Weighted Timed Automata: Model-Checking and Games

Patricia Bouyer

LSV – CNRS & ENS de Cachan – France

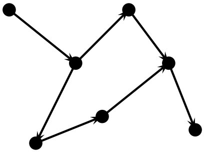
Based on joint works with Thomas Brihaye, Ed Brinksma, Véronique Bruyère, Franck Cassez, Emmanuel Fleury, François Laroussinie, Kim G. Larsen, Nicolas Markey, Jean-François Raskin, and Jacob Illum Rasmussen

Outline

Model-checking

Does the system satisfy the property?

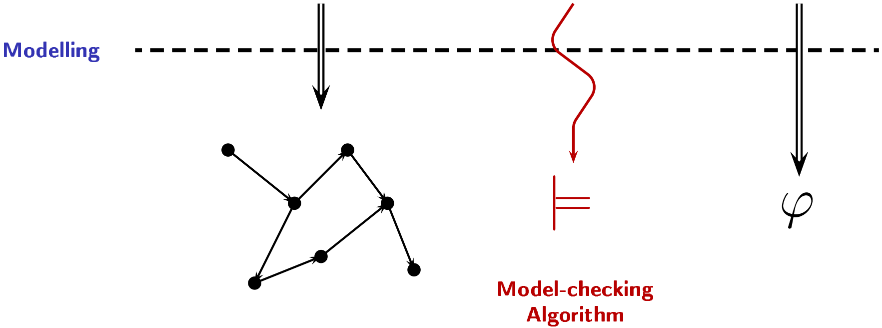
Modelling



φ

Model-checking

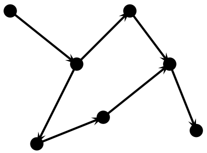
Does the system satisfy the property?



Controller synthesis

Can we guide the system so that it satisfies the property?

Modelling

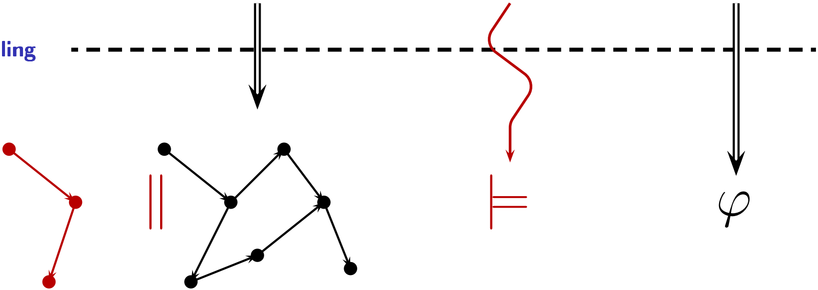


φ

Controller synthesis

Can we guide the system so that it satisfies the property?

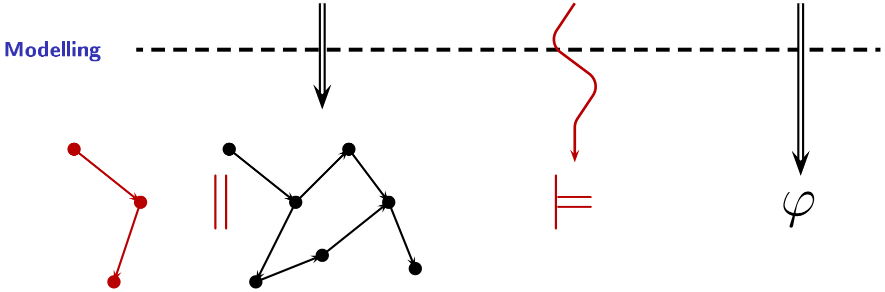
Modelling



Controller synthesis

Controller synthesis

Can we guide the system so that it satisfies the property?



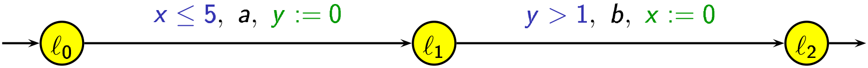
Controller synthesis

➜ modeled as two player games

Timed automata

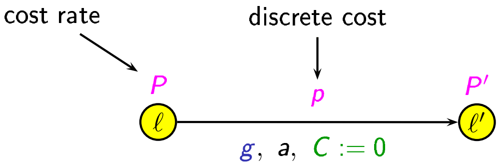
[Alur & Dill 90's]

x, y : clocks

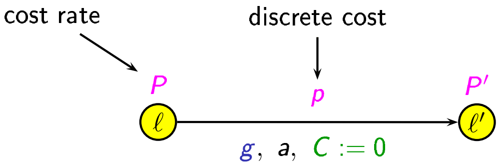


	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

Model of weighted timed automata [HSCC'01]



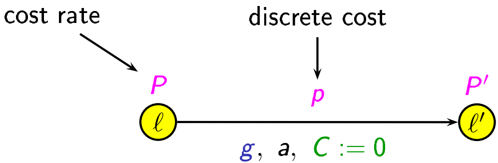
Model of weighted timed automata [HSCC'01]



- a configuration: (ℓ, v)
- two kinds of transitions:

$$\begin{cases} (\ell, v) \xrightarrow{\delta(d)} (\ell, v+d) \\ (\ell, v) \xrightarrow{a} (\ell', v') \text{ where } \begin{cases} v \models g \\ v' = [C \leftarrow 0]v \end{cases} \text{ for some } \ell \xrightarrow{g,a,C:=0} \ell' \end{cases}$$

Model of weighted timed automata [HSCC'01]



- a configuration: (ℓ, v)
- two kinds of transitions:

$$\begin{cases} (\ell, v) \xrightarrow{\delta(d)} (\ell, v+d) \\ (\ell, v) \xrightarrow{a} (\ell', v') \text{ where } \begin{cases} v \models g \\ v' = [C \leftarrow 0]v \end{cases} \text{ for some } \ell \xrightarrow{g,a,C:=0} \ell' \end{cases}$$

$$\mathrm{Cost}\Big((\ell, v) \xrightarrow{\delta(d)} (\ell, v+d)\Big) = P.d \qquad \mathrm{Cost}\Big((\ell, v) \xrightarrow{a} (\ell', v')\Big) = p$$

$\mathrm{Cost}(\rho)$ = accumulated cost along run ρ

An example

[Larsen, Behrmann, Brinksma, Fehnker, Hune, Pettersson, Romijn – CAV'01]

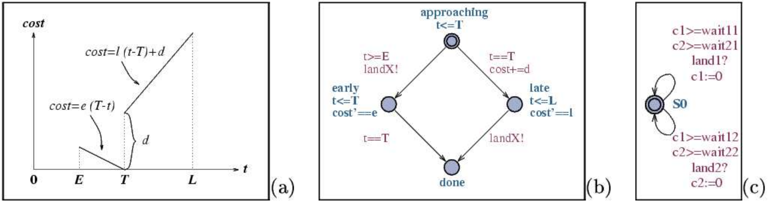
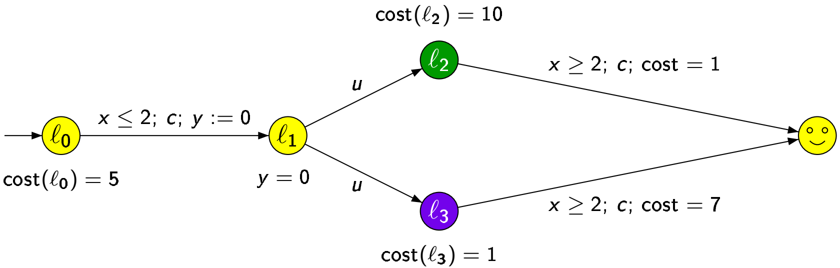
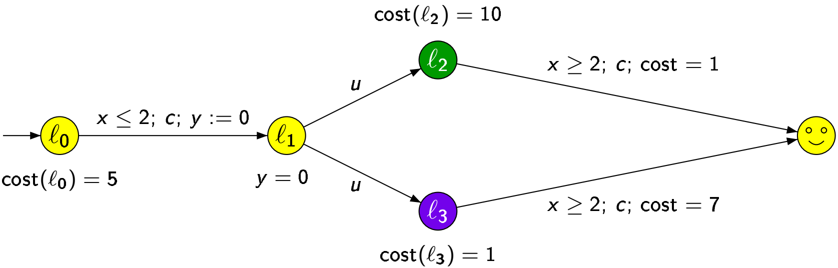


Fig. 2. Figure (a) depicts the cost of landing a plane at time t. Figure (b) shows an LPTA modelling the landing costs. Figure (c) shows an LPTA model of the runway.

An example

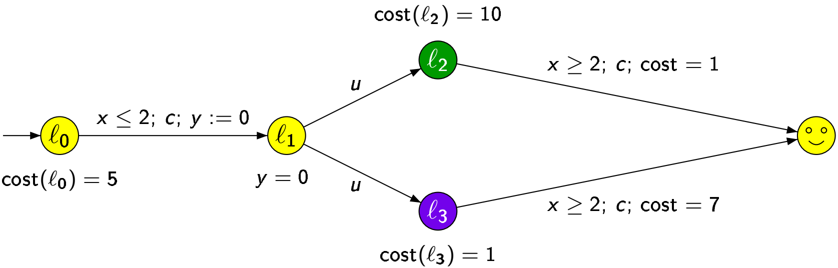

cost(ℓ_2) = 10
ℓ_2
$x \geq 2$; c; cost = 1
u
$x \leq 2$; c; $y := 0$
ℓ_0
ℓ_1
cost(ℓ_0) = 5
$y = 0$
u
ℓ_3
$x \geq 2$; c; cost = 7
cost(ℓ_3) = 1

An example



Question: what is the optimal cost for reaching the happy state?

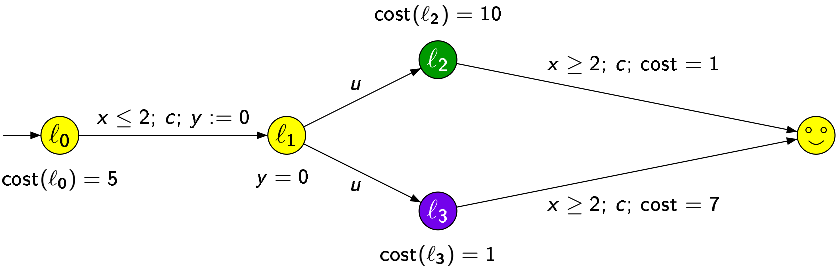
An example



Question: what is the optimal cost for reaching the happy state?

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

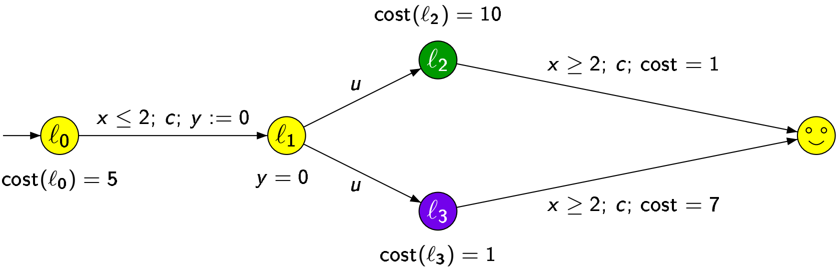
An example



Question: what is the optimal cost for reaching the happy state?

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

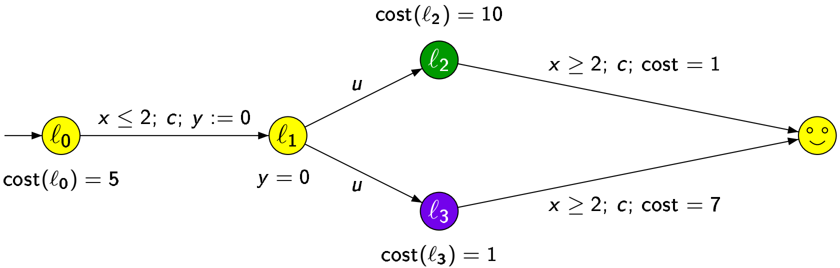
An example



Question: what is the optimal cost for reaching the happy state?

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

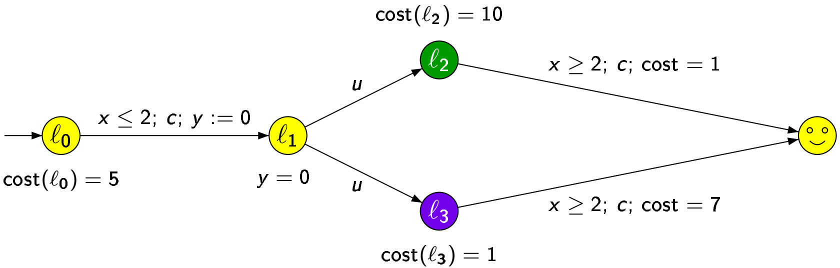
An example



Question: what is the optimal cost for reaching the happy state?

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

An example

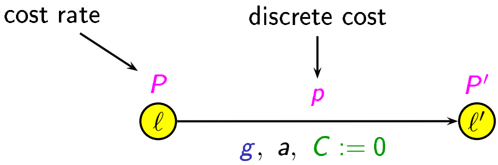


Question: what is the optimal cost for reaching the happy state?

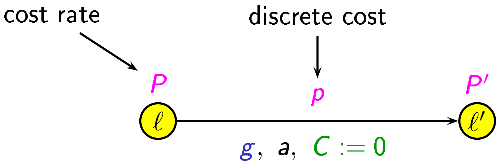
$$\inf_{0 \leq t \leq 2} \min \left(5t + 10(2-t) + 1 \,,\; 5t + (2-t) + 7 \right) = 9$$

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

Several issues on weighted timed automata

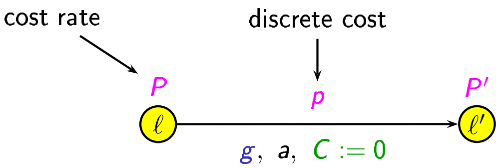


Several issues on weighted timed automata



- **Model-checking problems**
  - reachability with an optimization criterium on the cost
  - safety with a mean-cost optimization criterium
  - model-checking WCTL, an extension of CTL with cost constraints

Several issues on weighted timed automata



- **Model-checking problems**
  - reachability with an optimization criterium on the cost
  - safety with a mean-cost optimization criterium
  - model-checking WCTL, an extension of CTL with cost constraints
- **Optimal timed games**
  - optimal reachability timed games
  - optimal mean-cost timed games

Outline

Model-checking weighted timed automata

- Reachability with an optimization criterium on the cost

[Behrmann, Brinksma, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager – HSCC'01, TACAS'01, CAV'01]

[Alur, La Torre, Pappas – HSCC'01]

[Bouyer, Brihaye, Bruyère, Raskin – Subm.'06]

- Safety with a mean-cost optimization criterium

[Bouyer, Brinksma, Larsen – HSCC'04]

- Model-checking WCTL, an extension of CTL with cost constraints

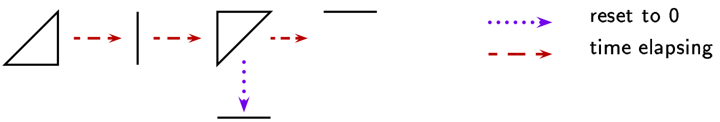
$$\mathbf{A}\,\mathbf{G}\,(\text{problem} \Rightarrow \mathbf{A}\,\mathbf{G}_{\leq 5}\,\text{repair})$$

[Brihaye, Bruyère, Raskin – FORMATS+FTRTFT'04]

[Bouyer, Brihaye, Markey – IPL'06]

[Bouyer, Laroussinie, Larsen, Markey, Rasmussen – Subm.'06]

The classical region abstraction



The corner-point abstraction

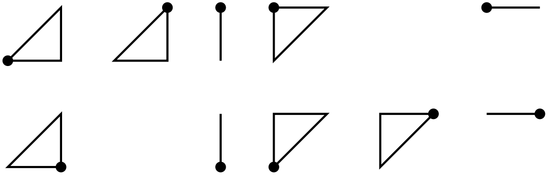
Idea: reduction to the discrete case

- **region abstraction:** not sufficient

The corner-point abstraction

Idea: reduction to the discrete case

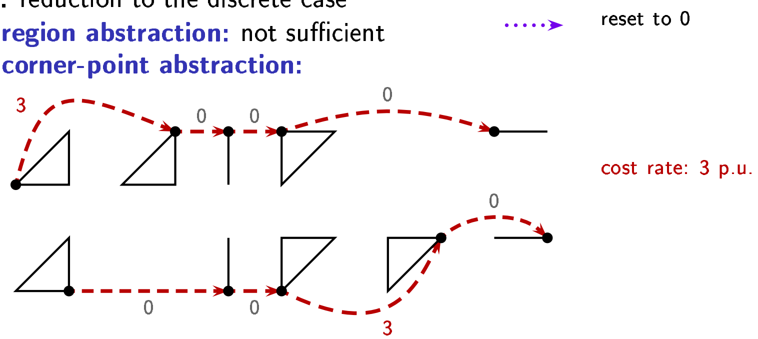
- **region abstraction:** not sufficient
- **corner-point abstraction:**



The corner-point abstraction

Idea: reduction to the discrete case

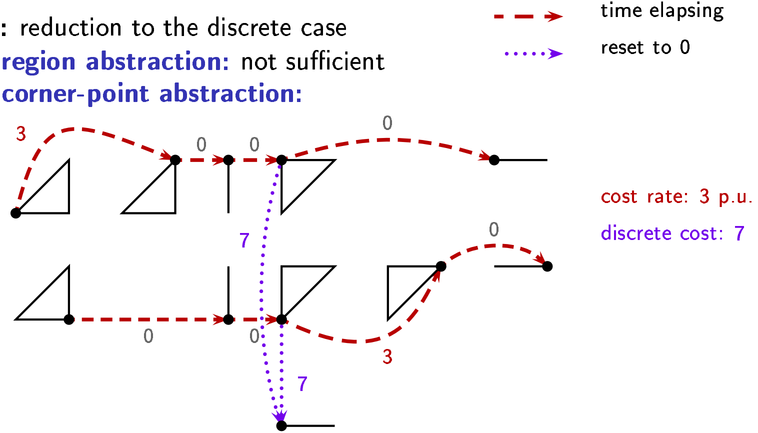
- **region abstraction:** not sufficient
- **corner-point abstraction:**



The corner-point abstraction

Idea: reduction to the discrete case

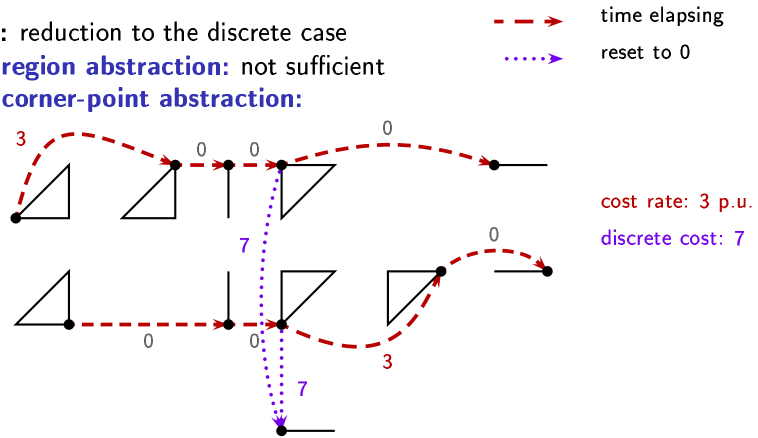
- **region abstraction:** not sufficient
- **corner-point abstraction:**



The corner-point abstraction

Idea: reduction to the discrete case

- **region abstraction:** not sufficient
- **corner-point abstraction:**



This abstraction is correct! ➜ PSPACE

- for computing optimal paths
- for computing optimal stationary behaviours

Outline

Decidability of timed games

Theorem **[Henzinger, Kopke 1999]**

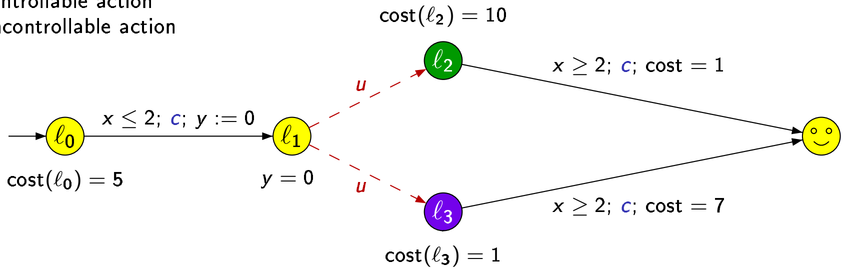
Safety and reachability control in timed automata are decidable and EXPTIME-complete.

(the attractor is computable...)

➜ classical regions are sufficient for solving such problems

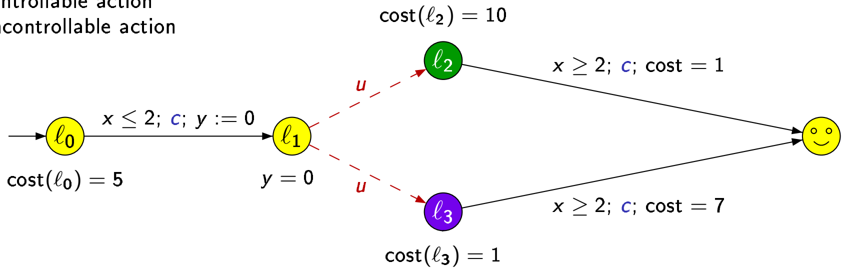
An example

c: controllable action
u: uncontrollable action



An example

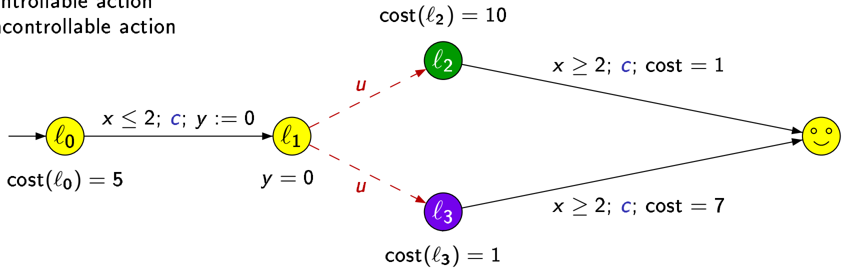
c: controllable action
u: uncontrollable action



Question: what is the optimal cost we can ensure in state ℓ_0?

An example

c: controllable action
u: uncontrollable action

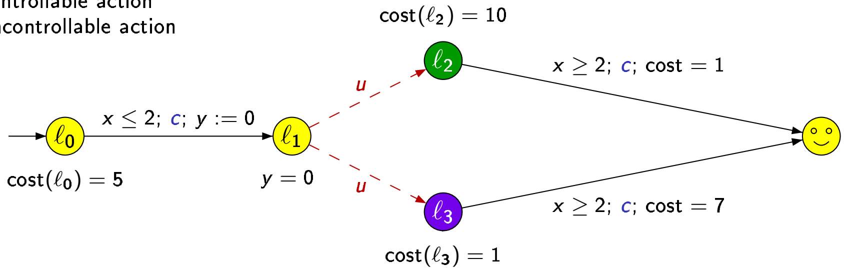


Question: what is the optimal cost we can ensure in state ℓ_0?

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

An example

c: controllable action
u: uncontrollable action

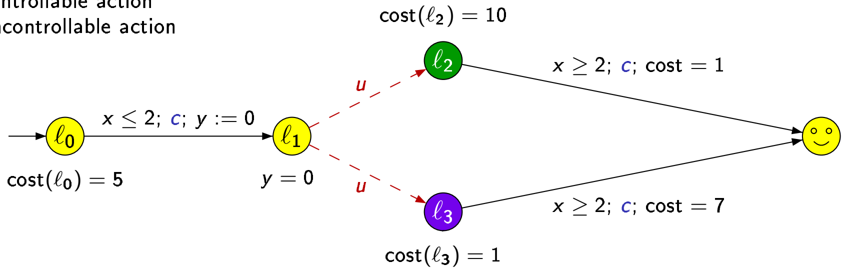


Question: what is the optimal cost we can ensure in state ℓ_0?

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

An example

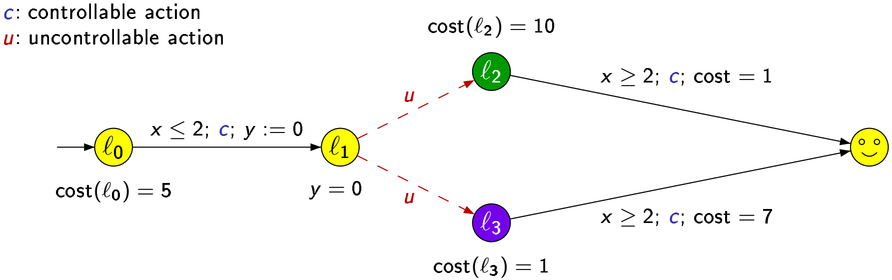
c: controllable action
u: uncontrollable action



Question: what is the optimal cost we can ensure in state ℓ_0?

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

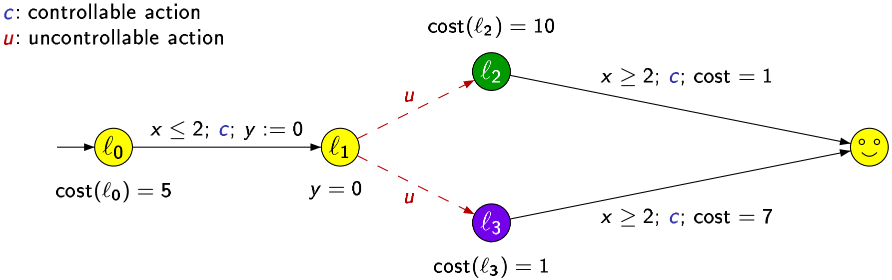
An example



Question: what is the optimal cost we can ensure in state ℓ_0?

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

An example



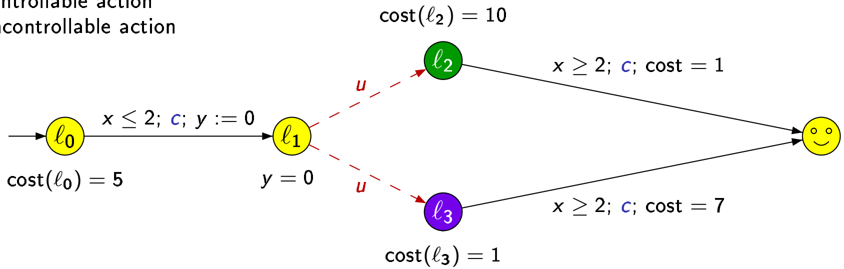
Question: what is the optimal cost we can ensure in state ℓ_0?

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

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

An example

c: controllable action
u: uncontrollable action



Question: what is the optimal cost we can ensure in state ℓ_0?

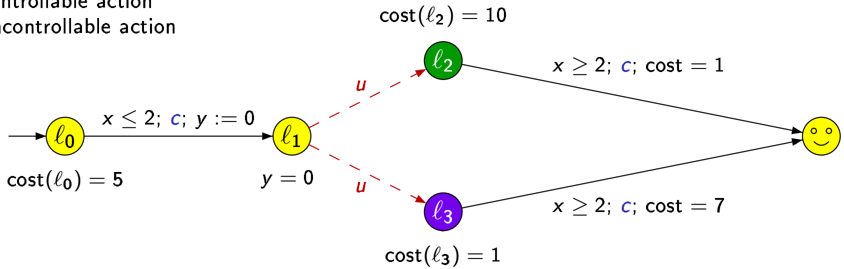
$$\inf_{0 \leq t \leq 2} \max \left(5t + 10(2-t) + 1 \,,\; 5t + (2-t) + 7 \right) = 14 + \frac{1}{3}$$

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

- How to automatically compute such optimal costs?

An example

c: controllable action
u: uncontrollable action



Question: what is the optimal cost we can ensure in state ℓ_0?

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

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

- How to automatically compute such optimal costs?
- How to synthesize optimal strategies (if one exists)?

A hot topic!

- [Asarin, Maler – HSCC'99]:
  - optimal time is computable in timed games

A hot topic!

- [Asarin, Maler – HSCC'99]:
  - optimal time is computable in timed games
- [La Torre, Mukhopadhyay, Murano – TCS@02]:
  - case of acyclic games

A hot topic!

- [Asarin, Maler – HSCC'99]:
  - optimal time is computable in timed games
- [La Torre, Mukhopadhyay, Murano – TCS@02]:
  - case of acyclic games
- [Alur, Bernadsky, Madhusudan – ICALP'04]:
  - complexity of k-step games
  - under a strongly non-*zeno* assumption, optimal cost is computable

A hot topic!

- [Asarin, Maler – HSCC'99]:
  - optimal time is computable in timed games
- [La Torre, Mukhopadhyay, Murano – TCS@02]:
  - case of acyclic games
- [Alur, Bernadsky, Madhusudan – ICALP'04]:
  - complexity of k-step games
  - under a strongly non-*zeno* assumption, optimal cost is computable
- [Bouyer, Cassez, Fleury, Larsen – FSTTCS'04]:
  - structural properties of strategies (e.g. memory)
  - under a strongly non-*zeno* assumption, optimal cost is computable

A hot topic!

- [Asarin, Maler – HSCC'99]:
  - optimal time is computable in timed games
- [La Torre, Mukhopadhyay, Murano – TCS@02]:
  - case of acyclic games
- [Alur, Bernadsky, Madhusudan – ICALP'04]:
  - complexity of k-step games
  - under a strongly non-*zeno* assumption, optimal cost is computable
- [Bouyer, Cassez, Fleury, Larsen – FSTTCS'04]:
  - structural properties of strategies (e.g. memory)
  - under a strongly non-*zeno* assumption, optimal cost is computable
- [Brihaye, Bruyère, Raskin – FORMATS'05]:
  - with five clocks, optimal cost is not computable!
  - with one clock and one stopwatch cost, optimal cost is computable

A hot topic!

- [Asarin, Maler – HSCC'99]:
  - optimal time is computable in timed games
- [La Torre, Mukhopadhyay, Murano – TCS@02]:
  - case of acyclic games
- [Alur, Bernadsky, Madhusudan – ICALP'04]:
  - complexity of k-step games
  - under a strongly non-*zeno* assumption, optimal cost is computable
- [Bouyer, Cassez, Fleury, Larsen – FSTTCS'04]:
  - structural properties of strategies (e.g. memory)
  - under a strongly non-*zeno* assumption, optimal cost is computable
- [Brihaye, Bruyère, Raskin – FORMATS'05]:
  - with five clocks, optimal cost is not computable!
  - with one clock and one stopwatch cost, optimal cost is computable
- [Bouyer, Brihaye, Markey – IPL'06]:
  - with three clocks, optimal cost is not computable

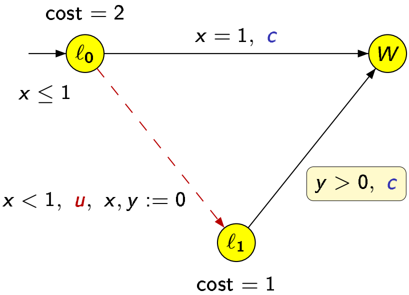
A hot topic!

- [Asarin, Maler – HSCC'99]:
  - optimal time is computable in timed games
- [La Torre, Mukhopadhyay, Murano – TCS@02]:
  - case of acyclic games
- [Alur, Bernadsky, Madhusudan – ICALP'04]:
  - complexity of k-step games
  - under a strongly non-*zeno* assumption, optimal cost is computable
- [Bouyer, Cassez, Fleury, Larsen – FSTTCS'04]:
  - structural properties of strategies (e.g. memory)
  - under a strongly non-*zeno* assumption, optimal cost is computable
- [Brihaye, Bruyère, Raskin – FORMATS'05]:
  - with five clocks, optimal cost is not computable!
  - with one clock and one stopwatch cost, optimal cost is computable
- [Bouyer, Brihaye, Markey – IPL'06]:
  - with three clocks, optimal cost is not computable
- [Bouyer, Laroussinie, Larsen, Markey, Rasmussen – Subm.'06]:
  - with one clock, optimal cost is computable

A hot topic!

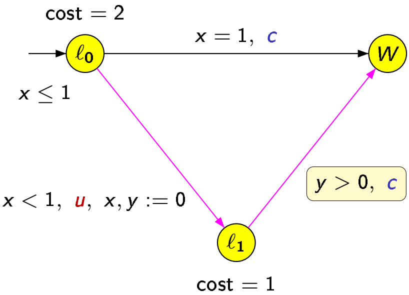
- [Asarin, Maler – HSCC'99]:
  - optimal time is computable in timed games
- [La Torre, Mukhopadhyay, Murano – TCS@02]:
  - case of acyclic games
- [Alur, Bernadsky, Madhusudan – ICALP'04]:
  - complexity of k-step games
  - under a strongly non-*zeno* assumption, optimal cost is computable
- [Bouyer, Cassez, Fleury, Larsen – FSTTCS'04]:
  - structural properties of strategies (e.g. memory)
  - under a strongly non-*zeno* assumption, optimal cost is computable
- [Brihaye, Bruyère, Raskin – FORMATS'05]:
  - with five clocks, optimal cost is not computable!
  - with one clock and one stopwatch cost, optimal cost is computable
- [Bouyer, Brihaye, Markey – IPL'06]:
  - with three clocks, optimal cost is not computable
- [Bouyer, Laroussinie, Larsen, Markey, Rasmussen – Subm.'06]:
  - with one clock, optimal cost is computable
- [Jurdziński, Trivedi – LICS'06]:
  - optimal mean-cost is computable in a (restrictive) case

Memoryless strategies are not powerful enough



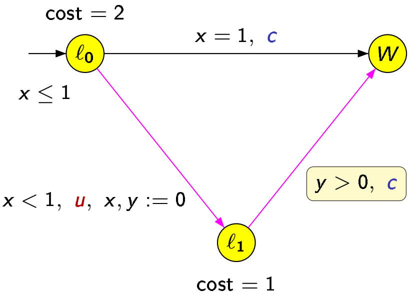
- **optimal cost:** 2
- **optimal strategy:**

Memoryless strategies are not powerful enough



- **optimal cost:** 2
- **optimal strategy:** if d is the time before a u occurs, and d' is the time waited in ℓ_1, the cost of the run is $2.d + d'$.

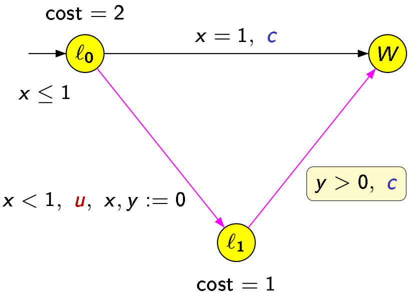
Memoryless strategies are not powerful enough



- **optimal cost:** 2
- **optimal strategy:** if d is the time before a u occurs, and d' is the time waited in ℓ_1, the cost of the run is $2.d + d'$.

$$2.d + d' \leq 2$$

Memoryless strategies are not powerful enough



- **optimal cost:** 2
- **optimal strategy:** if d is the time before a u occurs, and d' is the time waited in ℓ_1, the cost of the run is $2.d + d'$.

$$2.d + d' \leq 2$$

$$(\text{accumulated cost}) + d' \leq 2$$

Undecidability – Shape of the reduction

Original reduction: [Brihaye, Bruyère, Raskin – FORMATS'05]

This reduction: [Bouyer, Brihaye, Markey – IPL'06]

Undecidability – Shape of the reduction

Original reduction: [Brihaye, Bruyère, Raskin – FORMATS'05]

This reduction: [Bouyer, Brihaye, Markey – IPL'06]

Simulation of a two-counter machine:

- player 1 simulates the two-counter machine
- player 2 checks that player 1 does not cheat

Undecidability – Shape of the reduction

Original reduction: [Brihaye, Bruyère, Raskin – FORMATS'05]

This reduction: [Bouyer, Brihaye, Markey – IPL'06]

Simulation of a two-counter machine:

- player 1 simulates the two-counter machine
- player 2 checks that player 1 does not cheat

Encoding of the counters:

- counter c_1 is encoded by a clock x_1 s.t. $x_1 = \frac{1}{2^{c_1}}$
- counter c_2 is encoded by a clock x_2 s.t. $x_2 = \frac{1}{3^{c_2}}$
- x_1 and x_2 will be alternatively x, y or z

Undecidability – Shape of the reduction

Original reduction: [Brihaye, Bruyère, Raskin – FORMATS'05]

This reduction: [Bouyer, Brihaye, Markey – IPL'06]

Simulation of a two-counter machine:

- player 1 simulates the two-counter machine
- player 2 checks that player 1 does not cheat

Encoding of the counters:

- counter c_1 is encoded by a clock x_1 s.t. $x_1 = \frac{1}{2^{c_1}}$
- counter c_2 is encoded by a clock x_2 s.t. $x_2 = \frac{1}{3^{c_2}}$
- x_1 and x_2 will be alternatively x, y or z

The aim of player 1 is to win (reach a W-state) with cost ≤ 3,

Undecidability – Shape of the reduction

Original reduction: [Brihaye, Bruyère, Raskin – FORMATS'05]
This reduction: [Bouyer, Brihaye, Markey – IPL'06]

Simulation of a two-counter machine:

- player 1 simulates the two-counter machine
- player 2 checks that player 1 does not cheat

Encoding of the counters:

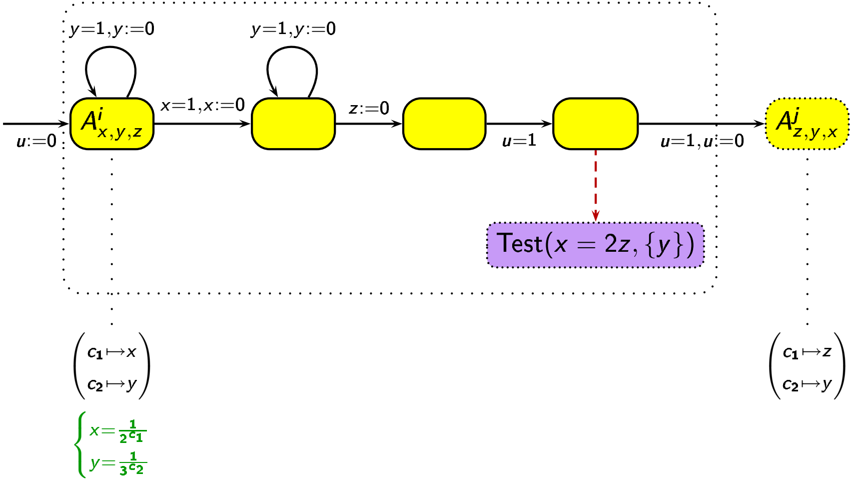
- counter c_1 is encoded by a clock x_1 s.t. $x_1 = \frac{1}{2^{c_1}}$
- counter c_2 is encoded by a clock x_2 s.t. $x_2 = \frac{1}{3^{c_2}}$
- x_1 and x_2 will be alternatively x, y or z

The aim of player 1 is to win (reach a W-state) with cost ≤ 3, and

> Player 1 has a winning strategy with cost ≤ 3
> iff
> the two-counter machine halts

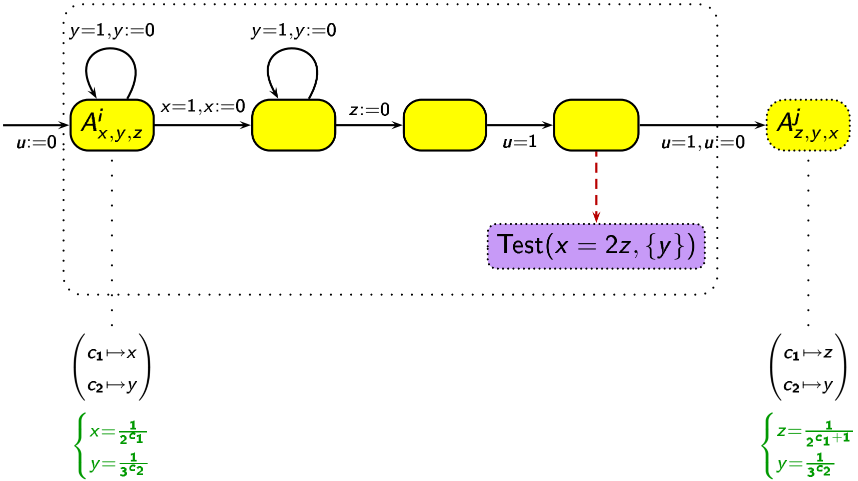
Simulation of an incrementation

Instruction i: c_1++; goto instruction j

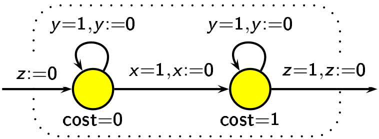


Simulation of an incrementation

Instruction i: $c_1 ++$; goto instruction j

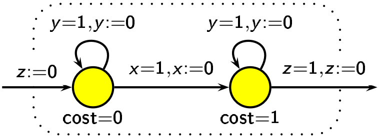


Adding x or $1 - x$ to the cost variable

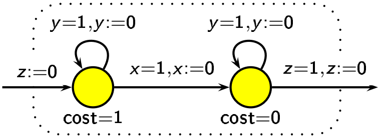


The cost is increased by x_0

Adding x or $1 - x$ to the cost variable



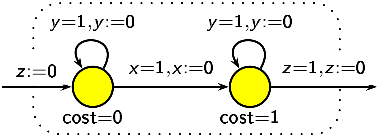
The cost is increased by x_0



The cost is increased by $1 - x_0$

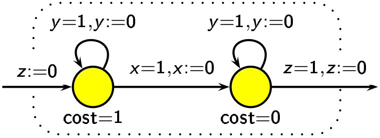
Adding x or $1 - x$ to the cost variable

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



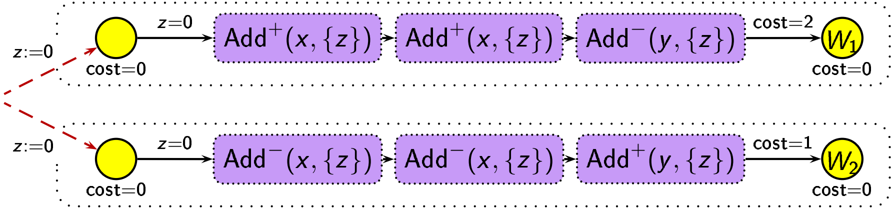
The cost is increased by x_0

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



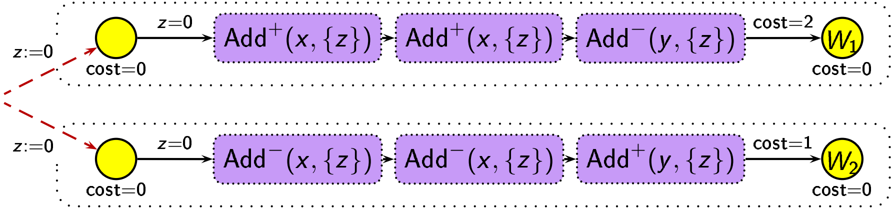
The cost is increased by $1 - x_0$

Checking $y = 2x$



In W_1, cost $= 2x_0 + (1 - y_0) + 2$.
In W_2, cost $= 2(1 - x_0) + y_0 + 1$.

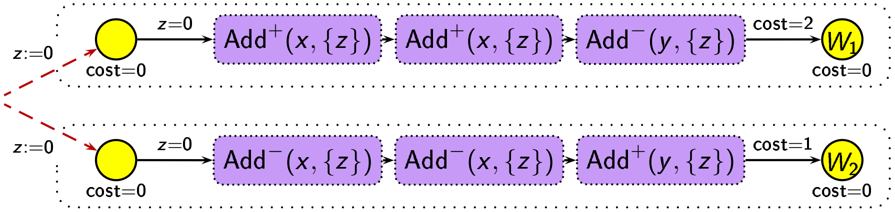
Checking $y = 2x$



In W_1, cost $= 2x_0 + (1 - y_0) + 2$.
In W_2, cost $= 2(1 - x_0) + y_0 + 1$.

- if $y_0 < 2x_0$, player 2 chooses the first branch: in W_1, cost > 3

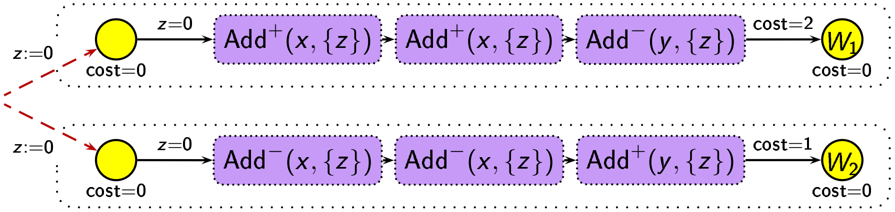
Checking $y = 2x$



In W_1, cost $= 2x_0 + (1 - y_0) + 2$.
In W_2, cost $= 2(1 - x_0) + y_0 + 1$.

- if $y_0 < 2x_0$, player 2 chooses the first branch: in W_1, cost > 3
- if $y_0 > 2x_0$, player 2 chooses the second branch: in W_2, cost > 3

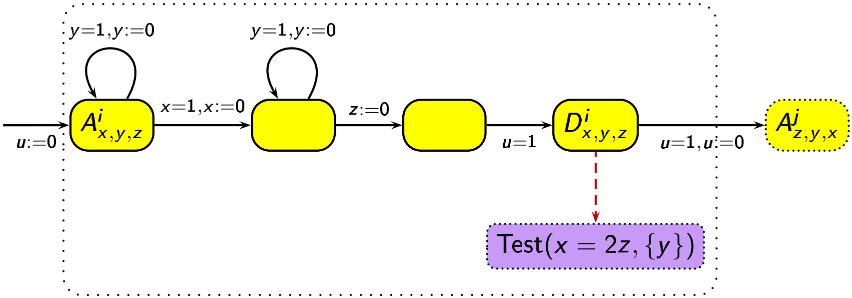
Checking $y = 2x$



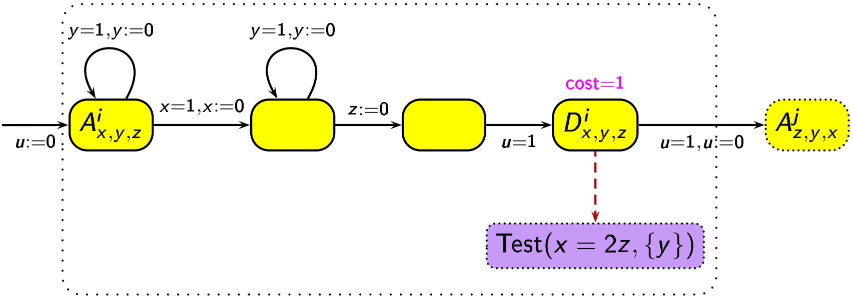
In W_1, cost $= 2x_0 + (1 - y_0) + 2$.
In W_2, cost $= 2(1 - x_0) + y_0 + 1$.

- if $y_0 < 2x_0$, player 2 chooses the first branch: in W_1, cost > 3
- if $y_0 > 2x_0$, player 2 chooses the second branch: in W_2, cost > 3
- if $y_0 = 2x_0$, in W_1 or in W_2, cost $= 3$.

How to get rid of tick clock u?

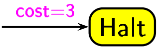


How to get rid of tick clock u?

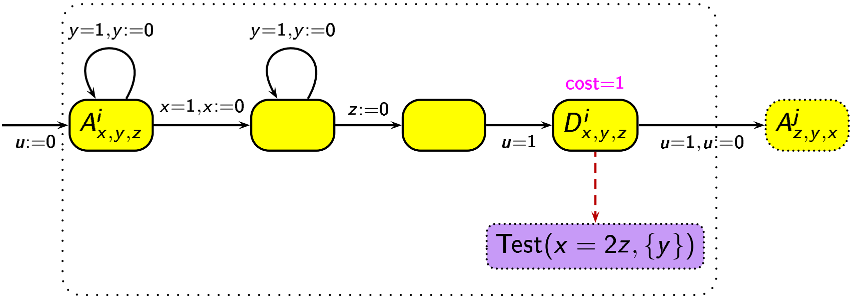


We will ensure that:

- no cost is accumulated in D-states

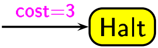


How to get rid of tick clock u?

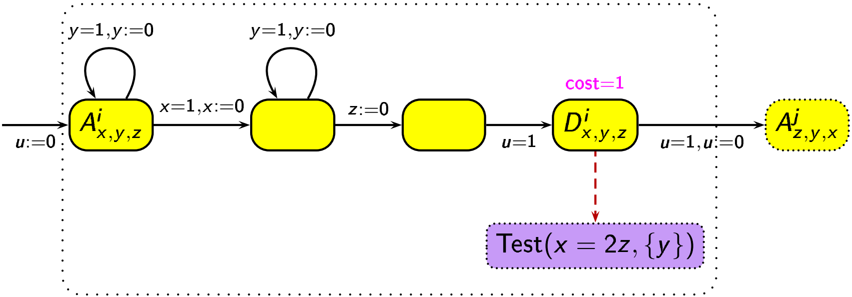


We will ensure that:

- no cost is accumulated in D-states
- the delay between the A-state and the D-state is 1 t.u.

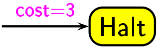


How to get rid of tick clock u?

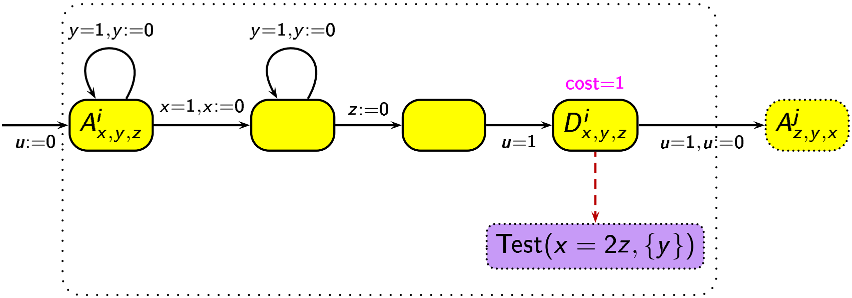


We will ensure that:

- no cost is accumulated in D-states
- the delay between the A-state and the D-state is 1 t.u.
  - the value of x in D is of the form $\frac{1}{2^n}$

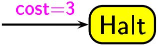


How to get rid of tick clock u?

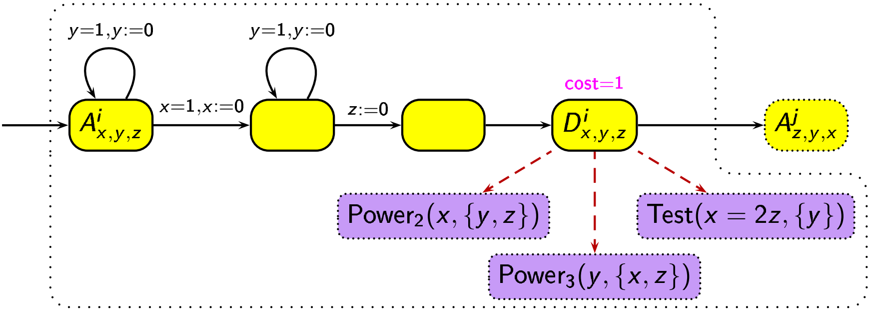


We will ensure that:

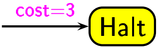
- no cost is accumulated in D-states
- the delay between the A-state and the D-state is 1 t.u.
  - the value of x in D is of the form $\frac{1}{2^n}$
  - the value of y in D is of the form $\frac{1}{3^m}$



How to get rid of tick clock u?

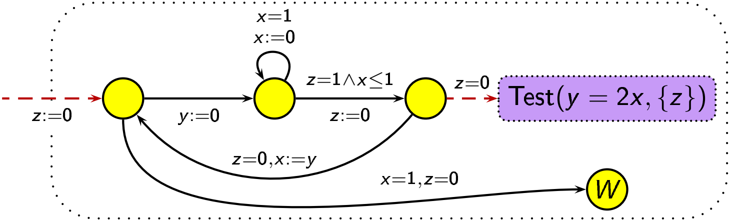


We will ensure that:



- no cost is accumulated in D-states
- the delay between the A-state and the D-state is 1 t.u.
  - the value of x in D is of the form $\frac{1}{2^n}$
  - the value of y in D is of the form $\frac{1}{3^m}$

Checking that x is of the form $\frac{1}{2^n}$



Outline

Conclusion and further work

Model-checking

- "basic" properties are decidable
- efficient symbolic computations have even been proposed

  ➜ implemented in tool Uppaal Cora
- branching-time properties are undecidable

Conclusion and further work

Model-checking

- "basic" properties are decidable
- efficient symbolic computations have even been proposed
  ➜ implemented in tool Uppaal Cora
- branching-time properties are undecidable
- what about linear-time properties?
- consider more general cost functions

Conclusion and further work

Model-checking

- "basic" properties are decidable
- efficient symbolic computations have even been proposed

  ➜ implemented in tool Uppaal Cora
- branching-time properties are undecidable
- what about linear-time properties?
- consider more general cost functions

Optimal timed games

- optimal cost is in general not computable in timed games
- under some assumption, it becomes computable
- complexity issues and properties of strategies have also been studied

Conclusion and further work

Model-checking

- “basic” properties are decidable
- efficient symbolic computations have even been proposed
  ➜ implemented in tool Uppaal Cora
- branching-time properties are undecidable
- what about linear-time properties?
- consider more general cost functions

Optimal timed games

- optimal cost is in general not computable in timed games
- under some assumption, it becomes computable
- complexity issues and properties of strategies have also been studied
- investigate further mean-cost optimal timed games
- approximate optimal cost
- propose more algorithmics solutions
- ...