϵ^+



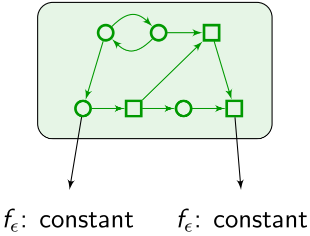
Second step: Kernels

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



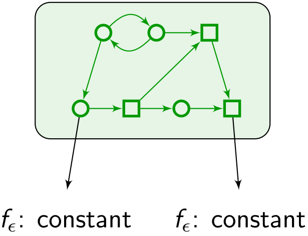
f_ϵ: constant f_ϵ: constant

Second step: Kernels



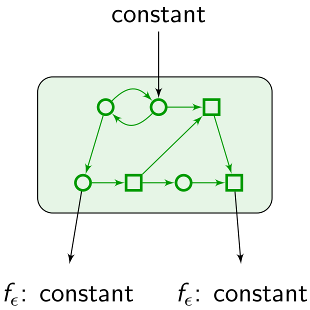
3. Refine/split the kernel along the new small regions and fix f_ϵ^- or f_ϵ^+, write f_ϵ
4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by f_ϵ)

Second step: Kernels



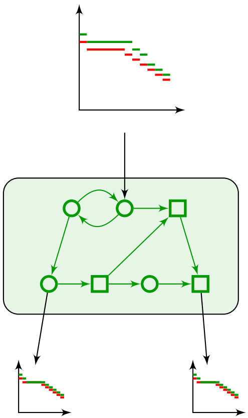
3. Refine/split the kernel along the new small regions and fix f_ϵ^- or f_ϵ^+, write f_ϵ
4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by f_ϵ)
5. Those can be solved using standard technics based on attractors: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

Second step: Kernels



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

Second step: Kernels



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

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

Outline

TiAMo = Timed Automata Model-checker

- Development started in September 2015
- Main developer: Maximilien Colange (LSV)
- Uses some previous code by Ocan Sankur (IRISA)

TiAMo = Timed Automata Model-checker

- Development started in September 2015
- Main developer: Maximilien Colange (LSV)
- Uses some previous code by Ocan Sankur (IRISA)

Why?

- Main tool for timed systems: Uppaal, developed since 1995

$\rightsquigarrow$ Unfortunately, not open source

$\rightsquigarrow$ Often hard to know what is exactly implemented

TiAMo = Timed Automata Model-checker

- Development started in September 2015
- Main developer: Maximilien Colange (LSV)
- Uses some previous code by Ocan Sankur (IRISA)

Why?

- Main tool for timed systems: Uppaal, developed since 1995

$\rightsquigarrow$ Unfortunately, not open source

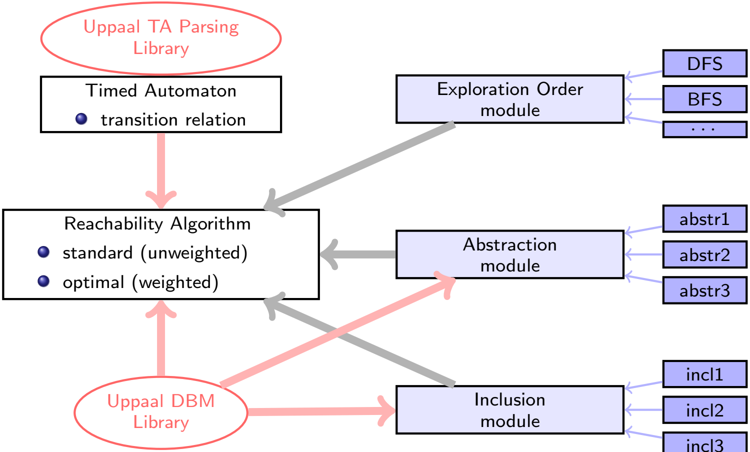
$\rightsquigarrow$ Often hard to know what is exactly implemented

What TiAMo targets

- Be a platform for experiments (open source!)
- Assert and compare algorithms

`https://git.lsv.fr/colange/tiamo`

TiAMo architecture



What is implemented

Exploration strategies

- BFS, DFS
- best cost first (for weighted models)
- preference-based (use a special "preference" variable in the model)
- "smart" BFS: inspired by [HT15]

[HT15] Herbreteau, Tran. Improving Search Order for Reachability Testing in Timed Automata *(FORMATS'15)*.

What is implemented

Exploration strategies

- BFS, DFS
- best cost first (for weighted models)
- preference-based (use a special "preference" variable in the model)
- "smart" BFS: inspired by [HT15]

Abstractions

- identity (i.e. no abstraction), to be used with abstract inclusion tests
- LU-abstraction [BBLP06]

[HT15] Herbreteau, Tran. Improving Search Order for Reachability Testing in Timed Automata *(FORMATS'15)*.

What is implemented

Exploration strategies

- BFS, DFS
- best cost first (for weighted models)
- preference-based (use a special "preference" variable in the model)
- "smart" BFS: inspired by [HT15]

Abstractions

- identity (i.e. no abstraction), to be used with abstract inclusion tests
- LU-abstraction [BBLP06]

Inclusions

- set inclusion
- weighted set inclusion [RLS06]
- abstract inclusion [HSW12]
- abstract weighted inclusion [BCM16]

[HT15] Herbreteau, Tran. Improving Search Order for Reachability Testing in Timed Automata *(FORMATS'15)*.

Experiments

- Various metrics
  - wall-clock time
  - number of symbolic states explored
  - number of comparisons done
- In various contexts
  - with different exploration strategies

Experiments

- Various metrics
  - wall-clock time
  - number of symbolic states explored
  - number of comparisons done
- In various contexts
  - with different exploration strategies

Examples taken from Uppaal-Cora

- The Aircraft Landing System (ALS)
- The Energy-optimal Task-graph Scheduling (ETS)
- The Vehicle Routing Problem with Time Windows (VRPTW)
- An unbounded-clock (ad hoc) model

Experiments

- Various metrics
  - wall-clock time
  - number of symbolic states explored
  - number of comparisons done
- In various contexts
  - with different exploration strategies

Examples taken from Uppaal-Cora

- The Aircraft Landing System (ALS)
- The Energy-optimal Task-graph Scheduling (ETS)
- The Vehicle Routing Problem with Time Windows (VRPTW)
- An unbounded-clock (ad hoc) model

Experimental results

- for mentioned weighted models
- -P (no pruning) / +P (pruning)
- $\subseteq$ (standard inclusion – no guarantee of term.) / $\sqsubseteq$ (abstract inclusion – guarantee of term.)

			# Waiting	# Passed	# stored	# tests	# succ. tests	time (s.)
ASL	+P	$\sqsubseteq$	11,820	4,785	9,324	3.7×10^{05}	13,676	0.3
		$\subseteq$	32,322	13,036	26,555	2.9×10^{06}	32,263	0.7
	-P	$\sqsubseteq$	1.7×10^{06}	1.5×10^{06}	6.9×10^{05}	8.1×10^{08}	1.2×10^{07}	312.7
		$\subseteq$	TO	TO	TO	TO	TO	TO
ETS	+P	$\sqsubseteq$	107	84	83	174	66	0.0
		$\subseteq$	664	606	590	17,684	455	0.0
VRPTW	+P	$\sqsubseteq$	6.0×10^{05}	4.8×10^{05}	5.6×10^{05}	6.2×10^{06}	1.7×10^{05}	11.3
		$\subseteq$	1.5×10^{06}	1.3×10^{06}	1.4×10^{06}	9.1×10^{07}	7.0×10^{05}	27.5
	-P	$\sqsubseteq$	1.3×10^{06}	1.3×10^{06}	1.3×10^{06}	2.5×10^{07}	7.0×10^{05}	23.9
		$\subseteq$	5.8×10^{06}	5.8×10^{06}	5.4×10^{06}	1.1×10^{09}	1.9×10^{06}	111.2
unbound.	+P	$\sqsubseteq$	14	13	14	135	3	0.0
		$\subseteq$	TO	TO	TO	TO	TO	TO
	-P	$\sqsubseteq$	14	14	14	135	3	0.0
		$\subseteq$	TO	TO	TO	TO	TO	TO

Outline

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- Various (un)decidability + symbolic technics
- Our new tool TiAMo

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- Various (un)decidability + symbolic technics
- Our new tool TiAMo

Future work

- Various theoretical issues
  - Apply further the idea of approximation
  - Stochastic uncertainty

Conclusion

Summary of the talk

- Overview of results concerning the optimal reachability problem in weighted timed automata and games
- Various (un)decidability + symbolic technics
- Our new tool TiAMo

Future work

- Various theoretical issues
  - Apply further the idea of approximation
  - Stochastic uncertainty
- Continue working on TiAMo
  - Implementation of (weighted) timed games
  - More applications (e.g. motion planning problems using the funnel automata approach [BMPS15])

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