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

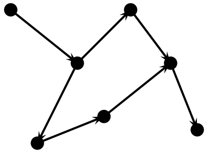
Outline

1. Introduction
2. Model-checking weighted timed automata
3. Optimal timed games
4. Conclusion

Model-checking

Does the system satisfy the property?

Modelling

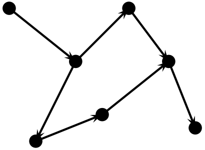


φ

Model-checking

Does the system satisfy the property?

Modelling



$\models$

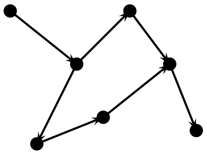
Model-checking Algorithm

φ

Controller synthesis

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

Modelling

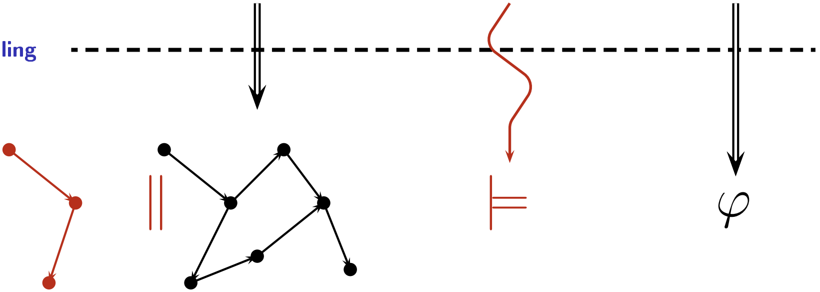


φ

Controller synthesis

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

Modelling

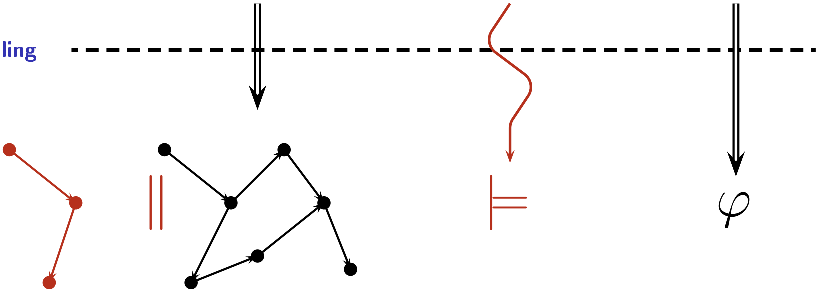


Controller synthesis

Controller synthesis

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

Modelling



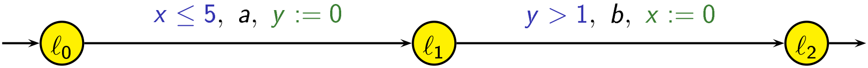
Controller synthesis

➜ modeled as two player games

Timed automata

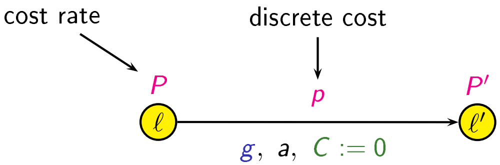
[Alur & Dill 90's]

x, y : clocks

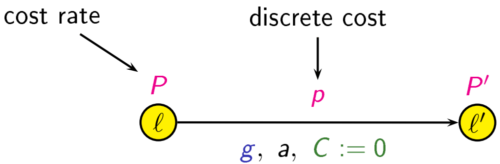


$$
\begin{array}{ccccccccc}
 & \ell_0 & \xrightarrow{\delta(4.1)} & \ell_0 & \xrightarrow{a} & \ell_1 & \xrightarrow{\delta(1.4)} & \ell_1 & \xrightarrow{b} & \ell_2 \\
x & 0 & & 4.1 & & 4.1 & & 5.5 & & 0 \\
y & 0 & & 4.1 & & 0 & & 1.4 & & 1.4
\end{array}
$$

Model of weighted/priced timed automata [HSCC'01]



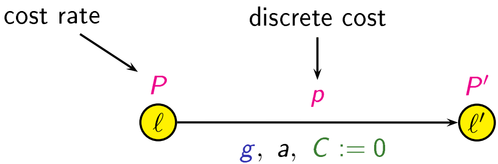
Model of weighted/priced timed automata [HSCC'01]



- 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/priced timed automata [HSCC'01]



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

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

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

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

An example

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

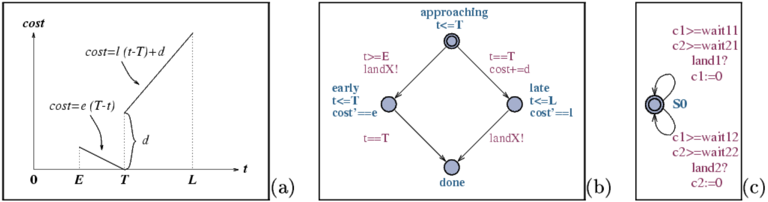
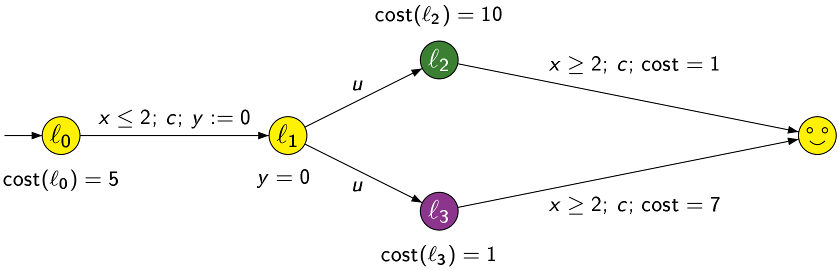
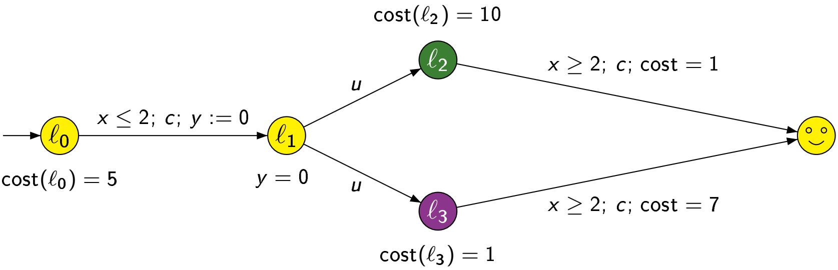


Fig. 2. Figure (a) depicts the cost of landing a plane at time t. Figure (b) shows an LPTA modelling the landing costs. Figure (c) shows an LPTA model of the runway.

An example

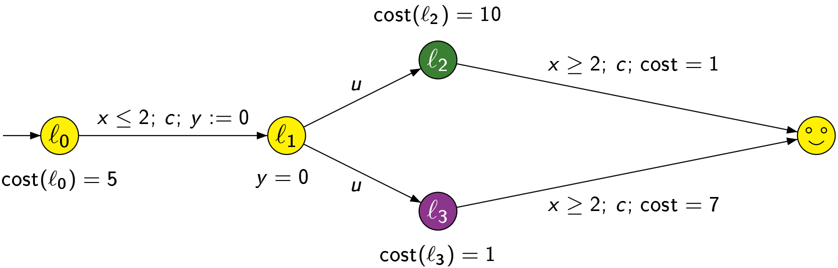

cost(ℓ_2) = 10
ℓ_2
$x \geq 2$; c; cost = 1
u
$x \leq 2$; c; $y := 0$
ℓ_0
ℓ_1
cost(ℓ_0) = 5
$y = 0$
u
ℓ_3
$x \geq 2$; c; cost = 7
cost(ℓ_3) = 1

An example



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

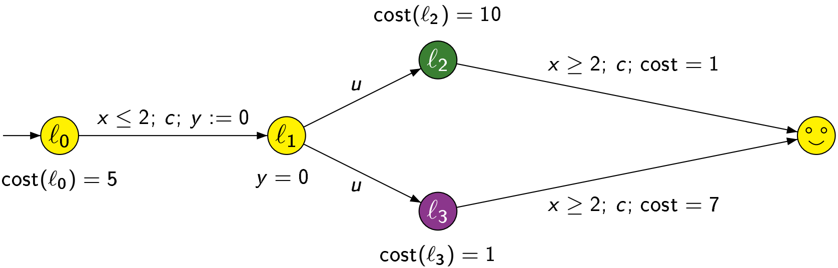
An example



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

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

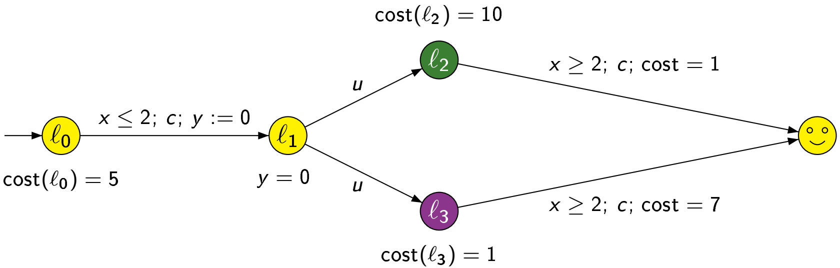
An example



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

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

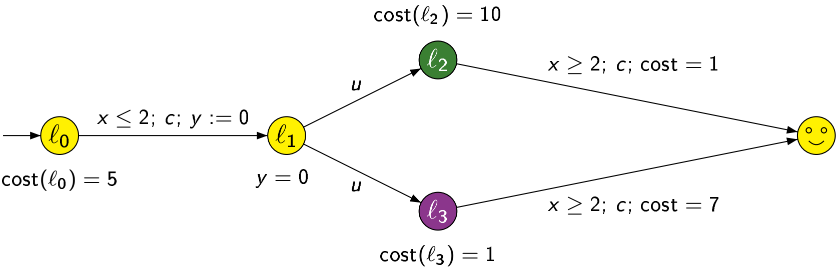
An example



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

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

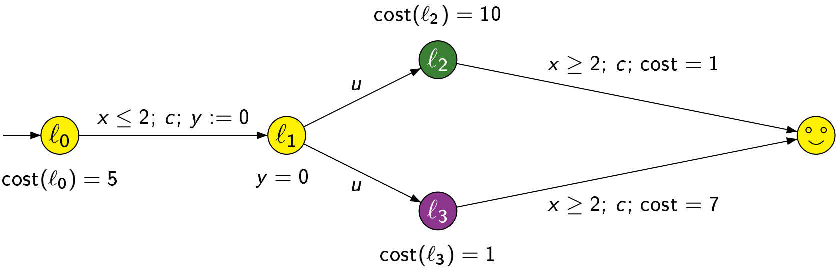
An example



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

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

An example

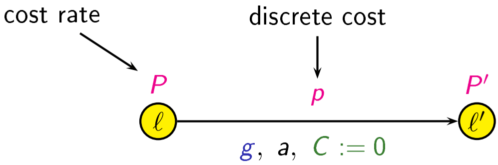


Question: what is the optimal cost for reaching the happy state?

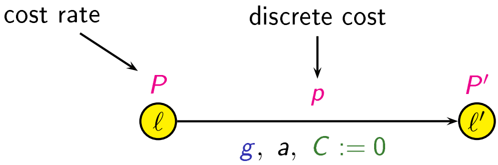
$$\inf_{0 \leq t \leq 2} \min\left(5t + 10(2-t) + 1 \,,\; 5t + (2-t) + 7 \right) = 9$$

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

Several issues on weighted timed automata

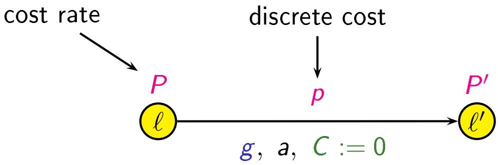


Several issues on weighted timed automata



- **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. 2006]

- Safety with a mean-cost optimization criterium

  [Bouyer, Brinksma, Larsen – HSCC'04]

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

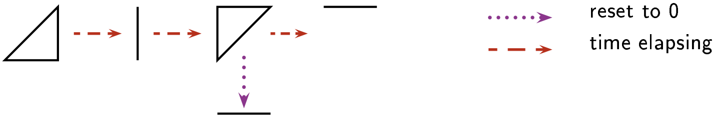
$$\mathbf{A}\,\mathbf{G}\,(\text{problem} \Rightarrow \mathbf{A}\,\mathbf{G}_{\leq 5}\,\text{repair})$$

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

  [Bouyer, Brihaye, Markey – IPL'06]

  [Bouyer, Laroussinie, Larsen, Markey, Rasmussen – 2006]

The classical region abstraction



The corner-point abstraction

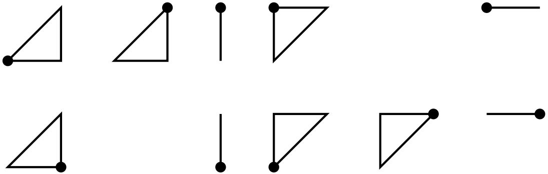
Idea: reduction to the discrete case

- **region abstraction:** not sufficient

The corner-point abstraction

Idea: reduction to the discrete case

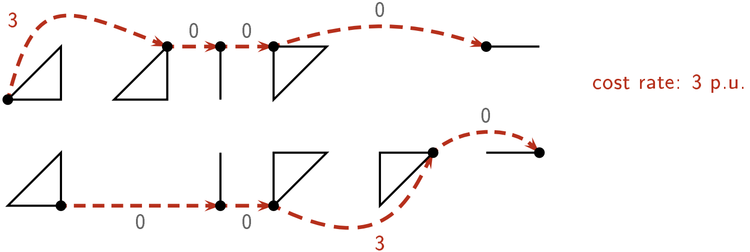
- **region abstraction:** not sufficient
- **corner-point abstraction/weighted discrete graph $\mathcal{A}_{\mathsf{cp}}$:**



The corner-point abstraction

Idea: reduction to the discrete case

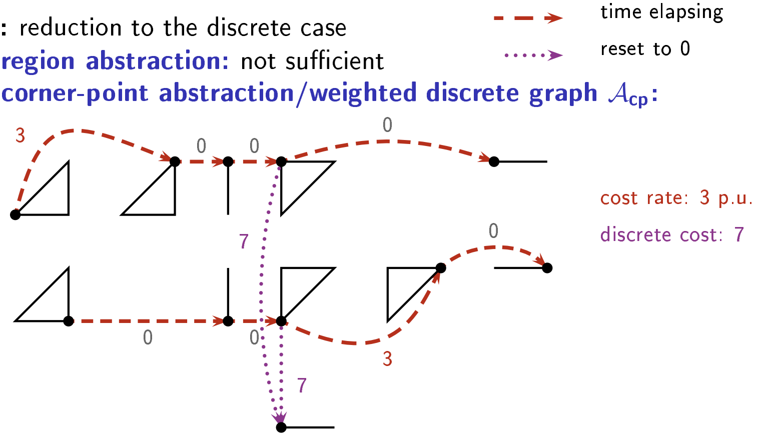
- **region abstraction:** not sufficient
- **corner-point abstraction/weighted discrete graph $\mathcal{A}_{\mathsf{cp}}$:**



The corner-point abstraction

Idea: reduction to the discrete case

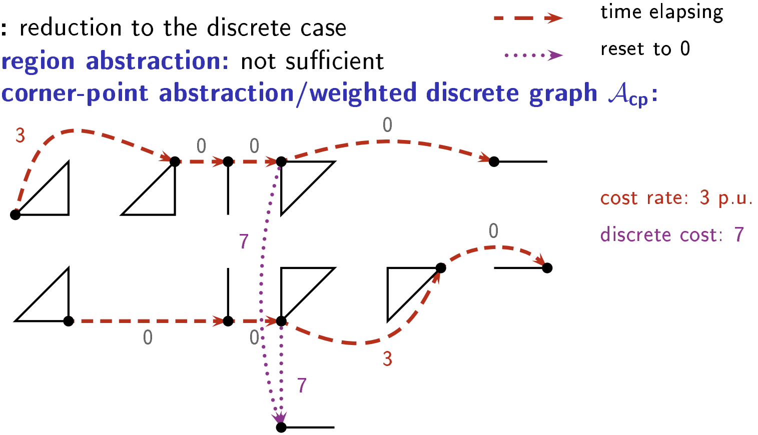
- **region abstraction:** not sufficient
- **corner-point abstraction/weighted discrete graph $\mathcal{A}_{cp}$:**



The corner-point abstraction

Idea: reduction to the discrete case

- **region abstraction:** not sufficient
- **corner-point abstraction/weighted discrete graph $\mathcal{A}_{cp}$:**



This abstraction is correct! ➜ PSPACE

- for computing optimal paths
- for computing optimal stationary behaviours

Optimal reachability

➜ optimal reachability along a given path can be viewed as a linear programming problem

Lemma

Let Z be a bounded zone and f be a function

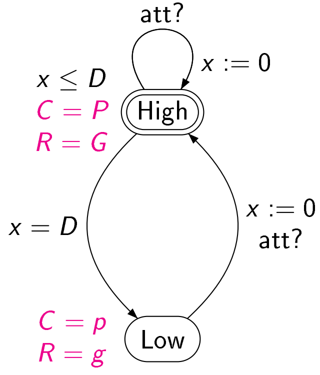
$$f : (t_1, \ldots, t_n) \mapsto \sum_{i=1}^{n} c_i\, t_i + c$$

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

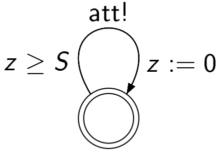
Then, abstract paths in $\mathcal{A}_{\mathrm{cp}}$ can be approximated by real path "ε-close" to the abstract path

Infinite stationary behaviours: An example

A production system:



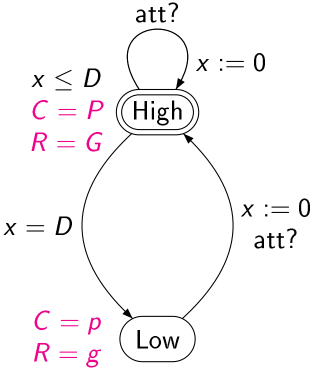
Single machine $M(D, G, P, g, p)$



Operator $O(S)$

Infinite stationary behaviours: An example

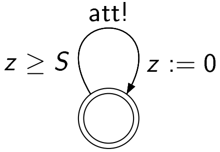
A production system:



Single machine $M(D, G, P, g, p)$

Question: How to minimize

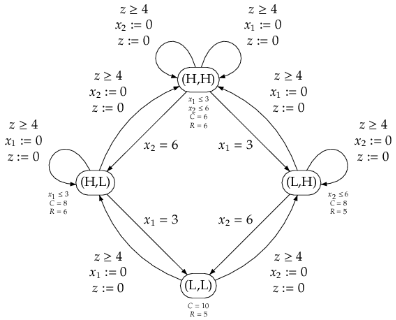
$$\lim_{n \to +\infty} \frac{accumulated\ \ cost(n)}{accumulated\ \ reward(n)}?$$



Operator $O(S)$

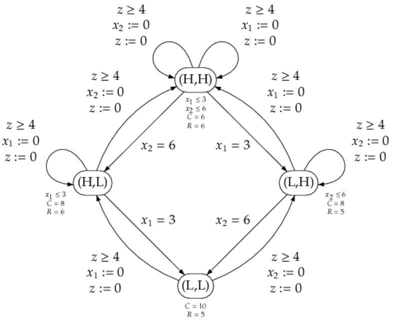
Infinite stationary behaviours: An example

Two machines $M_1(D = 3, P = 3, G = 4, p = 5, g = 3)$, $M_2(D = 6, P = 3, G = 2, p = 5, g = 2)$ and an Operator $O(4)$.



Infinite stationary behaviours: An example

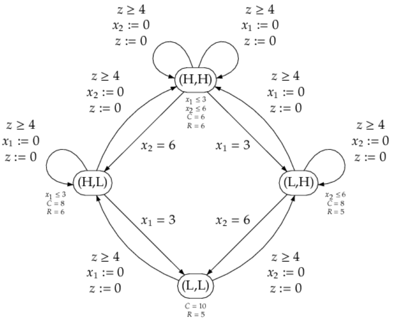
Two machines $M_1(D = 3, P = 3, G = 4, p = 5, g = 3)$,
$M_2(D = 6, P = 3, G = 2, p = 5, g = 2)$ and an Operator $O(4)$.



$((H, H), x_1 = x_2 = z = 0)$

Infinite stationary behaviours: An example

Two machines $M_1(D = 3, P = 3, G = 4, p = 5, g = 3)$, $M_2(D = 6, P = 3, G = 2, p = 5, g = 2)$ and an Operator $O(4)$.

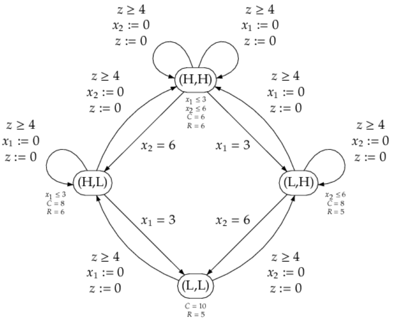


$$((H, H), x_1 = x_2 = z = 0)$$

$$\xrightarrow{18,18} ((L, H), x_1 = x_2 = z = 3)$$

Infinite stationary behaviours: An example

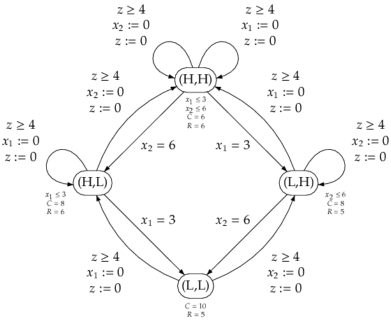
Two machines $M_1(D = 3, P = 3, G = 4, p = 5, g = 3)$, $M_2(D = 6, P = 3, G = 2, p = 5, g = 2)$ and an Operator $O(4)$.



$$((H, H), x_1 = x_2 = z = 0)$$
$$\xrightarrow{18,18} ((L, H), x_1 = x_2 = z = 3)$$
$$\xrightarrow{8,5} ((L, H), x_1 = x_2 = z = 4)$$

Infinite stationary behaviours: An example

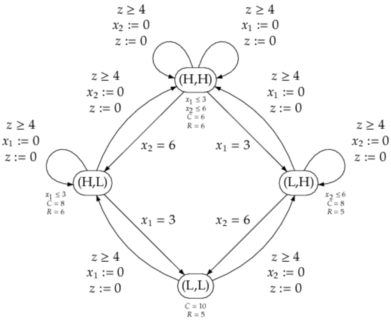
Two machines $M_1(D = 3, P = 3, G = 4, p = 5, g = 3)$,
$M_2(D = 6, P = 3, G = 2, p = 5, g = 2)$ and an Operator $O(4)$.



$$
\begin{aligned}
& ((H, H), x_1 = x_2 = z = 0) \\
\xrightarrow{18,18}\ & ((L, H), x_1 = x_2 = z = 3) \\
\xrightarrow{8,5}\ & ((L, H), x_1 = x_2 = z = 4) \\
\longrightarrow\ & ((H, H), x_1 = z = 0, x_2 = 4)
\end{aligned}
$$

Infinite stationary behaviours: An example

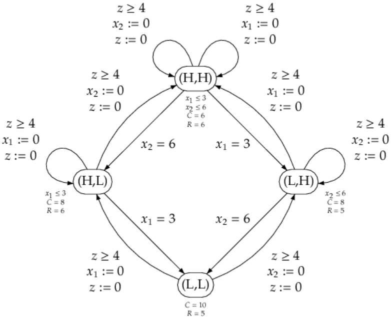
Two machines $M_1(D = 3, P = 3, G = 4, p = 5, g = 3)$, $M_2(D = 6, P = 3, G = 2, p = 5, g = 2)$ and an Operator $O(4)$.



$$
\begin{aligned}
& ((H,H), x_1 = x_2 = z = 0) \\
\xrightarrow{18,18}\ & ((L,H), x_1 = x_2 = z = 3) \\
\xrightarrow{8,5}\ & ((L,H), x_1 = x_2 = z = 4) \\
\longrightarrow\ & ((H,H), x_1 = z = 0, x_2 = 4) \\
\xrightarrow{12,12}\ & ((H,L), x_1 = z = 2, x_2 = 6) \\
\xrightarrow{8,6}\ & ((L,L), x_1 = z = 3, x_2 = 7) \\
\xrightarrow{10,5}\ & ((L,L), x_1 = z = 4, x_2 = 8) \\
\longrightarrow\ & ((H,L), x_1 = z = 0, x_2 = 8) \\
\xrightarrow{24,18}\ & ((L,L), x_1 = z = 3, x_2 = 11) \\
\xrightarrow{10,5}\ & ((L,L), x_1 = z = 4, x_2 = 12) \\
\longrightarrow\ & ((L,H), x_1 = 4, x_2 = z = 0) \\
\xrightarrow{32,20}\ & ((L,H), x_1 = 8, x_2 = z = 4) \\
\longrightarrow\ & ((H,H), x_1 = z = 0, x_2 = 4)
\end{aligned}
$$

Infinite stationary behaviours: An example

Two machines $M_1(D = 3, P = 3, G = 4, p = 5, g = 3)$, $M_2(D = 6, P = 3, G = 2, p = 5, g = 2)$ and an Operator $O(4)$.

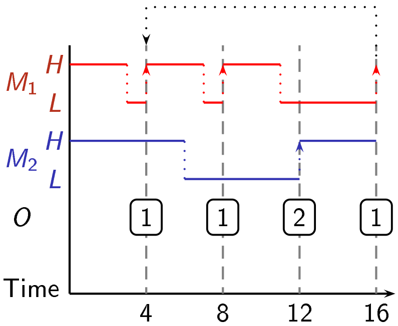


limit $\frac{cost}{reward} = \frac{96}{66} \simeq 1,455$

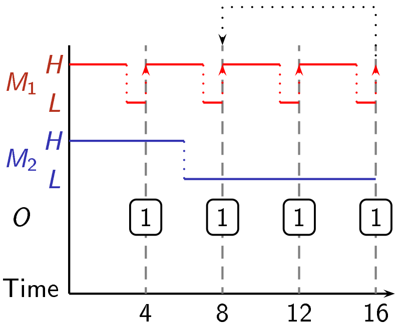
$$
\begin{aligned}
& ((H,H), x_1 = x_2 = z = 0) \\
\xrightarrow{18,18}\ & ((L,H), x_1 = x_2 = z = 3) \\
\xrightarrow{8,5}\ & ((L,H), x_1 = x_2 = z = 4) \\
\longrightarrow\ & ((H,H), x_1 = z = 0, x_2 = 4) \\
\xrightarrow{12,12}\ & ((H,L), x_1 = z = 2, x_2 = 6) \\
\xrightarrow{8,6}\ & ((L,L), x_1 = z = 3, x_2 = 7) \\
\xrightarrow{10,5}\ & ((L,L), x_1 = z = 4, x_2 = 8) \\
\longrightarrow\ & ((H,L), x_1 = z = 0, x_2 = 8) \\
\xrightarrow{24,18}\ & ((L,L), x_1 = z = 3, x_2 = 11) \\
\xrightarrow{10,5}\ & ((L,L), x_1 = z = 4, x_2 = 12) \\
\longrightarrow\ & ((L,H), x_1 = 4, x_2 = z = 0) \\
\xrightarrow{32,20}\ & ((L,H), x_1 = 8, x_2 = z = 4) \\
\longrightarrow\ & ((H,H), x_1 = z = 0, x_2 = 4)
\end{aligned}
$$

Infinite stationary behaviours: An example

Two machines $M_1(D = 3, P = 3, G = 4, p = 5, g = 3)$, $M_2(D = 6, P = 3, G = 2, p = 5, g = 2)$ and an Operator $O(4)$.



(a) Schedule with mean-cost 1,455



(b) Schedule with mean-cost 1,478

From timed to discrete behaviours (1)

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

Lemma

Let Z be a bounded zone and f be a function

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

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

From timed to discrete behaviours (1)

- **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.

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

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

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

From timed to discrete behaviours (1)

- **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 $\mathit{inf}_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

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

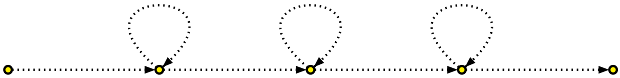
$$\text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi)$$

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

☞ optimal finite behaviours are not prefix-closed

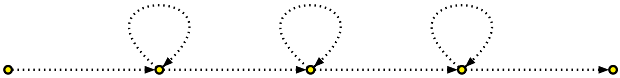
From timed to discrete behaviours (2)

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



From timed to discrete behaviours (2)

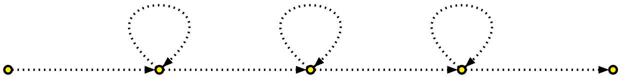
- **Infinite behaviours:** decompose each sufficiently long projection into cycles



The linear part will be negligible!

From timed to discrete behaviours (2)

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

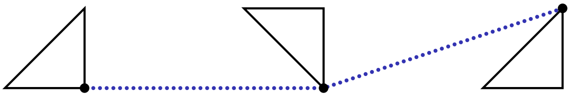


The linear part will be negligible!

➜ the optimal cycle of $\mathcal{A}_{\mathsf{cp}}$ is better than any infinite path of $\mathcal{A}$

From discrete to timed behaviours

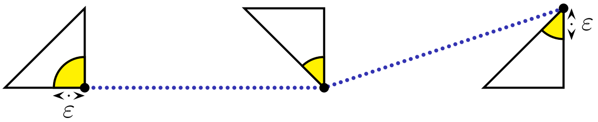
Approximation of abstract paths:



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

From discrete to timed behaviours

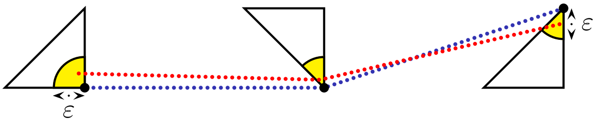
Approximation of abstract paths:



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

From discrete to timed behaviours

Approximation of abstract paths:

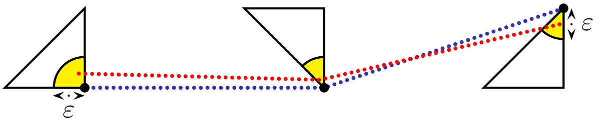


For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$, there exists a path π_ε of $\mathcal{A}$ s.t.

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

From discrete to timed behaviours

Approximation of abstract paths:



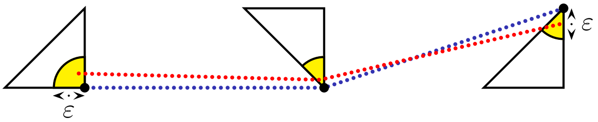
For any path Π of $\mathcal{A}_{\mathsf{cp}}$, for any $\varepsilon > 0$, there exists a path π_ε of $\mathcal{A}$ s.t.

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

➜ This is sufficient under the positive strongly diverging reward.

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.

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

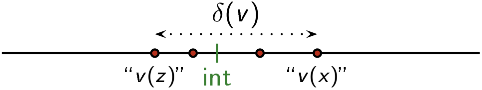
➜ This is sufficient under the positive strongly diverging reward.

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

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

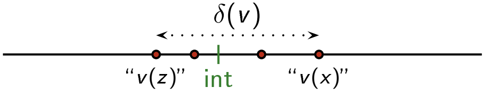
Approximation of abstract paths

Diameter of a valuation:

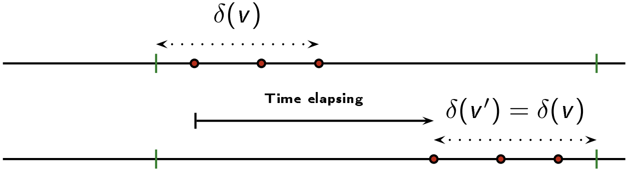


Approximation of abstract paths

Diameter of a valuation:

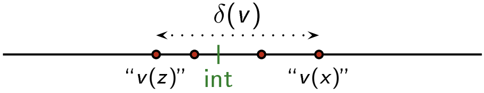


Computing successors:

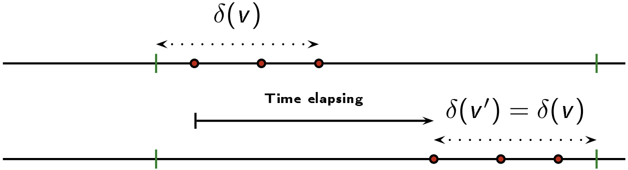


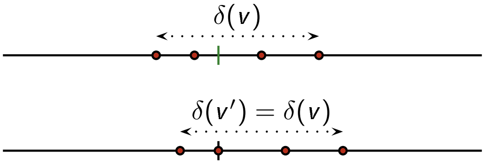
Approximation of abstract paths

Diameter of a valuation:

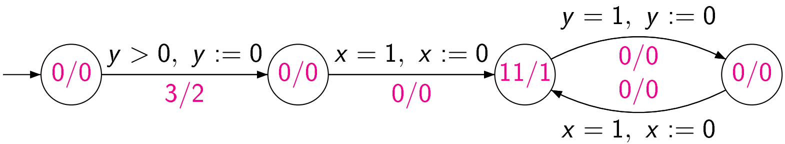


Computing successors:



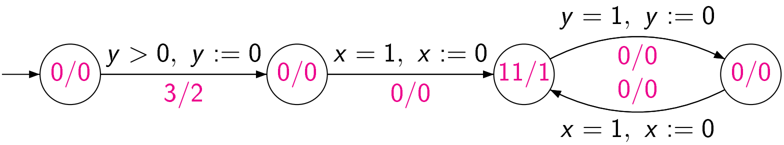


Hypothesis: strongly non-Zeno reward



$\pi_{d,n}$: path s.t. the first transition is taken at date d and the loop is taken n times.

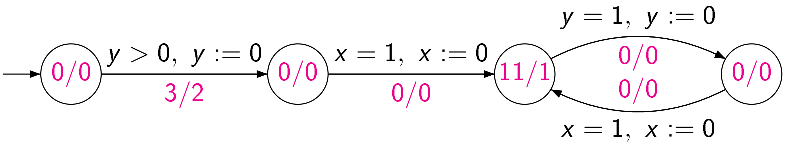
Hypothesis: strongly non-Zeno reward



$\pi_{d,n}$: path s.t. the first transition is taken at date d and the loop is taken n times.

$reward(\pi_{d,n}) = 2 + d.n$ and $cost(\pi_{d,n}) = 3 + 11d.n$

Hypothesis: strongly non-Zeno reward

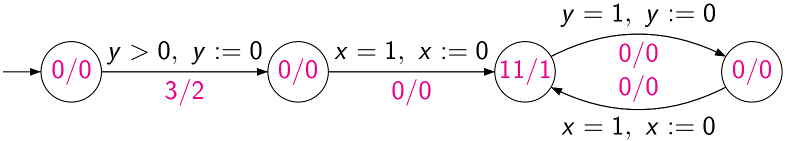


$\pi_{d,n}$: path s.t. the first transition is taken at date d and the loop is taken n times.

$reward(\pi_{d,n}) = 2 + d.n$ and $cost(\pi_{d,n}) = 3 + 11d.n$

For any real infinite path π_d, mean-cost$(\pi_d) = 11$ but mean-cost$(\pi_0) = \frac{3}{2}$.

Hypothesis: strongly non-Zeno reward



$\pi_{d,n}$: path s.t. the first transition is taken at date d and the loop is taken n times.

$$\mathit{reward}(\pi_{d,n}) = 2 + d.n \qquad \text{and} \qquad \mathit{cost}(\pi_{d,n}) = 3 + 11d.n$$

For any real infinite path π_d, mean-cost$(\pi_d) = 11$ but mean-cost$(\pi_0) = \frac{3}{2}$.

➜ this automaton is not strongly reward diverging

Model-checking WCTL

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

- With more than five clocks, model-checking WCTL is undecidable
  [Brihaye, Bruyère, Raskin – FORMATS+FTRTFT'04]
- With more than three clocks, model-checking WCTL is undecidable
  [Bouyer, Brihaye, Markey – IPL'06]

➜ Short explanation at the end of the talk

- With one clock, model-checking WCTL is decidable
  [Bouyer, Laroussinie, Larsen, Markey, Rasmussen – 2006]

Outline

Decidability of timed games

Theorem [Henzinger, Kopke 1999]

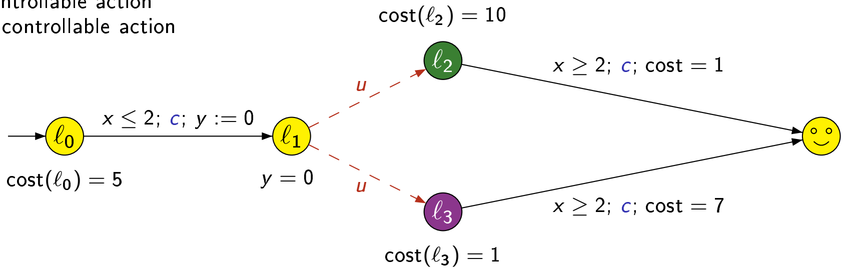
Safety and reachability control in timed automata are decidable and EXPTIME-complete.

(the attractor is computable...)

➜ classical regions are sufficient for solving such problems

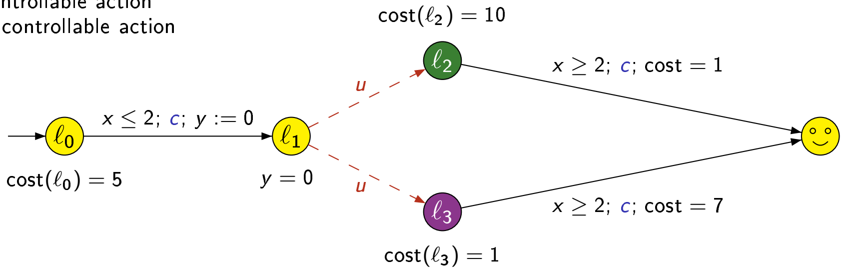
An example

c: controllable action
u: uncontrollable action



An example

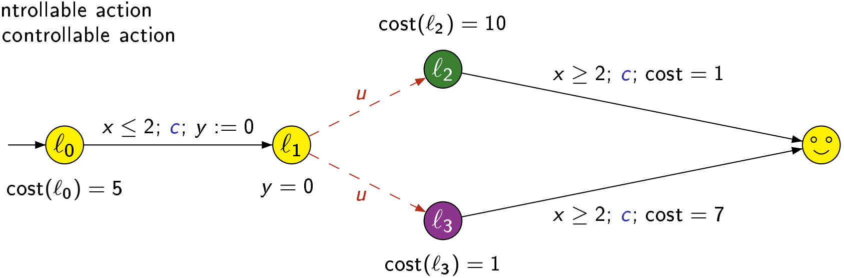
c: controllable action
u: uncontrollable action



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

An example

c: controllable action
u: uncontrollable action

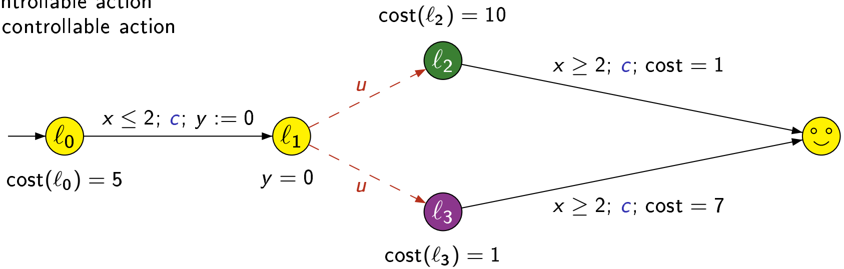


Question: what is the optimal cost we can ensure in state ℓ_0?

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

An example

c: controllable action
u: uncontrollable action

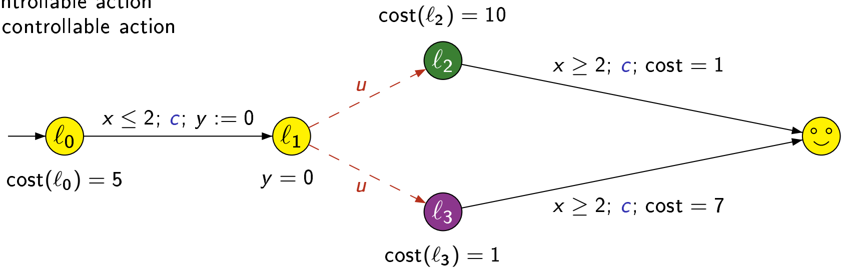


Question: what is the optimal cost we can ensure in state ℓ_0?

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

An example

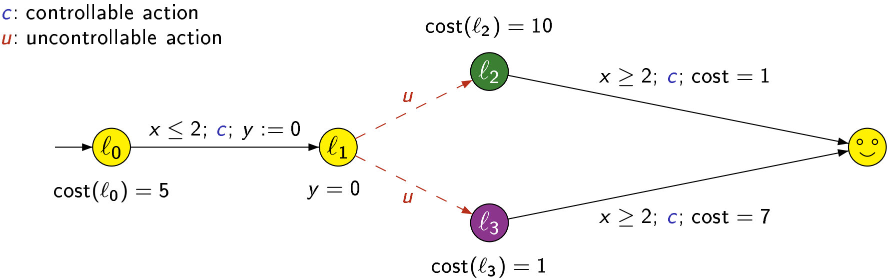
c: controllable action
u: uncontrollable action



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

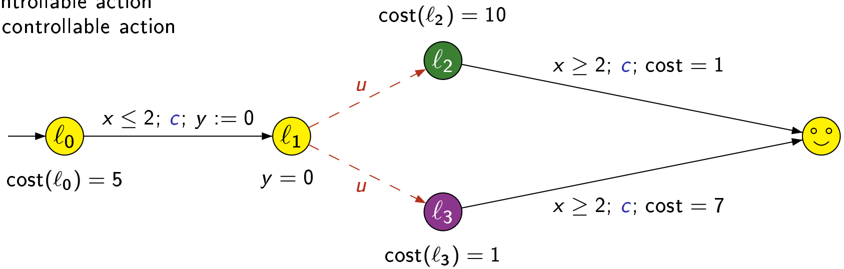


Question: what is the optimal cost we can ensure in state ℓ_0?

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

An example

c: controllable action
u: uncontrollable action



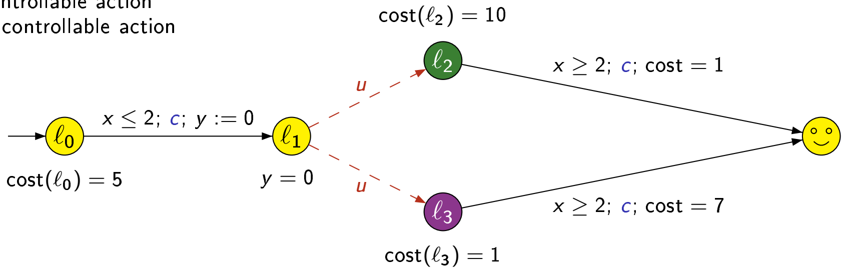
Question: what is the optimal cost we can ensure in state ℓ_0?

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

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

An example

c: controllable action
u: uncontrollable action



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

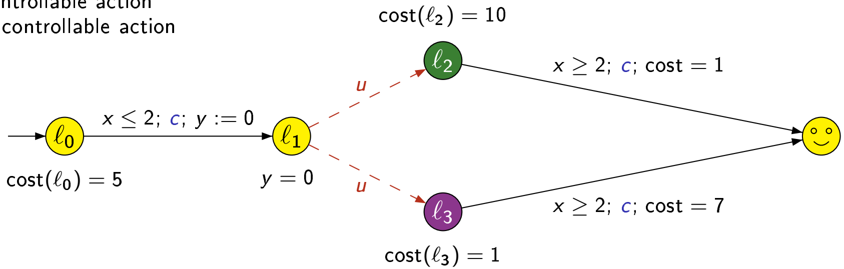
$$\inf_{0 \leq t \leq 2} \max\ (\ 5t + 10(2-t) + 1\ ,\ 5t + (2-t) + 7\ ) = 14 + \frac{1}{3}$$

➜ **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

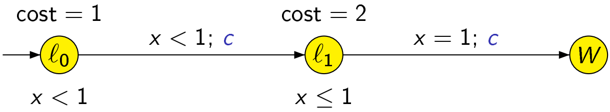
A hot topic!

- [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, Larsen, Markey, Rasmussen – Subm.'06]:
  - with one clock, optimal cost is computable

A hot topic!

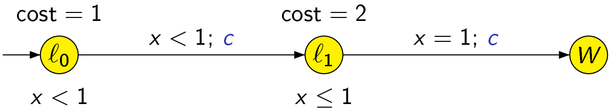
- [Asarin, Maler – HSCC'99]:
  - optimal time is computable in timed games
- [La Torre, Mukhopadhyay, Murano – TCS@02]:
  - case of acyclic games
- [Alur, Bernadsky, Madhusudan – ICALP'04]:
  - complexity of k-step games
  - under a strongly non-Zeno assumption, optimal cost is computable
- [Bouyer, 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, Larsen, Markey, Rasmussen – Subm.'06]:
  - with one clock, optimal cost is computable ➜ See Kim's talk

Do optimal strategies always exist?



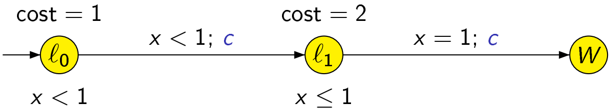
$$\begin{cases} f(\ell_0, x < 1) = \lambda \\ \\ f(\ell_1, x < 1) = \lambda \\ f(\ell_1, x = 1) = c \end{cases}$$

Do optimal strategies always exist?



$$\begin{cases} f(\ell_0, x < 1) = \lambda \\ \\ f(\ell_1, x < 1) = \lambda \\ f(\ell_1, x = 1) = c \end{cases} \quad \rightsquigarrow \quad \begin{cases} f_\varepsilon(\ell_0, x < 1 - \varepsilon) = \lambda \\ f_\varepsilon(\ell_0, 1 - \varepsilon \leq x < 1) = c \\ f_\varepsilon(\ell_1, x < 1) = \lambda \\ f_\varepsilon(\ell_1, x = 1) = c \end{cases}$$

Do optimal strategies always exist?

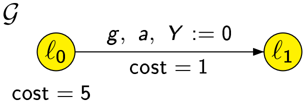


$$\begin{cases} f(\ell_0, x < 1) = \lambda \\ \\ f(\ell_1, x < 1) = \lambda \\ f(\ell_1, x = 1) = c \end{cases} \quad \rightsquigarrow \quad \begin{cases} f_\varepsilon(\ell_0, x < 1 - \varepsilon) = \lambda \\ f_\varepsilon(\ell_0, 1 - \varepsilon \leq x < 1) = c \\ f_\varepsilon(\ell_1, x < 1) = \lambda \\ f_\varepsilon(\ell_1, x = 1) = c \end{cases}$$

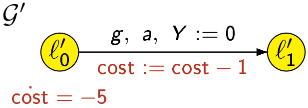
➜ **no optimal strategy exists**, but rather a family $(f_\varepsilon)_{\varepsilon > 0}$ of ε-approximating strategies ($\text{cost}(f_\varepsilon) = 1 + \varepsilon$)

An encoding (1)

Idea: tranform the cost into a decreasing linear hybrid variable



$\Longrightarrow$

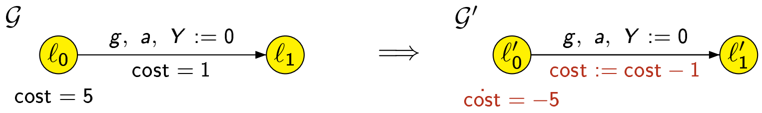


Winning: W

Winning: $W \wedge \mathsf{cost} \geq 0$

An encoding (1)

Idea: tranform the cost into a decreasing linear hybrid variable



Winning: W

Winning: $W \wedge \text{cost} \geq 0$

Theorem

For priced timed games (under some hypotheses),

$$\left.\begin{array}{r} \exists f \text{ winning strategy in } \mathcal{G} \\ \text{s.t. } \text{cost}(f,(\ell,v)) \leq \gamma \end{array}\right\} \iff (\ell, v, \text{cost} = \gamma) \text{ winning in } \mathcal{G}'$$

+ constructive proof

An encoding (2)

The set of winning states in $\mathcal{G}'$ is upward-closed for the cost, *i.e.* of the form

$$\bigcup_{i \in I} (P_i \wedge cost \succ_i k_i) \qquad \text{(with } \succ_i \text{ either } > \text{ or } \geq\text{)}$$

An encoding (2)

The set of winning states in $\mathcal{G}'$ is upward-closed for the cost, *i.e.* of the form

$$\bigcup_{i \in I} (P_i \wedge cost \succ_i k_i) \qquad \text{(with } \succ_i \text{ either } > \text{ or } \geq\text{)}$$

Corollary

For priced timed games (under some hypotheses),

- "reachable" optimal cost, or not (cost $\geq \gamma$ or cost $> \gamma$)
- existence of an optimal strategy decidable

+ constructive proof

An encoding (2)

The set of winning states in $\mathcal{G}'$ is upward-closed for the cost, *i.e.* of the form

$$\bigcup_{i \in I} (P_i \wedge cost \succ_i k_i) \qquad \text{(with } \succ_i \text{ either } > \text{ or } \geq\text{)}$$

Corollary

For priced timed games (under some hypotheses),

- "reachable" optimal cost, or not (cost $\geq \gamma$ or cost $> \gamma$)
- existence of an optimal strategy decidable

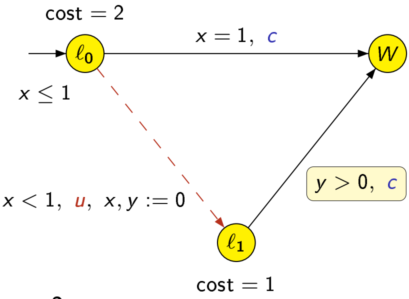
+ constructive proof

Nature of the strategy:

- state-based for the hybrid game, thus **cost-dependent** for the timed game
- cost-dependence is unavoidable in general!
- cost-independent strategies for syntactical restrictions of the games

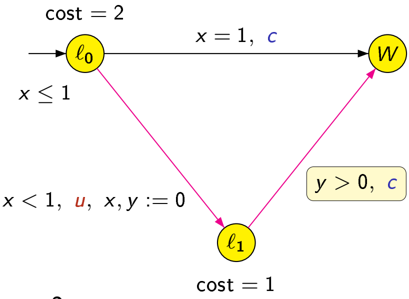
  c: large constraints, u: strict constraints

Memoryless strategies are not powerful enough



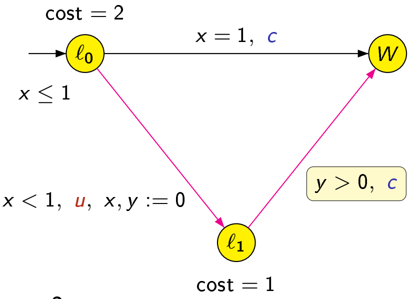
- **optimal cost:** 2
- **optimal strategy:**

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 $2.d + d'$.

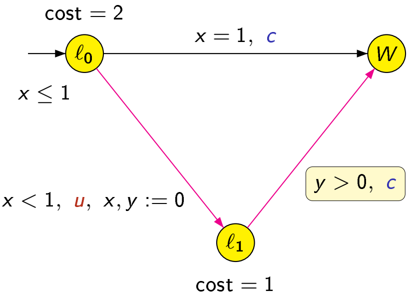
Memoryless strategies are not powerful enough



- **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 $2.d + d'$.

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

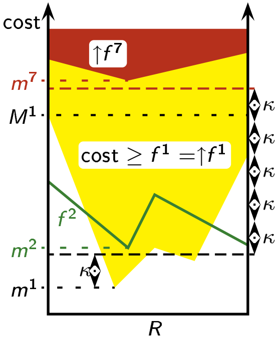
$$(\text{accumulated cost}) + d' \leq 2$$

Hypotheses for termination

- all clocks are bounded (not restrictive)
- the cost function is ***strictly non-Zeno***
  → This condition is restrictive, but is decidable

Hypotheses for termination

- all clocks are bounded (not restrictive)
- the cost function is ***strictly non-Zeno***
  - → This condition is restrictive, but is decidable



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:

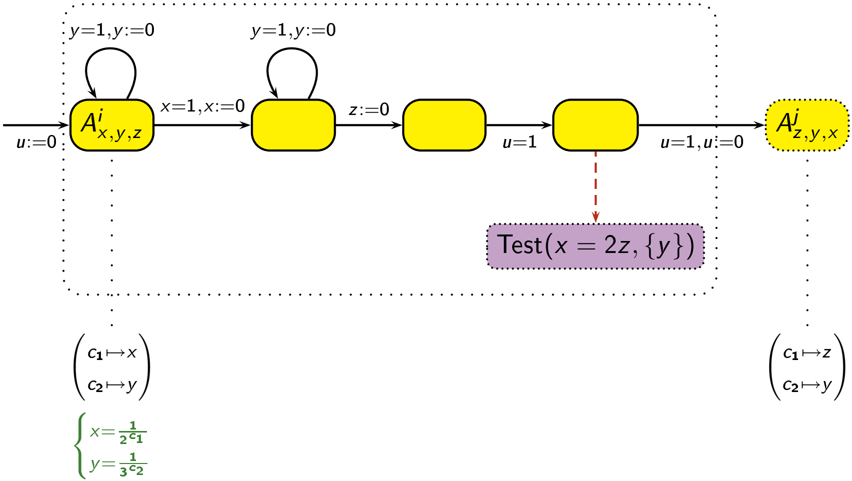
- counter c_1 is encoded by a clock x_1 s.t. $x_1 = \frac{1}{2^{c_1}}$
- counter c_2 is encoded by a clock x_2 s.t. $x_2 = \frac{1}{3^{c_2}}$
- x_1 and x_2 will be alternatively x, y or z

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

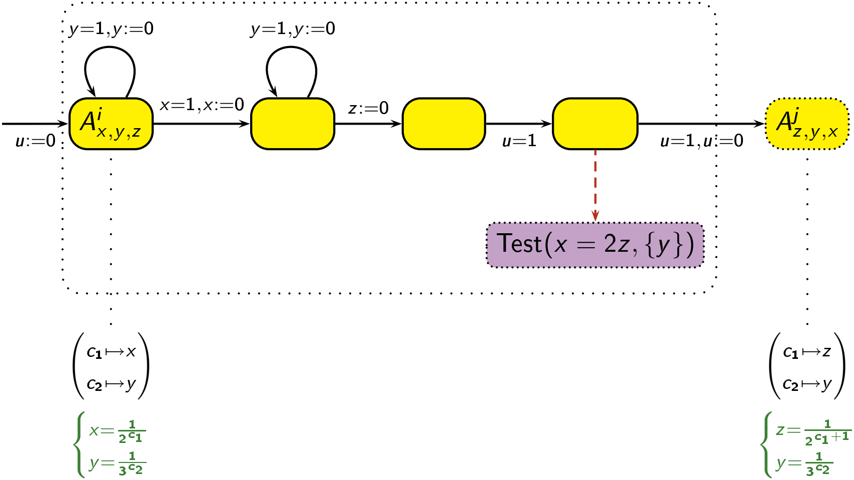
Simulation of an incrementation

Instruction i: c_1++; goto instruction j

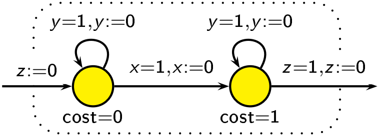


Simulation of an incrementation

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

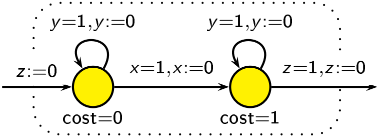


Adding x or $1 - x$ to the cost variable

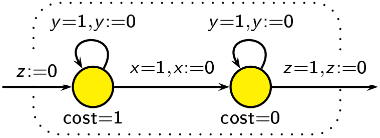


The cost is increased by x_0

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



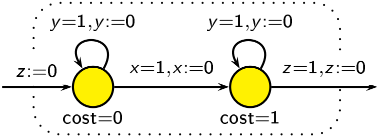
The cost is increased by x_0



The cost is increased by $1 - x_0$

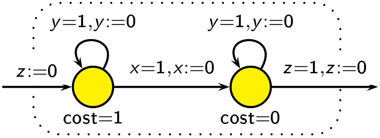
Adding x or $1 - x$ to the cost variable

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



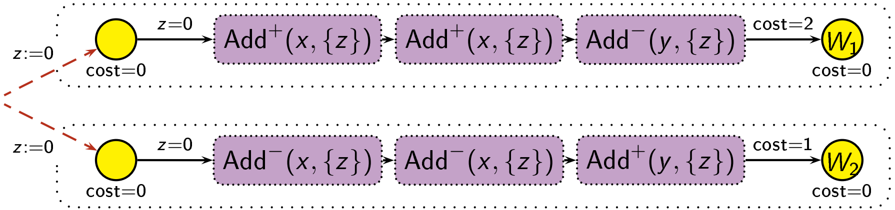
The cost is increased by x_0

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



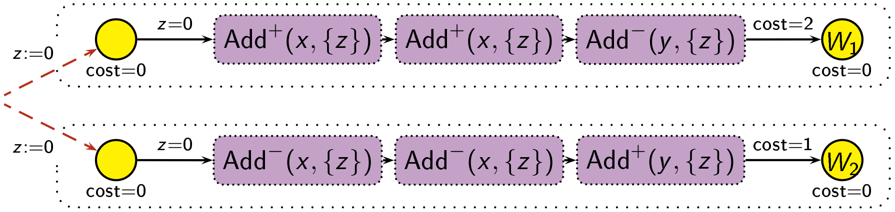
The cost is increased by $1 - x_0$

Checking $y = 2x$



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

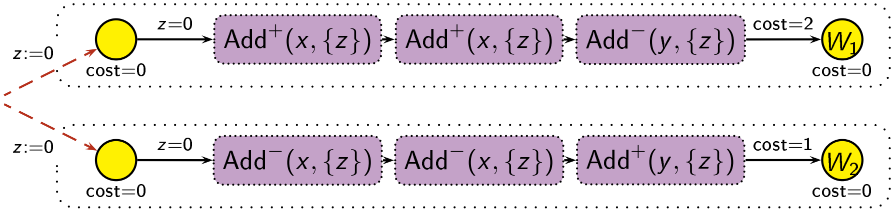
Checking $y = 2x$



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

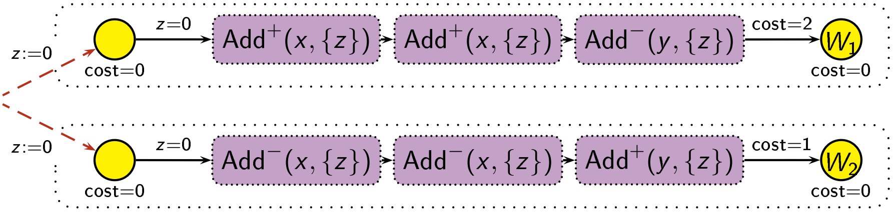
Checking $y = 2x$



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

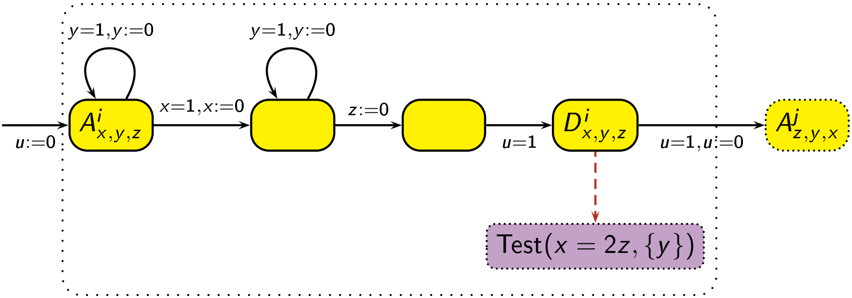
Checking $y = 2x$



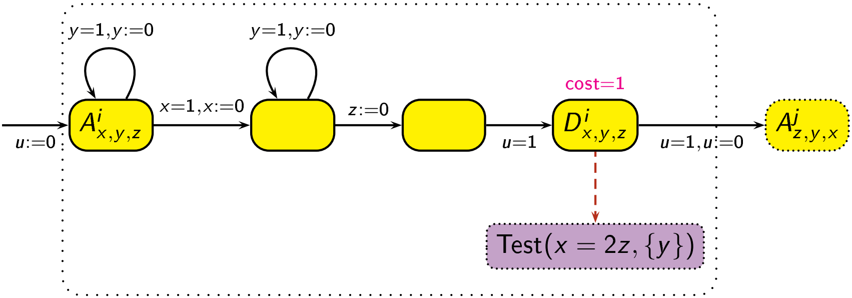
In W_1, cost $= 2x_0 + (1 - y_0) + 2$.
In W_2, cost $= 2(1 - x_0) + y_0 + 1$.

- 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?

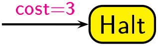


How to get rid of tick clock u?

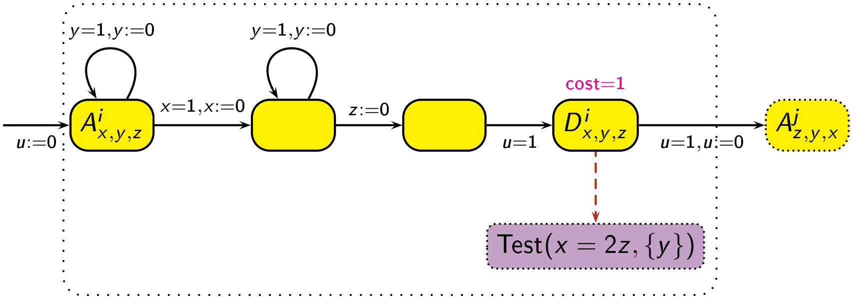


We will ensure that:

- no cost is accumulated in D-states

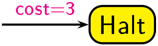


How to get rid of tick clock u?

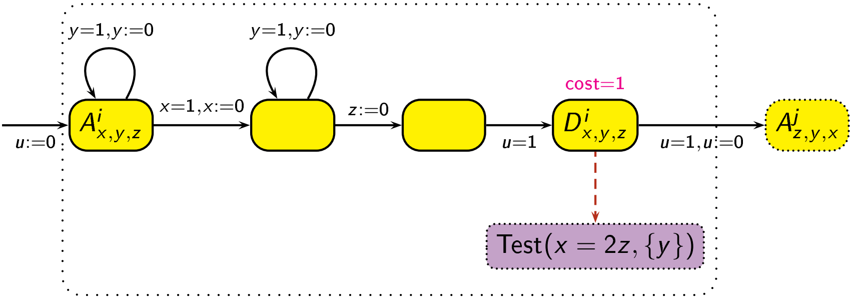


We will ensure that:

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

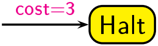


How to get rid of tick clock u?

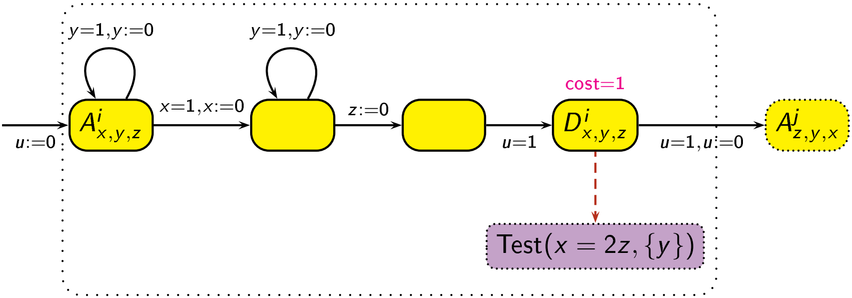


We will ensure that:

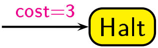
- no cost is accumulated in D-states
- the delay between the A-state and the D-state is 1 t.u.
  - the value of x in D is of the form $\frac{1}{2^n}$



How to get rid of tick clock u?

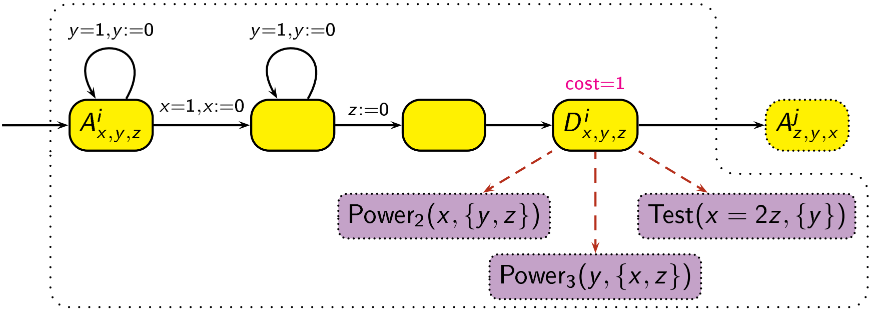


We will ensure that:

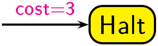


- no cost is accumulated in D-states
- the delay between the A-state and the D-state is 1 t.u.
  - the value of x in D is of the form $\frac{1}{2^n}$
  - the value of y in D is of the form $\frac{1}{3^m}$

How to get rid of tick clock u?

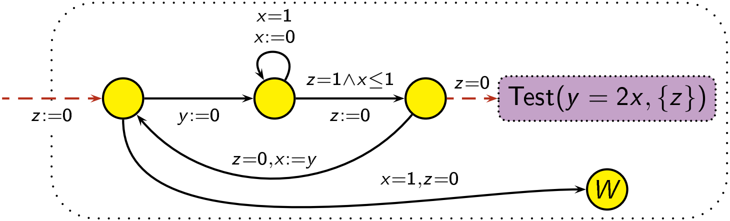


We will ensure that:



- 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 $\frac{1}{2^n}$



Extension to undecidability of WCTL

We build the same automaton $\mathcal{A}_{\mathcal{M}}$, and prove that:

the two-counter machine $\mathcal{M}$ halts iff $\mathcal{A}_{\mathcal{M}} \models \Phi$

where

$$\Phi \equiv \mathbf{E}\,(D \rightarrow \varphi)\,\mathbf{U}_{\leq 0}\mathsf{Halt}$$

$$\text{with } \varphi \equiv \bigwedge_{i=1,2,3} \mathbf{E}\,(D\,\mathbf{U}_{\leq 0}\varphi_i)$$

$\varphi_1 \equiv S \wedge \mathbf{E}\,\mathbf{F}_{\leq 1} T \wedge \mathbf{E}\,\mathbf{F}_{\geq 1} T$ evaluated in $\mathsf{Test}(x = 2z, \{y\})$

$\varphi_2 \equiv P_2 \wedge \mathbf{E}\,((Q_2 \rightarrow \mathbf{E}\,(Q_2\,\mathbf{U}\,\varphi_1))\,\mathbf{U}\,R_2)$ evaluated in $\mathsf{Power}_2(x, \{y, z\})$

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
- o-minimal optimal timed games
- ...