Synthesis of Timed Systems

Patricia Bouyer

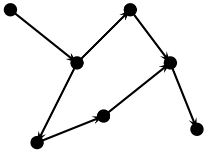
LSV – CNRS & ENS de Cachan

Spring School GAMES

Model-checking

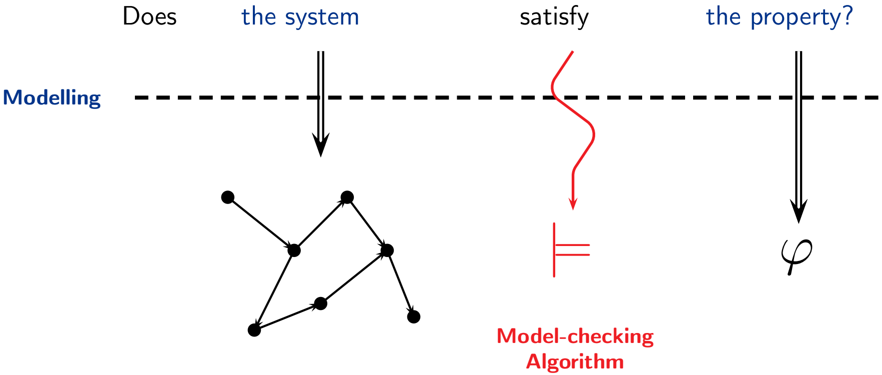
Does the system satisfy the property?

Modelling



φ

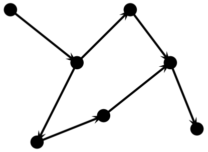
Model-checking



Controller synthesis

Can we force the system satisfying the property?

Modelling

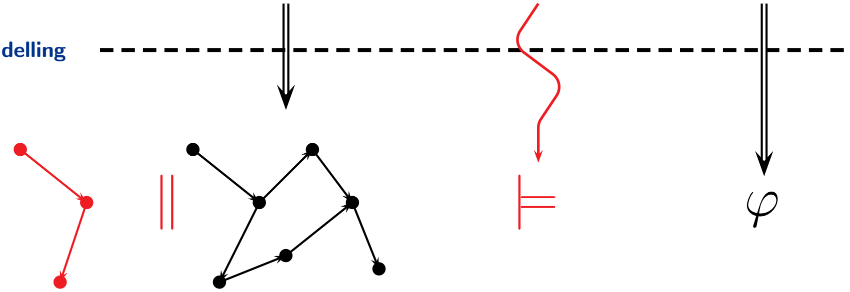


φ

Controller synthesis

Can we force the system satisfying the property?

Modelling



Controller synthesis

Time!

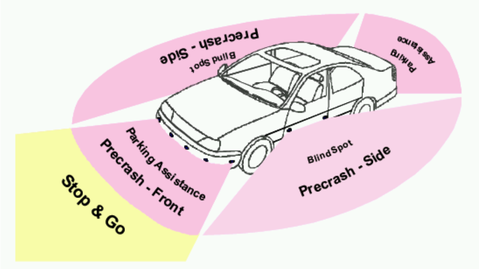
Context: verification and control of embedded critical systems

Time:

- naturally appears in real systems
- appears in properties (for ex. bounded response time)
- systems continuously interact with environment

➜ We need to take care of timing aspects

An example, the car periphery supervision

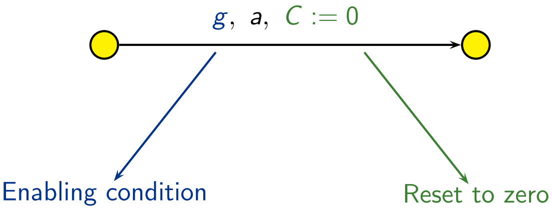


- Embedded system
- Hostile environment
- Sensors
  - distances
  - speeds

Outline

Timed automata [Alur & Dill 90's]

- A finite control structure + variables (clocks)
- A transition is of the form:



- An enabling condition (or guard) is:

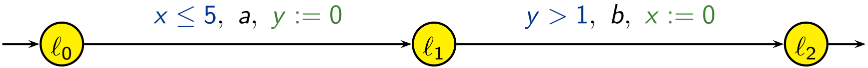
$$g \quad ::= \quad x \sim c \quad | \quad g \wedge g$$

where $\sim \in \{<, \leq, =, \geq, >\}$

- An invariant in each location

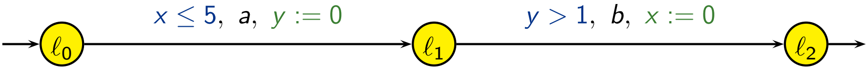
Timed automata (example)

x, y : clocks



Timed automata (example)

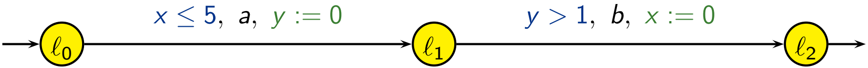
x, y : clocks



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

Timed automata (example)

x, y : clocks

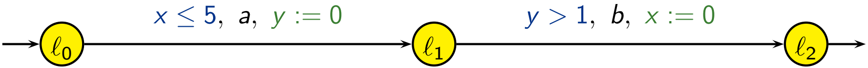


$$\begin{array}{ccccccccc} & \ell_0 & \xrightarrow{\delta(4.1)} & \ell_0 & \xrightarrow{a} & \ell_1 & \xrightarrow{\delta(1.4)} & \ell_1 & \xrightarrow{b} & \ell_2 \\ x & 0 & & 4.1 & & 4.1 & & 5.5 & & 0 \\ y & 0 & & 4.1 & & 0 & & 1.4 & & 1.4 \end{array}$$

(clock) valuation

Timed automata (example)

x, y : clocks



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

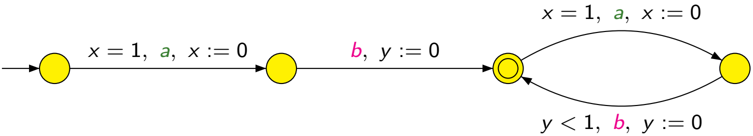
(clock) valuation

➜ timed word $(a, 4.1)(b, 5.5)$

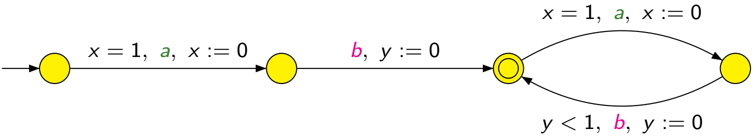
Timed automata semantics

- $\mathcal{A} = (\Sigma, L, X, \longrightarrow)$ is a TA
- **Configurations:** $(\ell, v) \in L \times T^X$ where T is the time domain
- **Timed Transition System:**
  - action transition: $(\ell, v) \xrightarrow{a} (\ell', v')$ if $\exists \ell \xrightarrow{g,a,r} \ell' \in \mathcal{A}$ s.t.
    $$\begin{cases} v \models g \\ v' = v[r \leftarrow 0] \end{cases}$$
  - delay transition: $(\ell, v) \xrightarrow{\delta(d)} (\ell, v+d)$ if $d \in T$

Timed languages

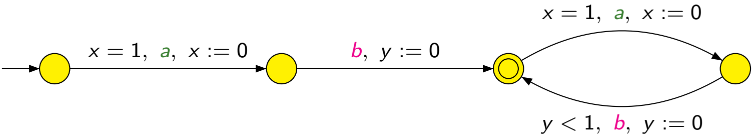


Timed languages



- **Dense-time:**
  $L_{dense} = \{((ab)^\omega, \tau) \mid \forall i, \ \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1}\}$

Timed languages



- **Dense-time:**
  $L_{dense} = \{((ab)^\omega, \tau) \mid \forall i,\ \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1}\}$

- **Discrete-time:** $L_{discrete} = \emptyset$

Verification

Emptiness problem: is the language accepted by a timed automaton empty?

- reachability properties (final states)
- basic liveness properties (Büchi (or other) conditions)

Verification

Emptiness problem: is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  ➜ classical methods can not be applied

Verification

Emptiness problem: is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  ➜ classical methods can not be applied

- **Positive key point:** variables (clocks) have the same speed

Verification

Emptiness problem: is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  ➜ classical methods can not be applied

- **Positive key point:** variables (clocks) have the same speed

Theorem [Alur & Dill 1990's]

The emptiness problem for timed automata is decidable.
It is PSPACE-complete.

Verification

Emptiness problem: is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  ➜ classical methods can not be applied

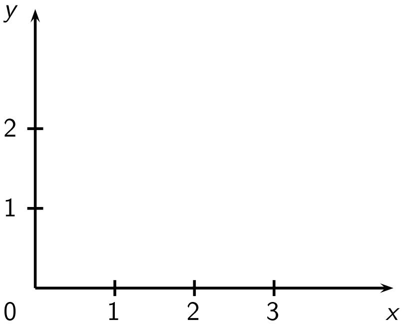
- **Positive key point:** variables (clocks) have the same speed

Theorem [Alur & Dill 1990's]

The emptiness problem for timed automata is decidable.
It is PSPACE-complete.

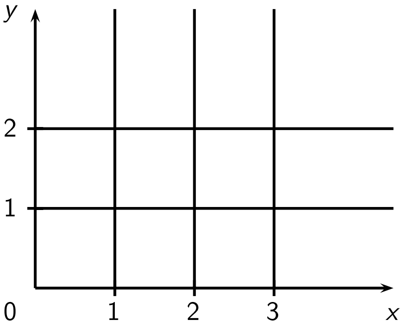
Method: construct a finite abstraction

The region abstraction



Equivalence of finite index

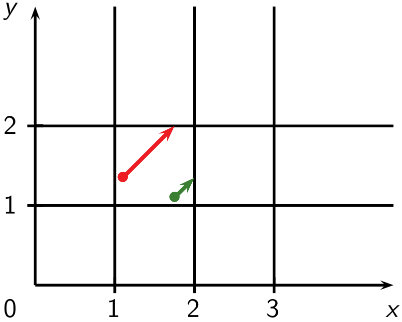
The region abstraction



Equivalence of finite index

- “compatibility” between regions and constraints

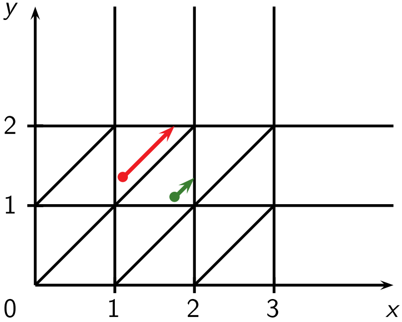
The region abstraction



Equivalence of finite index

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

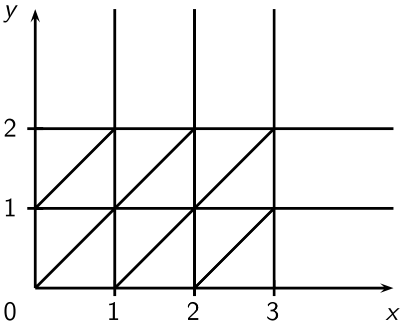
The region abstraction



Equivalence of finite index

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

The region abstraction

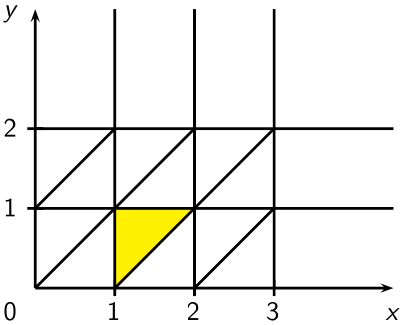


Equivalence of finite index

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

➜ a **bisimulation** property

The region abstraction



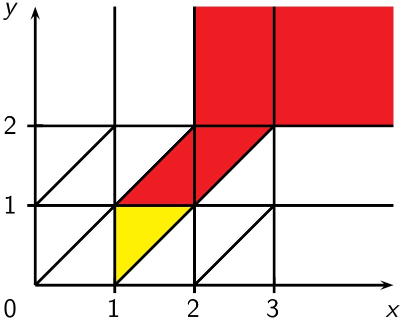
Equivalence of finite index

region defined by
$I_x =]1;2[,\ I_y =]0;1[$
$\{x\} < \{y\}$

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

➜ a **bisimulation** property

The region abstraction



Equivalence of finite index

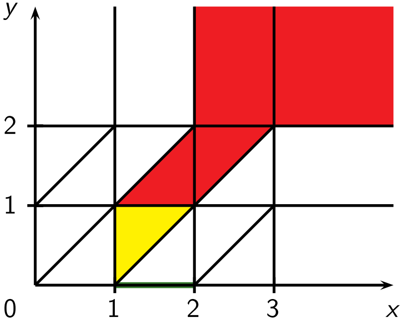
region defined by
$I_x =]1;2[$, $I_y =]0;1[$
$\{x\} < \{y\}$

delay successors

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

➜ a **bisimulation** property

The region abstraction



Equivalence of finite index

region defined by

$I_x =]1;2[$, $I_y =]0;1[$

$\{x\} < \{y\}$

delay successors

successor by reset

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

➜ a **bisimulation** property

The region automaton

timed automaton $\bigotimes$ region abstraction

$\ell \xrightarrow{g,a,C:=0} \ell'$ is transformed into:

$(\ell, R) \xrightarrow{a} (\ell', R')$ if there exists $R'' \in \mathrm{Succ}_t^*(R)$ s.t.

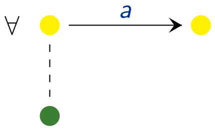
- $R'' \subseteq g$
- $[C \leftarrow 0]R'' \subseteq R'$

➜ **time-abstract bisimulation**

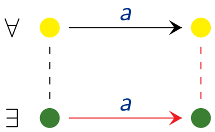
$\mathcal{L}$**(reg. aut.) = UNTIME(**$\mathcal{L}$**(timed aut.))**

where $\mathrm{UNTIME}((a_1, t_1)(a_2, t_2)\dots) = a_1 a_2 \dots$

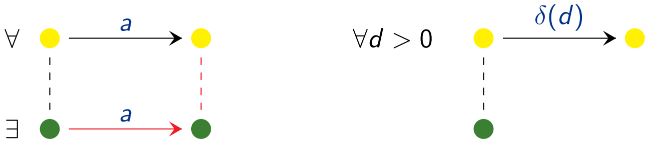
Time-abstract bisimulation



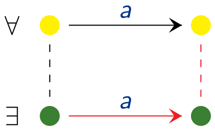
Time-abstract bisimulation

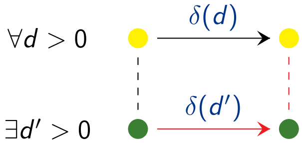


Time-abstract bisimulation

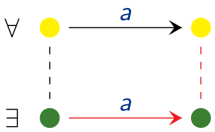


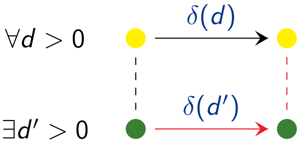
Time-abstract bisimulation





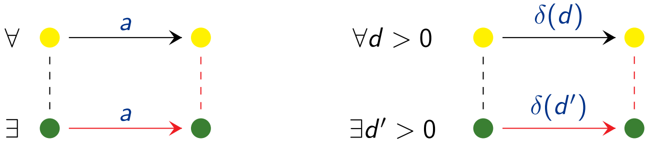
Time-abstract bisimulation





$$(\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \xrightarrow{a_2, t_2} (\ell_2, v_2) \xrightarrow{a_3, t_3} \dots$$

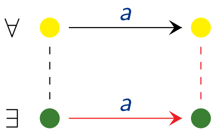
Time-abstract bisimulation

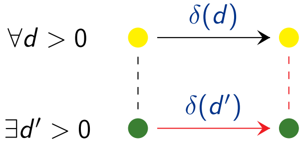


$$\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \dots \\
\downarrow & & \downarrow & & \downarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3} & \dots
\end{array}$$

with $v_i \in R_i$ for all i.

Time-abstract bisimulation

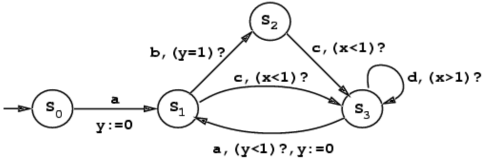


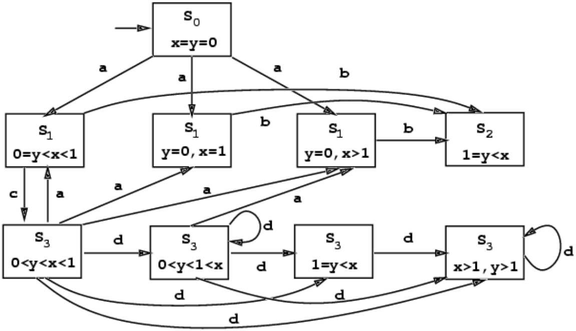


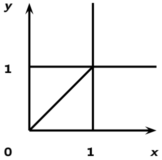
$$\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \dots \\
\updownarrow & & \updownarrow & & \updownarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3} & \dots
\end{array}$$

with $v_i \in R_i$ for all i.

An example [Alur & Dill 1990's]







Consequence of region automata construction

Region automata: correct finite abstraction for checking reachability/Büchi-like properties

Consequence of region automata construction

Region automata: correct finite abstraction for checking reachability/Büchi-like properties

However, everything can not be reduced to finite automata...

A model not far from undecidability

Properties

- Universality is undecidable [Alur & Dill 90's]
- Inclusion is undecidable [Alur & Dill 90's]
- Determinizability is undecidable [Tripakis 2003]
- Complementability is undecidable [Tripakis 2003]
- ...

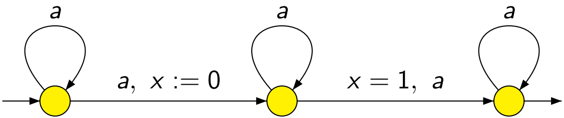
A model not far from undecidability

Properties

- Universality is undecidable [Alur & Dill 90's]
- Inclusion is undecidable [Alur & Dill 90's]
- Determinizability is undecidable [Tripakis 2003]
- Complementability is undecidable [Tripakis 2003]
- ...

Example

A non-determinizable/non-complementable timed automaton:



The two-counter machine

Definition

A two-counter machine is a finite set of instructions over two counters (x and y):

- Incrementation:
  (p): $x := x + 1$; goto (q)
- Decrementation:
  (p): if $x > 0$ then $x := x - 1$; goto (q) else goto (r)

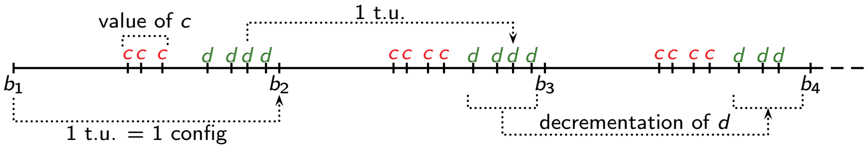
Theorem [Minsky 67]

The halting problem and the recurring problem for two-counter machines are undecidable.

Undecidability of universality

Theorem [Alur & Dill 90's]

Universality of timed automata is undecidable.

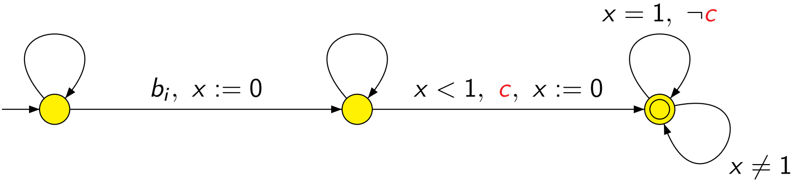


- one configuration is encoded in one time unit
- number of c's: value of counter c
- number of d's: value of counter d
- one time unit between two corresponding c's (resp. d's)

➜ We encode "non-behaviours" of a two-counter machine

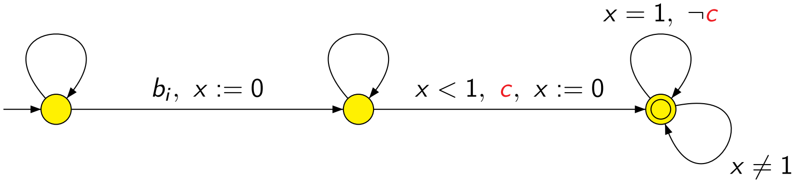
Example

Module to check that if instruction i does not decrease counter c, then all actions c appearing less than 1 t.u. after b_i has to be followed by an other c 1 t.u. later.



Example

Module to check that if instruction i does not decrease counter c, then all actions c appearing less than 1 t.u. after b_i has to be followed by an other c 1 t.u. later.



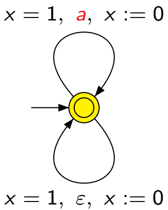
The union of all small modules is not universal
iff
The two-counter machine has a recurring computation

Power of ε-transitions

[Bérard, Diekert, Gastin, Petit 1998]

Proposition

- ε-transitions can not be removed in timed automata.
- Timed automata with ε-transitions are strictly more expressive than timed automata without ε-transitions.



On *zeno* behaviours

Definition

An infinite timed behaviour $(a_1, t_1)(a_2, t_2)\ldots$ is *zeno* whenever the infinite timed sequence $(t_i)_{i \geq 1}$ is bounded.

➜ infinitely many discrete events within a bounded amount of time

Example

$$\left(a, \frac{1}{2}\right)\left(a, \frac{3}{4}\right)\left(a, \frac{7}{8}\right)\ldots\left(a, 1 - \frac{1}{2^n}\right)\ldots$$

On *zeno* behaviours

Definition

An infinite timed behaviour $(a_1, t_1)(a_2, t_2) \dots$ is *zeno* whenever the infinite timed sequence $(t_i)_{i \geq 1}$ is bounded.

➜ infinitely many discrete events within a bounded amount of time

Example

$$\left(a, \frac{1}{2}\right) \left(a, \frac{3}{4}\right) \left(a, \frac{7}{8}\right) \dots \left(a, 1 - \frac{1}{2^n}\right) \dots$$

Proposition [Alur & Dill 90's]

It is decidable whether a timed automaton only generates non-*zeno* behaviours, and it is decidable whether there is a non-*zeno* behaviour which satisfies a Büchi condition.

(Using the region automaton)

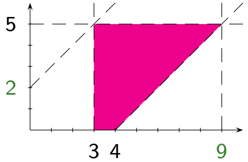
A symbolic representation for timed systems

A **zone** is a set of valuations defined by a constraint of the form

$$\varphi \ ::= \ x \sim c \ \mid \ x - y \sim c \ \mid \ \varphi \wedge \varphi$$

Example

The constraint $(x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$ defines the zone:



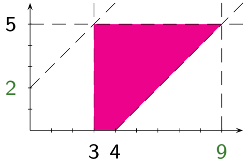
A symbolic representation for timed systems

A **zone** is a set of valuations defined by a constraint of the form

$$\varphi \ ::= \ x \sim c \ \mid \ x - y \sim c \ \mid \ \varphi \wedge \varphi$$

Example

The constraint $(x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$ defines the zone:



- $\{v + t \mid v \models \varphi \text{ and } t \geq 0\}$ is a zone **[Future]**
- $\{[Y \leftarrow 0]v \mid v \models \varphi\}$ is a zone **[Reset]**
- $\{v - t \mid v \models \varphi \text{ and } t \geq 0\}$ is a zone **[Past]**
- ...

Outline

☺ in the last ten years, a prolific literature

☺ in the last ten years, a prolific literature

☹ one paper = one definition of timed games

- symmetrical games or not
- take care of *zeno* (and *zeno*-like) behaviours or not
- turn-based games or not
- ...

See bibliography for references...

Timed games with surprise

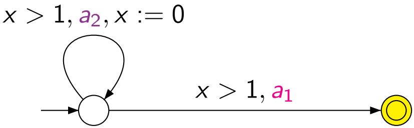
The most achieved model of timed games
[de Alfaro, Faella, Henzinger, Majumdar, Stoelinga 2003]

- two players
- each player i chooses a pair (δ_i, a_i) **Semantics with surprise**
  each player i chooses either a delay δ_i, or an action a_i
  Semantics without surprise
- the next move is determined by the following rules:
  - if $\delta_i < \delta_j$, wait δ_i and do a_i
  - if $\delta_i = \delta_j$, wait δ_i and do either a_i or a_j (non-deterministic choice)
- classical ω-regular winning condition + time divergence condition or non-responsible for time divergence

Proposition

Reachability and safety timed games are not determined.

Example (Aim: enforce (or avoid) yellow state)



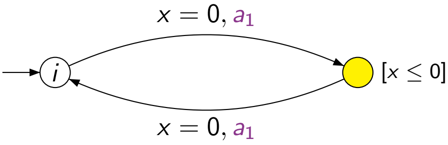
- if player 2 proposes move (Δ_2, a_2) with $\Delta_2 > 1$, then player 1 can propose move $(1 + \frac{\Delta_2 - 1}{2}, a_1)$
- if player 2 proposes move (Δ_2, a_2) with $\Delta_2 \leq 1$, then player 1 can propose move $(1, \perp)$

If player 2 never proposes $\Delta_2 > 1$, then it has to be blamed for time convergence.

Proposition

- Memoryless strategies suffice to win safety games.
- Memoryless strategies do not suffice to win reachability games.

Example (Aim: enforce yellow state)

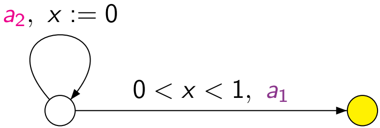


In state $(i, x = 0)$, the strategy has to remember if the other state has already been visited or not.

Proposition

Surprise is sometimes necessary to win...

Example (Aim: enforce yellow state)



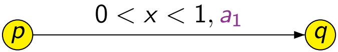
Player 1 has a winning strategy:

- after n rounds, his strategy is $(\frac{1}{2^{n+1}}, a_1)$
- if yellow state is not reached, then time converges and this is due to player 2
- player 1 wins!

Proposition

"Real" strategies based on regions are not sufficient...

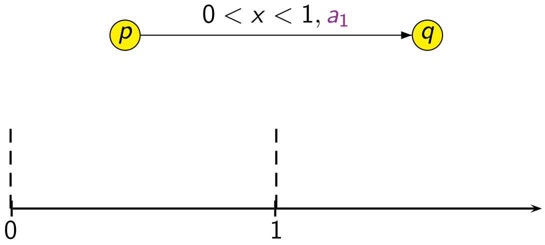
Example



Proposition

"Real" strategies based on regions are not sufficient...

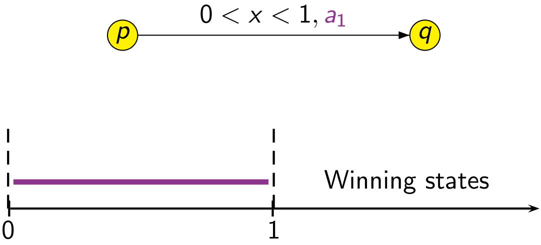
Example



Proposition

"Real" strategies based on regions are not sufficient...

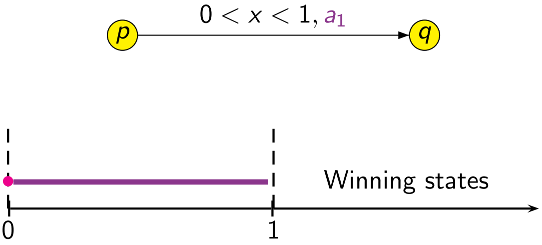
Example



Proposition

"Real" strategies based on regions are not sufficient...

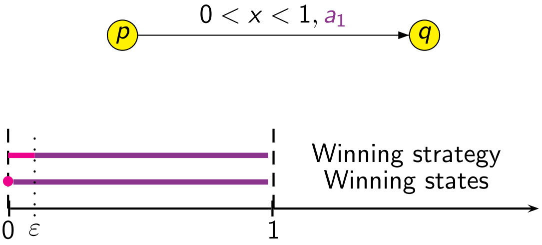
Example



Proposition

"Real" strategies based on regions are not sufficient...

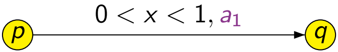
Example

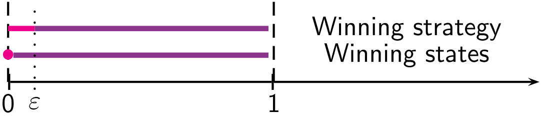


Proposition

"Real" strategies based on regions are not sufficient...

Example





Time elapsing $\neq$ discrete action!

Decidability results

[de Alfaro, Faella, Henzinger, Majumdar, Stoelinga 2003]

Theorem

Timed games with a parity winning condition can be solved effectively. Moreover *persistent* strategies are sufficient, and history based on visited regions is sufficient.

- time divergence and blameless conditions are expressed as an untimed parity condition
- for such a parity condition, winning only depends on the region

Outline

Framework of these games

Environment against controller
(Non-symmetrical game)

- some actions are controllable Σ_c
- some actions are uncontrollable Σ_u
- player "environment" can:
  - interrupt time elapsing,
  - enforce *zeno* behaviours
  - ...
- a **plant** $\mathcal{P}$ is a deterministic timed automaton over alphabet $\Sigma_c \cup \Sigma_u$ (it represents both real system and environment)

Strategies and controllers

- A **strategy** is a partial function

$$f : Runs(\mathcal{P}) \longrightarrow \Sigma_c \cup \{\lambda\} \qquad\qquad \lambda : \text{ time elapsing}$$

Strategies and controllers

- A **strategy** is a partial function

$$f : Runs(\mathcal{P}) \longrightarrow \Sigma_c \cup \{\lambda\} \qquad\qquad \lambda : \text{ time elapsing}$$

  - needs to satisfy some *continuity* property:

$$f(\rho) = \lambda \implies \exists t > 0,\ \forall 0 \leq t' < t,\ f(\rho \xrightarrow{\delta(t')}) = \lambda$$

Strategies and controllers

- A **strategy** is a partial function

$$f : Runs(\mathcal{P}) \longrightarrow \Sigma_c \cup \{\lambda\} \qquad\qquad \lambda: \text{ time elapsing}$$

  - needs to satisfy some *continuity* property:

$$f(\rho) = \lambda \implies \exists t > 0,\ \forall 0 \leq t' < t,\ f(\rho \xrightarrow{\delta(t')}) = \lambda$$

- A **controller** is a deterministic timed automaton over $\Sigma_c \cup \Sigma_u$ which will run in parallel with $\mathcal{P}$

Strategies and controllers

- A **strategy** is a partial function

$$f : Runs(\mathcal{P}) \longrightarrow \Sigma_c \cup \{\lambda\} \qquad\qquad \lambda : \text{ time elapsing}$$

  - needs to satisfy some *continuity* property:

$$f(\rho) = \lambda \implies \exists t > 0,\ \forall 0 \leq t' < t,\ f(\rho \xrightarrow{\delta(t')}) = \lambda$$

- A **controller** is a deterministic timed automaton over $\Sigma_c \cup \Sigma_u$ which will run in parallel with $\mathcal{P}$

How powerful can it be?

Strategies and controllers

- A **strategy** is a partial function

$$f : Runs(\mathcal{P}) \longrightarrow \Sigma_c \cup \{\lambda\} \qquad\qquad \lambda : \text{ time elapsing}$$

  - needs to satisfy some *continuity* property:

$$f(\rho) = \lambda \implies \exists t > 0,\ \forall 0 \leq t' < t,\ f(\rho \xrightarrow{\delta(t')}) = \lambda$$

- A **controller** is a deterministic timed automaton over $\Sigma_c \cup \Sigma_u$ which will run in parallel with $\mathcal{P}$

How powerful can it be? **Not too much!**

Strategies and controllers

- A **strategy** is a partial function

$$f : Runs(\mathcal{P}) \longrightarrow \Sigma_c \cup \{\lambda\} \qquad \lambda : \text{ time elapsing}$$

  - needs to satisfy some *continuity* property:

$$f(\rho) = \lambda \implies \exists t > 0,\ \forall 0 \leq t' < t,\ f(\rho \xrightarrow{\delta(t')}) = \lambda$$

- A **controller** is a deterministic timed automaton over $\Sigma_c \cup \Sigma_u$ which will run in parallel with $\mathcal{P}$

How powerful can it be? **Not too much!**

  - needs to be *non-restricting* for uncontrollable actions

Strategies and controllers

- A **strategy** is a partial function

$$f : Runs(\mathcal{P}) \longrightarrow \Sigma_c \cup \{\lambda\} \qquad\qquad \lambda : \text{ time elapsing}$$

  - needs to satisfy some *continuity* property:

$$f(\rho) = \lambda \implies \exists t > 0,\ \forall 0 \leq t' < t,\ f(\rho \xrightarrow{\delta(t')}) = \lambda$$

- A **controller** is a deterministic timed automaton over $\Sigma_c \cup \Sigma_u$ which will run in parallel with $\mathcal{P}$

How powerful can it be? **Not too much!**

  - needs to be *non-restricting* for uncontrollable actions
  - needs to be *non-blocking*: if there is no deadlock in the original plant, there will be no deadlock in the controlled system

Winning a timed game

- the controlled system (or the outcomes of the game) is $\mathcal{P} \parallel \mathcal{C}$

Winning a timed game

- the controlled system (or the outcomes of the game) is $\mathcal{P} \parallel \mathcal{C}$
- specifications (or winning conditions)
  - **Internal specifications:** conditions on the states of the plant
    - **safety:** the controlled system avoids bad states
    - **reachability:** $\mathcal{C}$ enforces a good state
    - ...

Winning a timed game

- the controlled system (or the outcomes of the game) is $\mathcal{P} \parallel \mathcal{C}$
- specifications (or winning conditions)
  - **Internal specifications:** conditions on the states of the plant
    - **safety:** the controlled system avoids bad states
    - **reachability:** $\mathcal{C}$ enforces a good state
    - ...
  - **External specifications:** given by a timed automaton $\mathcal{S}$
    - representing desired behaviours

$$L(\mathcal{P} \parallel \mathcal{C}) \subseteq L(\mathcal{S})$$

    - representing undesired behaviours

$$L(\mathcal{P} \parallel \mathcal{C}) \cap L(\mathcal{S}) = \emptyset$$

Winning a timed game

- the controlled system (or the outcomes of the game) is $\mathcal{P} \parallel \mathcal{C}$
- specifications (or winning conditions)
  - **Internal specifications:** conditions on the states of the plant
    - **safety:** the controlled system avoids bad states
    - **reachability:** $\mathcal{C}$ enforces a good state
    - ...
  - **External specifications:** given by a timed automaton $\mathcal{S}$
    - representing desired behaviours

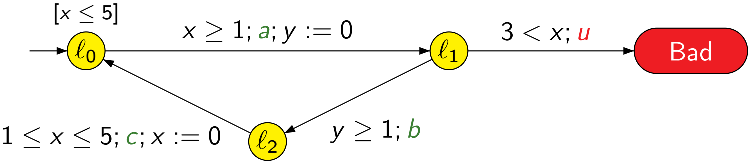
$$L(\mathcal{P} \parallel \mathcal{C}) \subseteq L(\mathcal{S})$$

    - representing undesired behaviours

$$L(\mathcal{P} \parallel \mathcal{C}) \cap L(\mathcal{S}) = \emptyset$$

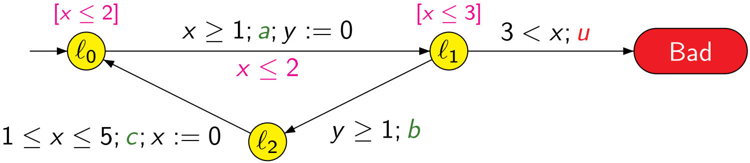
external det. specification $\equiv$ internal specification

An example



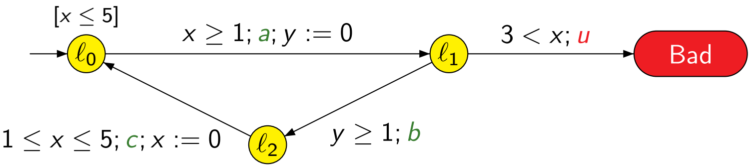
Aim: control the system in such a way that "Bad" state is avoided.

An example



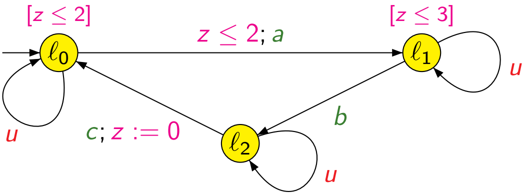
Aim: control the system in such a way that "Bad" state is avoided.

An example

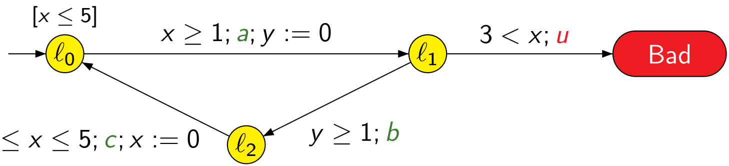


Aim: control the system in such a way that "Bad" state is avoided.

A controller:

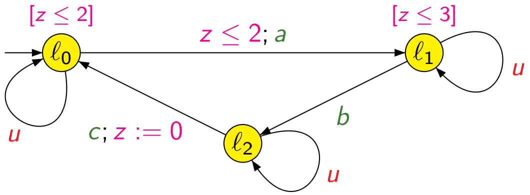


An example



Aim: control the system in such a way that "Bad" state is avoided.

A controller:



A winning strategy: $\begin{cases} f(\ell_0, x < 1) = \lambda \\ f(\ell_0, x = 1) = a \end{cases}$ $\begin{cases} f(\ell_1, x < 2) = \lambda \\ f(\ell_1, x = 2) = b \\ f(\ell_2, x = 2) = c \end{cases}$

Computing winning states

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

$$\mathrm{cPred}(X) = \bigcup_{c \in \Sigma_c} \mathrm{Pred}^c(X) \qquad \mathrm{uPred}(X) = \bigcup_{u \in \Sigma_u} \mathrm{Pred}^u(X)$$

Computing winning states

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

$$\mathrm{cPred}(X) = \bigcup_{c \in \Sigma_c} \mathrm{Pred}^c(X) \qquad \mathrm{uPred}(X) = \bigcup_{u \in \Sigma_u} \mathrm{Pred}^u(X)$$

- time controllable predecessor of X:

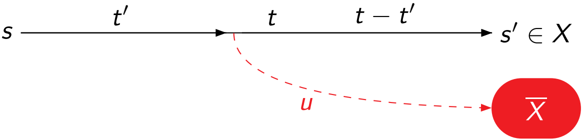
$$s \xrightarrow{\quad t \quad} s' \in X$$

Computing winning states

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

$$\mathrm{cPred}(X) = \bigcup_{c \in \Sigma_c} \mathrm{Pred}^c(X) \qquad \mathrm{uPred}(X) = \bigcup_{u \in \Sigma_u} \mathrm{Pred}^u(X)$$

- time controllable predecessor of X:

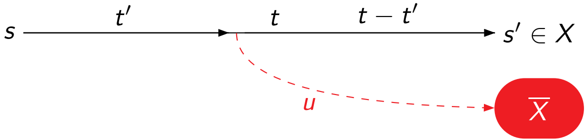


Computing winning states

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

$$\mathrm{cPred}(X) = \bigcup_{c \in \Sigma_c} \mathrm{Pred}^c(X) \qquad \mathrm{uPred}(X) = \bigcup_{u \in \Sigma_u} \mathrm{Pred}^u(X)$$

- time controllable predecessor of X:



$$\mathrm{Pred}_\delta(X, Y) = \{s \mid \exists t \geq 0,\ s \xrightarrow{t} s',\ s' \in X \text{ and } \mathrm{Post}_{[0,t]}(s) \subseteq \overline{Y}\}$$

where $\mathrm{Post}_{[0,t]}(s) = \{s' \mid \exists 0 \leq t' \leq t,\ s \xrightarrow{t'} s'\}$

Computing winning states

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

Proposition (Attractor)

The greatest fixpoint W^* of the equation $X = G \cap \pi(X)$ is the set of states from which we can stay in G.

Properties of W^*

- if X is a union of regions, then $\pi(X)$ is a union of regions
- W^* is effectively computable using zones

Exercise

We take $\mathcal{R}$ the set of regions of the plant.

Let R be a region, and $(R_i)_{i \in I}$ be regions. Then,

- $\mathrm{uPred}(\ell, R)$ is a finite set of regions
- $\mathrm{cPred}(\ell, R)$ is a finite set of regions
- $\mathrm{Pred}_\delta((\ell, R), \cup_{i \in I}(\ell_i, R_i))$ is a finite union of regions

Exercise

We take $\mathcal{R}$ the set of regions of the plant.

Let R be a region, and $(R_i)_{i \in I}$ be regions. Then,

- $\mathrm{uPred}(\ell, R)$ is a finite set of regions
- $\mathrm{cPred}(\ell, R)$ is a finite set of regions
- $\mathrm{Pred}_\delta((\ell, R), \cup_{i \in I}(\ell_i, R_i))$ is a finite union of regions

Why is that true?

Exercise

We take $\mathcal{R}$ the set of regions of the plant.

Let R be a region, and $(R_i)_{i \in I}$ be regions. Then,

- $\mathrm{uPred}(\ell, R)$ is a finite set of regions
- $\mathrm{cPred}(\ell, R)$ is a finite set of regions
- $\mathrm{Pred}_\delta((\ell, R), \cup_{i \in I}(\ell_i, R_i))$ is a finite union of regions

Why is that true?

Region-equivalence is a time-abstract bisimulation!

From winning states to winning strategies

- From winning states, we can construct a controller (adding invariants to the plant and restricting guards of the plant)
- To synthesize a real strategy, we need a more involved computation: if R is a thin region on which the strategy is λ, and if on the successor region of R, say R', the strategy is defined as being c, then split R' in two parts, the first one on which we define the strategy as being λ, and the second one on which the strategy is defined as being c

➜ Computations using polyhedra

Decidability and complexity

Theorem [Henzinger, Kopke 1999]

Safety and reachability control are decidable and are EXPTIME-complete.

➜ simulation of an alternating Turing machine using polynomial space

Decidability and complexity

Theorem [Henzinger, Kopke 1999]

Safety and reachability control are decidable and are EXPTIME-complete.

➜ simulation of an alternating Turing machine using polynomial space

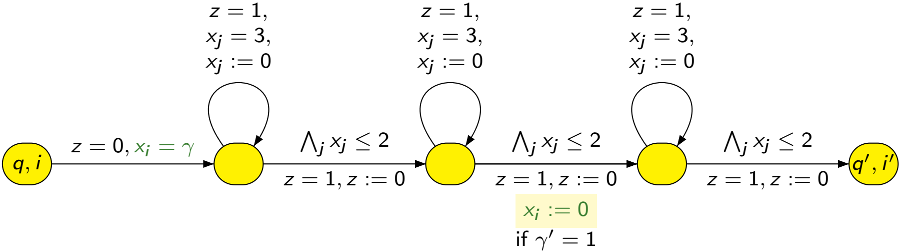
M ATM using $p(\cdot)$ space, and w input for M.

We construct the plant $\mathcal{P}$ as follows:

- set of states contains $Q \times \{1, \ldots, p(|w|)\}$
  - if $q \in Q$ is an AND-node of M, then all outgoing transitions from some (q, i) will be uncontrollable
  - if $q \in Q$ is an OR-node of M, then all outgoing transitions from some (q, i) will be controllable
- set of clocks is $\{x_i \mid 1 \leq i \leq p(|w|)\}$

Cell i of M contains $\gamma \in \{0,1,2\}$ encoded by $x_i = \gamma$

If $(q, \gamma, q', \gamma', \delta)$ is a transition of M, then defining $i' = \delta(i)$,



$\mathcal{P}$ can be controlled to enforce final state
iff
M accepts w

External specifications

External specifications: given by a timed automaton $\mathcal{S}$

- representing desired behaviours

$$L(\mathcal{P} \parallel \mathcal{C}) \subseteq L(\mathcal{S})$$

- representing undesired behaviours

$$L(\mathcal{P} \parallel \mathcal{C}) \cap L(\mathcal{S}) = \emptyset$$

Several undecidability results

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing desired behaviours is undecidable.

Several undecidability results

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing desired behaviours is undecidable.

➜ by reduction of universality problem for timed automata

Several undecidability results

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing desired behaviours is undecidable.

➜ by reduction of universality problem for timed automata

- take $\mathcal{A}$ a timed automaton over Σ

Several undecidability results

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing desired behaviours is undecidable.

➜ by reduction of universality problem for timed automata

- take $\mathcal{A}$ a timed automaton over Σ
- take $\mathcal{P}$ universal plant over Σ

Several undecidability results

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing desired behaviours is undecidable.

➜ by reduction of universality problem for timed automata

- take $\mathcal{A}$ a timed automaton over Σ
- take $\mathcal{P}$ universal plant over Σ
- assume all actions of Σ are **uncontrollable**

  Note: If $\mathcal{C}$ controller, $L(\mathcal{P} \parallel \mathcal{C})$ is universal.

Several undecidability results

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing desired behaviours is undecidable.

➜ by reduction of universality problem for timed automata

- take $\mathcal{A}$ a timed automaton over Σ
- take $\mathcal{P}$ universal plant over Σ
- assume all actions of Σ are **uncontrollable**

  Note: If $\mathcal{C}$ controller, $L(\mathcal{P} \parallel \mathcal{C})$ is universal.

There exists a controller for $\mathcal{P}$ w.r.t. the positive specification $\mathcal{A}$
iff
$\mathcal{A}$ is universal

Several undecidability results (cont'd)

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing undesired behaviours is undecidable.

Several undecidability results (cont'd)

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing undesired behaviours is undecidable.

➜ by reduction of non-universality problem for timed automata

Several undecidability results (cont'd)

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing undesired behaviours is undecidable.

➜ by reduction of non-universality problem for timed automata

- take $\mathcal{A}$ a timed automaton over Σ

Several undecidability results (cont'd)

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing undesired behaviours is undecidable.

➜ by reduction of non-universality problem for timed automata

- take $\mathcal{A}$ a timed automaton over Σ
- take $\mathcal{P}$ universal plant over Σ

Several undecidability results (cont'd)

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing undesired behaviours is undecidable.

➜ by reduction of non-universality problem for timed automata

- take $\mathcal{A}$ a timed automaton over Σ
- take $\mathcal{P}$ universal plant over Σ
- assume all actions of Σ are **controllable**

Several undecidability results (cont'd)

[D'Souza, Madhusudan 2002]

Theorem

Timed control with an external specification representing undesired behaviours is undecidable.

➜ by reduction of non-universality problem for timed automata

- take $\mathcal{A}$ a timed automaton over Σ
- take $\mathcal{P}$ universal plant over Σ
- assume all actions of Σ are **controllable**

There exists a controller $\mathcal{C}$ such that $L(\mathcal{P} \parallel \mathcal{C}) \cap L(\mathcal{A}) = \emptyset$
iff
$\mathcal{A}$ is non-universal

Indeed, any timed word not accepted by $\mathcal{A}$ is a controller...

Some more decidability results

- **Deterministic specification**

Theorem

Timed control with an external **deterministic** specification is decidable.

A controller however needs to use clocks of the plant and of the specification...

Some more decidability results

- **Deterministic specification**

Theorem

Timed control with an external **deterministic** specification is decidable.

A controller however needs to use clocks of the plant and of the specification...

- **Fixing the resources of the controller**

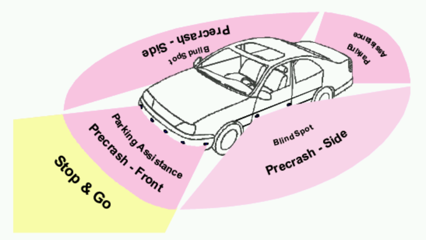
Theorem

Timed control with an external specification representing undesired behaviours is decidable when the resources of the controller are fixed.

(See later)

Why partial observation?

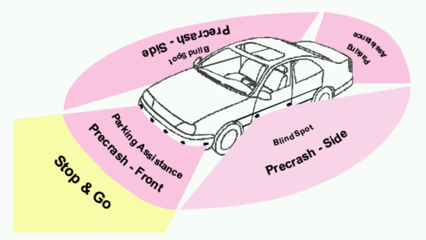
Example (The car periphery supervision)



Environment is seen through **sensors**.

Why partial observation?

Example (The car periphery supervision)

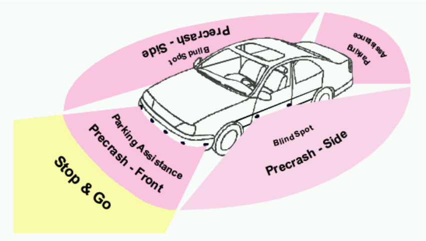


Environment is seen through **sensors**.

- some actions are non-controllable

Why partial observation?

Example (The car periphery supervision)



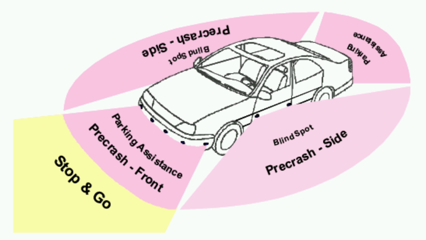
Environment is seen through **sensors**.

- some actions are non-controllable
- some non-controllable actions are even non-observable

[Partial observability]

Why partial observation?

Example (The car periphery supervision)



Environment is seen through **sensors**.

- some actions are non-controllable
- some non-controllable actions are even non-observable

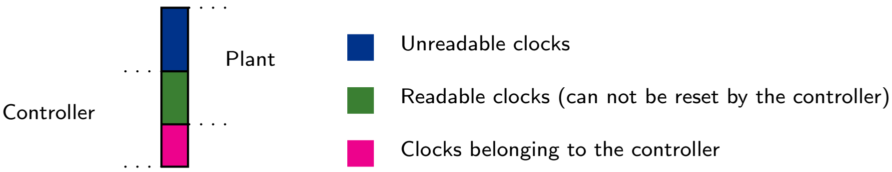
[Partial observability]

Difficulties:

- ε-transitions can not be removed from timed automata
- timed automata can not be determinized

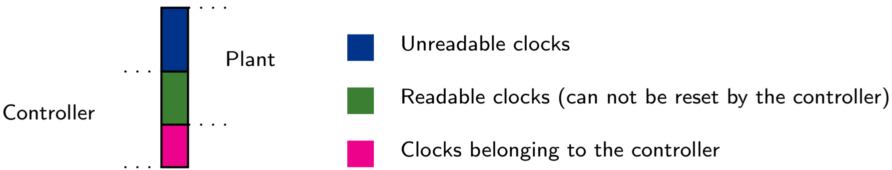
Timed framework: what's specific?

- Clocks of the plant can be readable or unreadable (for the controller)



Timed framework: what's specific?

- Clocks of the plant can be readable or unreadable (for the controller)

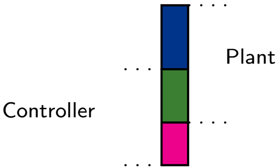


- Which constants clocks can be compared with? $\rightsquigarrow$ $(X, m, \max)$

$$x \sim c \quad \Longrightarrow \quad c \in \frac{\mathbb{Z}}{m} \ \text{ and } \ |c| \leq \max$$

Timed framework: what's specific?

- Clocks of the plant can be readable or unreadable (for the controller)



Unreadable clocks

Readable clocks (can not be reset by the controller)

Clocks belonging to the controller

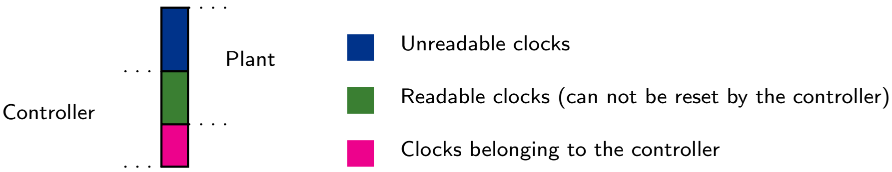
- Which constants clocks can be compared with? $\rightsquigarrow$ $(X, m, \max)$

$$x \sim c \quad \Longrightarrow \quad c \in \frac{\mathbb{Z}}{m} \quad \text{and} \quad |c| \leq \max$$

➜ **Resources**

Timed framework: what's specific?

- Clocks of the plant can be readable or unreadable (for the controller)



- Which constants clocks can be compared with? $\rightsquigarrow$ $(X, m, \max)$

$$x \sim c \quad \Longrightarrow \quad c \in \frac{\mathbb{Z}}{m} \text{ and } |c| \leq \max$$

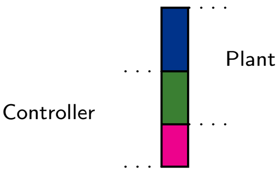
➜ **Resources**

Two different problems:

- fixing the resources, does there exist a controller s.t. ...?

Timed framework: what's specific?

- Clocks of the plant can be readable or unreadable (for the controller)



Unreadable clocks

Readable clocks (can not be reset by the controller)

Clocks belonging to the controller

- Which constants clocks can be compared with? $\rightsquigarrow$ $(X, m, \max)$

$$x \sim c \quad \Longrightarrow \quad c \in \frac{\mathbb{Z}}{m} \quad \text{and} \quad |c| \leq \max$$

➜ **Resources**

Two different problems:

- fixing the resources, does there exist a controller s.t. ...?
- do there exist resources s.t. there exists a controller s.t. ...?

Summary of previous results

Full observability hypothesis			
Resources	Det. Spec. (Internal/External)	External Non-deterministic Spec.	
		Desired behaviors	Undesired behaviors
Fixed	Decidable [WTH91,AMPS98]	Undecidable [DM02]	Decidable [DM02]
Non-fixed	Decidable [DM02]	Undecidable [DM02]	Undecidable [DM02]

Summary of previous results

Full observability hypothesis			
Resources	Det. Spec. (Internal/External)	External Non-deterministic Spec.	
		Desired behaviors	Undesired behaviors
Fixed	Decidable [WTH91,AMPS98]	Undecidable [DM02]	Decidable [DM02]
Non-fixed	Decidable [DM02]	Undecidable [DM02]	Undecidable [DM02]

Partial observability hypothesis			
Resources	Det. Spec. (Internal/External)	External Non deterministic Spec.	
		Desired behaviors	Undesired behaviors
Fixed	?	Undecidable	?
Non-fixed	?	Undecidable	Undecidable

Notations

- some actions are controllable Σ_c
- some actions are uncontrollable Σ_u
  - some uncontrollable actions are observable Σ_u^o
  - some uncontrollable actions are non-observable Σ_u^n
- a **plant** $\mathcal{P}$ is a DTA over $\Sigma_c \cup \Sigma_u^o \cup \Sigma_u^n$
- a **controller** $\mathcal{C}$ is a DTA over $\Sigma_c \cup \Sigma_u^o$
- the **controlled system** is $\mathcal{P} \parallel \mathcal{C}$ where synchronization is only over observable and controllable events

Synthesis with non-fixed resources

Full observability hypothesis			
Resources	Det. Spec. (Internal/External)	External Non-deterministic Spec.	
		Desired behaviors	Undesired behaviors
Fixed	Decidable [WTH91,AMPS98]	Undecidable [DM02]	Decidable [DM02]
Non-fixed	Decidable [DM02]	Undecidable [DM02]	Undecidable [DM02]

Partial observability hypothesis			
Resources	Det. Spec. (Internal/External)	External Non deterministic Spec.	
		Desired behaviors	Undesired behaviors
Fixed	?	Undecidable	?
Non-fixed	Undecidable [BDMP03]	Undecidable	Undecidable

Remark: reachability and safety control problems are undecidable!

Reachability control under partial observability

Theorem

Reachability control under partial observation is undecidable.

Reachability control under partial observability

Theorem

Reachability control under partial observation is undecidable.

➜ by reduction of universality problem for timed automata

Reachability control under partial observability

Theorem

Reachability control under partial observation is undecidable.

➜ by reduction of universality problem for timed automata

Take $\mathcal{A}$ a (complete) timed automaton. Construct $\mathcal{P}$ as follows.

$$\ell \xrightarrow{g,\, a,\, C:=0} \ell' \quad \text{is replaced by} \quad \ell \xrightarrow{(\ell,\, g,\, a,\, C:=0,\, \ell'),\, z:=0} \bullet \xrightarrow{g \wedge z=0,\, a,\, C:=0} \ell'$$

Reachability control under partial observability

Theorem

Reachability control under partial observation is undecidable.

➜ by reduction of universality problem for timed automata

Take $\mathcal{A}$ a (complete) timed automaton. Construct $\mathcal{P}$ as follows.

$$\ell \xrightarrow{g, a, C := 0} \ell' \quad \text{is replaced by} \quad \ell \xrightarrow{(\ell, g, a, C := 0, \ell'), z := 0} \bullet \xrightarrow{g \wedge z = 0, a, C := 0} \ell'$$

Thus,

- $\mathcal{P}$ is a *deterministic* timed automaton, thus a plant
- $(\delta_0, t_0)(a_0, t'_0)(\delta_1, t_1)(a_1, t'_1)\dots$ is accepted by $\mathcal{P}$ iff $t_i = t'_i$ for every i and $(a_0, t_0)(a_1, t_1)\dots$ is accepted by $\mathcal{A}$ along the path $\delta_0\delta_1\dots$

We note $\Delta = \{(\ell, g, a, C := 0, \ell')$ transition of $\mathcal{A}\}$ and make all actions from Δ non-observable.

Take $\mathcal{A}$ a (complete) timed automaton. Construct $\mathcal{P}$ as follows.

$\ell \xrightarrow{g,\, a,\, C := 0} \ell'$ is replaced by $\ell \xrightarrow{(\ell, g, a, C := 0, \ell'),\, z := 0} \bullet \xrightarrow{g \wedge z = 0,\, a,\, C := 0} \ell'$

There exists a controller $\mathcal{C}$ which enforces non-final states of $\mathcal{P}$
iff
$\mathcal{A}$ is not universal

Take $\mathcal{A}$ a (complete) timed automaton. Construct $\mathcal{P}$ as follows.

$\ell \xrightarrow{g,a,C:=0} \ell'$ is replaced by $\ell \xrightarrow{(\ell,g,a,C:=0,\ell'),\,z:=0} \bullet \xrightarrow{g \wedge z=0,\,a,\,C:=0} \ell'$

There exists a controller $\mathcal{C}$ which enforces non-final states of $\mathcal{P}$
iff
$\mathcal{A}$ is not universal

Indeed, for any timed word $\gamma = (a_0, t_0)(a_1, t_1)\ldots$,

$\mathcal{P} \parallel \gamma$ represents all the possible runs for γ with transitions in $\mathcal{A}$

Safety control under partial observability

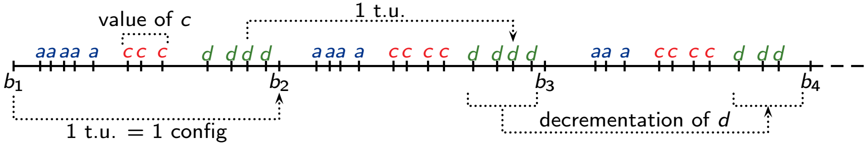
Theorem

Safety control under partial observation is undecidable.

We cannot reduce to the universality problem for timed automata. It requires a more involved proof. We will mimic the proof of the undecidability of the universality problem for timed automata.

➜ by reduction of the non-halting problem for a two-counter machine

Simulation of a two-counter machine



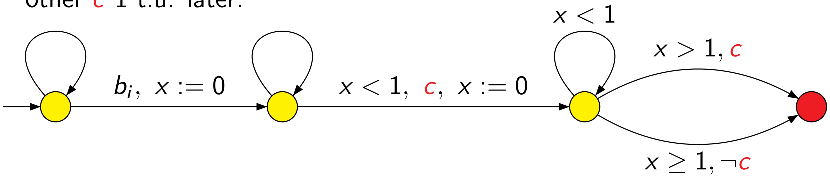
- one configuration is encoded in one time unit
- number of c's: value of counter c
- number of d's: value of counter d
- one time unit between two corresponding c's (resp. d's)
- a finite number of a's is generated (which represents the length of a possible halting computation) at the beginning of the computation, this number decreases by one at each configuration

a, c, d are supposed controllable

partial observability is used for modelling non-determinism

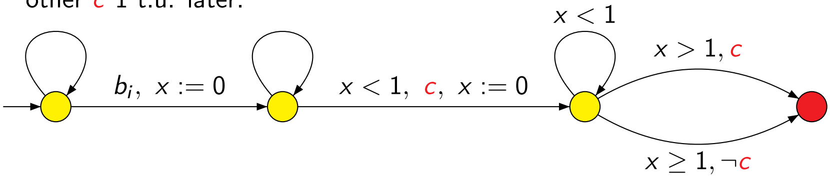
Examples

Module to check that if instruction i does not decrease counter c, then all actions c appearing less than 1 t.u. after b_i has to be followed by an other c 1 t.u. later.

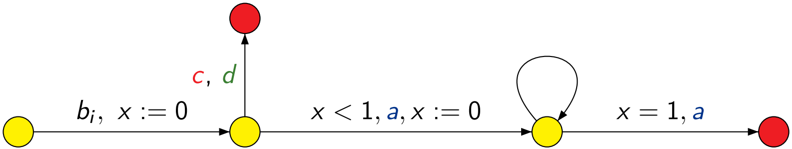


Examples

Module to check that if instruction i does not decrease counter c, then all actions c appearing less than 1 t.u. after b_i has to be followed by an other c 1 t.u. later.

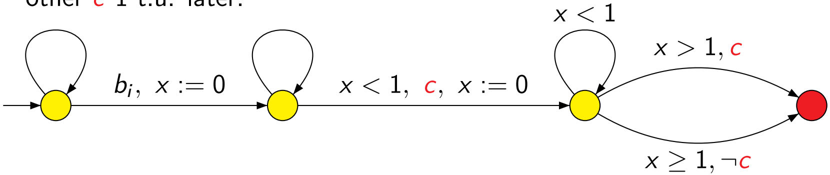


Module to check that the number of a's decreases.

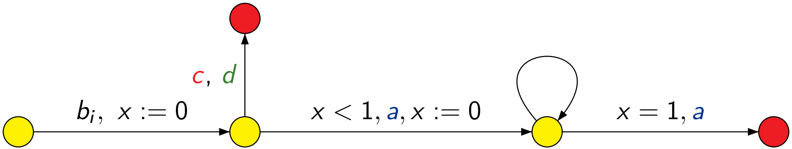


Examples

Module to check that if instruction i does not decrease counter c, then all actions c appearing less than 1 t.u. after b_i has to be followed by an other c 1 t.u. later.



Module to check that the number of a's decreases.



There is a controller to avoid red states
iff
the two-counter machine halts

Fixing the resources

Full observability hypothesis			
Resources	Det. Spec. (Internal/External)	External Non-deterministic Spec.	
		Desired behaviors	Undesired behaviors
Fixed	Decidable [WTH91,AMPS98]	Undecidable [DM02]	Decidable [DM02]
Non-fixed	Decidable [DM02]	Undecidable [DM02]	Undecidable [DM02]

Partial observability hypothesis			
Resources	Det. Spec. (Internal/External)	External Non deterministic Spec.	
		Desired behaviors	Undesired behaviors
Fixed	Decidable [BDMP03]	Undecidable	Decidable [BDMP03]
Non-fixed	Undecidable	Undecidable	Undecidable

Fixing the resources

Theorem

Under partial observability, the controller synthesis problem with fixed resources is decidable, for deterministic specifications or non deterministic specifications representing undesired behaviors.

Fixing the resources

Theorem

Under partial observability, the controller synthesis problem with fixed resources is decidable, for deterministic specifications or non deterministic specifications representing undesired behaviors.

- Controller synthesis problem as a (syntactic) timed game
- Solving a timed game
  - construction of an untimed arena
  - construction of an untimed winning condition
- Apply results on untimed games

- **Resources:** $\mu = (X, m, \max)$

$$x \sim c \quad \Longrightarrow \quad c \in \frac{\mathbb{Z}}{m} \quad \text{and} \quad |c| \leq \max$$

- A controller will be an automaton over *symbolic alphabet*

$$\Gamma = \mathcal{G}(\mu) \times (\Sigma_c \cup \Sigma_u^o) \times 2^X$$

where $\mathcal{G}(\mu)$ represents all *atomic* constraints over μ

Example ($\mu = (\{x, y\}, 1, 2)$)

- $0 < x < 1 \wedge y = 1$ is an atomic constraint
- $1 < x < 2 \wedge y > 2$ is an atomic constraint
- $x = 0 \wedge y \geq 1$ is **not** an atomic constraint

- A transition of the controller is thus of the form

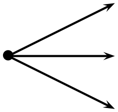
$$\ell \xrightarrow{0<x<1 \wedge y>2,\ a,\ x:=0} \ell'$$

Timed games

On the same symbolic alphabet Γ as the controller:

Timed games

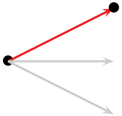
On the same symbolic alphabet Γ as the controller:



Player C

Timed games

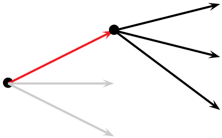
On the same symbolic alphabet Γ as the controller:



Player E

Timed games

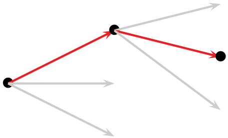
On the same symbolic alphabet Γ as the controller:



Player C

Timed games

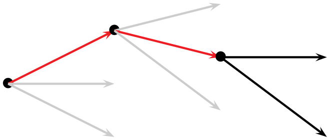
On the same symbolic alphabet Γ as the controller:



Player E

Timed games

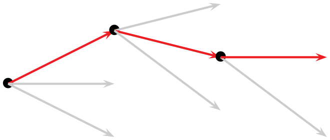
On the same symbolic alphabet Γ as the controller:



Player C

Timed games

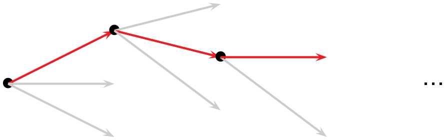
On the same symbolic alphabet Γ as the controller:



Player E

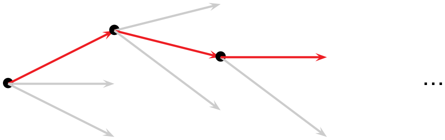
Timed games

On the same symbolic alphabet Γ as the controller:



Timed games

On the same symbolic alphabet Γ as the controller:

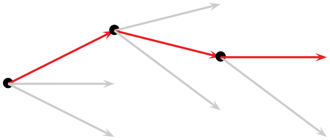


- whether the play γ is winning or not depends on the synchronization of the plant with γ:

$$L(\mathcal{P} \parallel \gamma) \cap L(\mathcal{S}) = \emptyset$$

Timed games

On the same symbolic alphabet Γ as the controller:



...

- whether the play γ is winning or not depends on the synchronization of the plant with γ:

$$L(\mathcal{P} \parallel \gamma) \cap L(\mathcal{S}) = \emptyset$$

- usual notion of strategy, with additional hypotheses for the non-blocking and non-restricting hypotheses

Untimed games

- **Construction of the arena:** universal automaton $\mathcal{U}_\Gamma$ over Γ

Untimed games

- **Construction of the arena:** universal automaton $\mathcal{U}_\Gamma$ over Γ

- **Construction of the winning condition:** based on the fact that the set
$$\{\gamma \in \Gamma^\infty \ \mid\ L(\mathcal{P} \parallel \gamma) \cap L(\mathcal{S}) = \emptyset\}$$
is regular (where Γ is the symbolic alphabet for the controller).

Untimed games

- **Construction of the arena:** universal automaton $\mathcal{U}_\Gamma$ over Γ
    - additional condition for non-restricting hypothesis

- **Construction of the winning condition:** based on the fact that the set

$$\{\gamma \in \Gamma^\infty \;\mid\; L(\mathcal{P} \parallel \gamma) \cap L(\mathcal{S}) = \emptyset\}$$

is regular (where Γ is the symbolic alphabet for the controller).

Untimed games

- **Construction of the arena:** universal automaton $\mathcal{U}_\Gamma$ over Γ
  - additional condition for non-restricting hypothesis
  - what about non-blocking hypothesis?

    ➜ we need a more involved construction with information on $\mathcal{P}$

    (projection of the region automaton of $\mathcal{P} \parallel \mathcal{U}_\Gamma$ onto observable actions and clocks)

- **Construction of the winning condition:** based on the fact that the set

$$\{\gamma \in \Gamma^\infty \mid L(\mathcal{P} \parallel \gamma) \cap L(\mathcal{S}) = \emptyset\}$$

is regular (where Γ is the symbolic alphabet for the controller).

Untimed games

- **Construction of the arena:** universal automaton $\mathcal{U}_\Gamma$ over Γ
  - additional condition for non-restricting hypothesis
  - what about non-blocking hypothesis?

    ➜ we need a more involved construction with information on $\mathcal{P}$

    (projection of the region automaton of $\mathcal{P} \parallel \mathcal{U}_\Gamma$ onto observable actions and clocks)

- **Construction of the winning condition:** based on the fact that the set

$$\{\gamma \in \Gamma^\infty \mid L(\mathcal{P} \parallel \gamma) \cap L(\mathcal{S}) = \emptyset\}$$

is regular (where Γ is the symbolic alphabet for the controller).

➜ classical untimed game

Untimed games

- **Construction of the arena:** universal automaton $\mathcal{U}_\Gamma$ over Γ
  - additional condition for non-restricting hypothesis
  - what about non-blocking hypothesis?

    ➜ we need a more involved construction with information on $\mathcal{P}$

    (projection of the region automaton of $\mathcal{P} \parallel \mathcal{U}_\Gamma$ onto observable actions and clocks)

- **Construction of the winning condition:** based on the fact that the set

$$\{\gamma \in \Gamma^\infty \mid L(\mathcal{P} \parallel \gamma) \cap L(\mathcal{S}) = \emptyset\}$$

is regular (where Γ is the symbolic alphabet for the controller).

➜ classical untimed game

➜ the problem is 2EXPTIME

Untimed games

- **Construction of the arena:** universal automaton $\mathcal{U}_\Gamma$ over Γ
  - additional condition for non-restricting hypothesis
  - what about non-blocking hypothesis?

    ➜ we need a more involved construction with information on $\mathcal{P}$

    (projection of the region automaton of $\mathcal{P} \parallel \mathcal{U}_\Gamma$ onto observable actions and clocks)

- **Construction of the winning condition:** based on the fact that the set

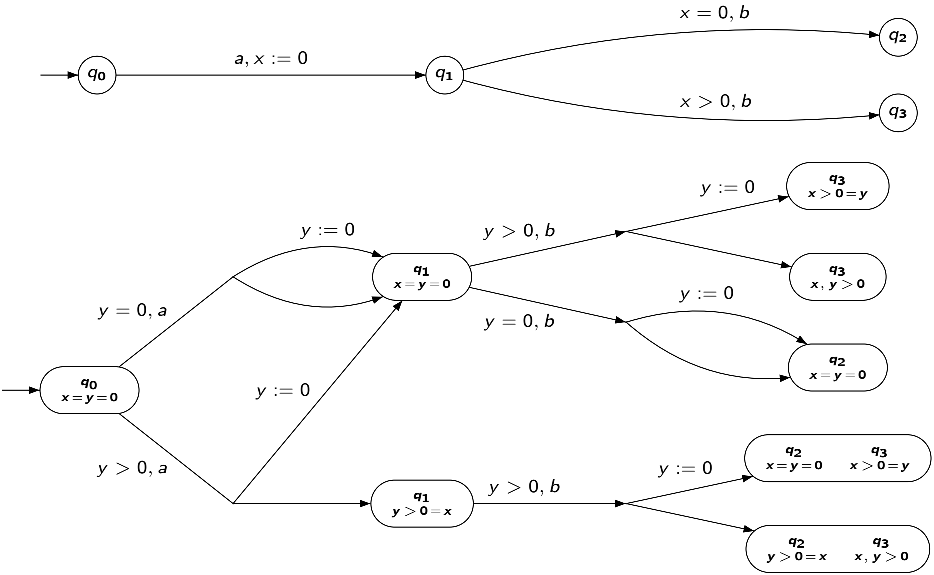
$$\{\gamma \in \Gamma^\infty \mid L(\mathcal{P} \parallel \gamma) \cap L(\mathcal{S}) = \emptyset\}$$

is regular (where Γ is the symbolic alphabet for the controller).

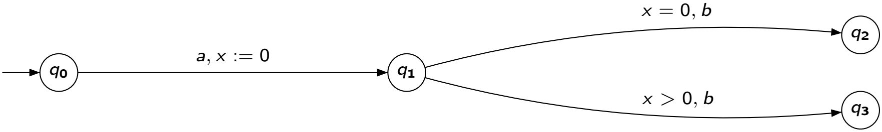
➜ classical untimed game

➜ the problem is 2EXPTIME-complete

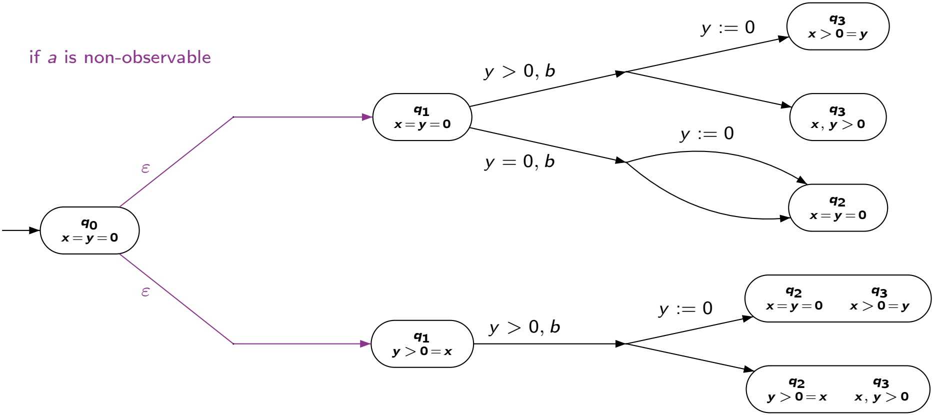
Resources $\mu = (\{y\}, 1, 0)$.



Resources $\mu = (\{y\}, 1, 0)$.



if a is non-observable



Outline

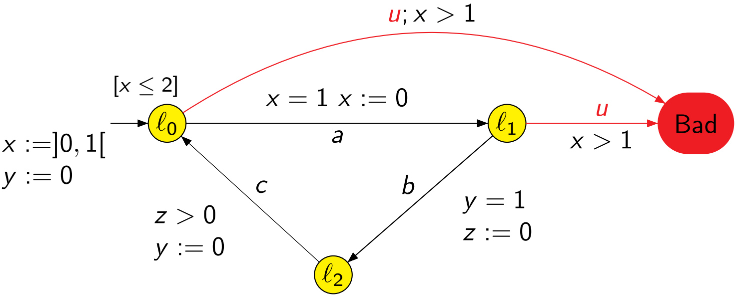
Zenoness...

- is often avoided (by assuming the plant is strictly non-*zeno*)
  [AMPS98,...]

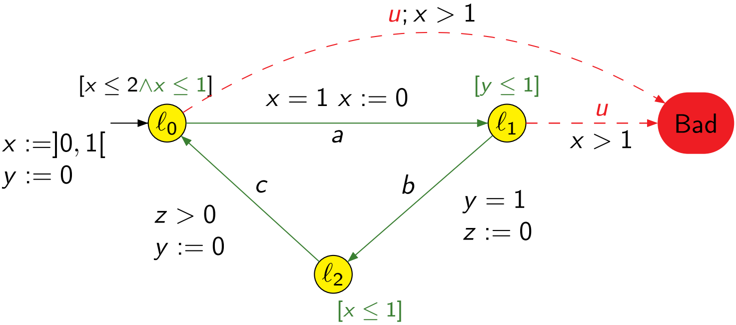
- is incorporated in the winning condition **[dAFH+03]**

Is that sufficient?

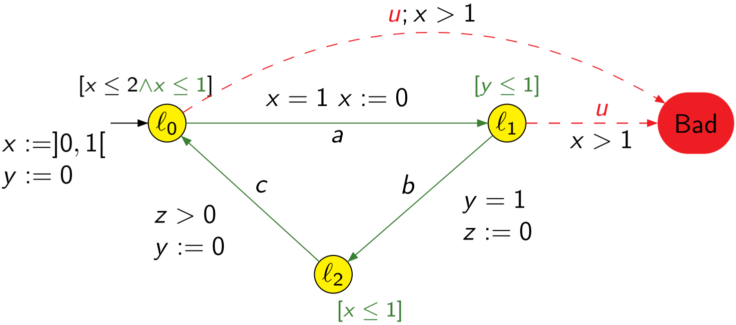
Problems with dense-time control



Problems with dense-time control

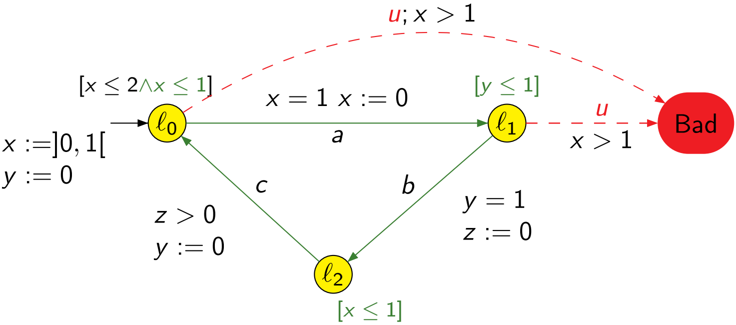


Problems with dense-time control



- δ_i : time in ℓ_2 during loop i
- The controller must ensure: $\sum_{i=1}^{i=+\infty} \delta_i < 1 - x_0$

Problems with dense-time control



- δ_i : time in ℓ_2 during loop i
- The controller must ensure: $\sum_{i=1}^{i=+\infty} \delta_i < 1 - x_0$

This is **impossible** with a *sampling-time* controller, *no matter how fast it is!*

[Cassez, Henzinger, Raskin 2002]

Sampling-time control

[Cassez, Henzinger, Raskin 2002]

- system (within environment) evolves continuously with time
- controller can enforce controllable actions only at discrete dates

Sampling-time control

[Cassez, Henzinger, Raskin 2002]

- system (within environment) evolves continuously with time
- controller can enforce controllable actions only at discrete dates

Question: is there a sampling rate sufficient for controlling the system?

Sampling-time control

[Cassez, Henzinger, Raskin 2002]

- system (within environment) evolves continuously with time
- controller can enforce controllable actions only at discrete dates

Question: is there a sampling rate sufficient for controlling the system?

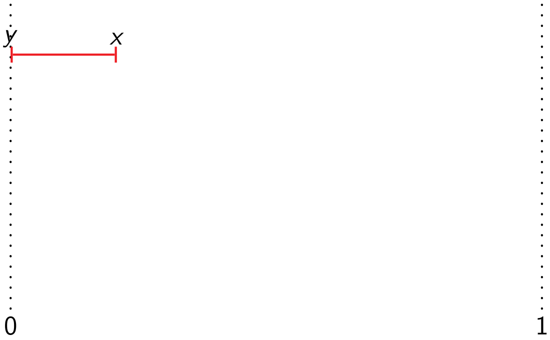
Theorem

Unknown sampling rate safety control is undecidable.

➜ reduction of the halting problem for a two-counter machine.

Idea: if n is the length of a halting computation for the machine, then $\frac{1}{n}$ will be a sampling rate for the control problem

Encoding of the value of a counter:

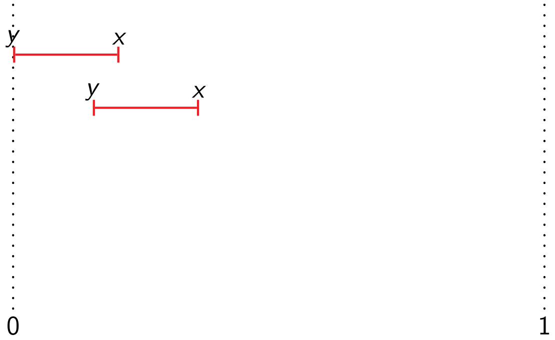


The value of counter c is:

- $(x - y).n$

Idea: if n is the length of a halting computation for the machine, then $\frac{1}{n}$ will be a sampling rate for the control problem

Encoding of the value of a counter:

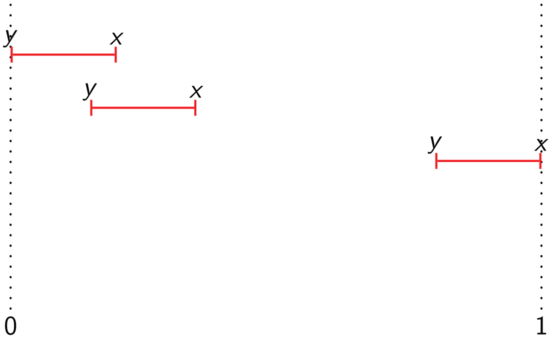


The value of counter c is:

- $(x - y).n$

Idea: if n is the length of a halting computation for the machine, then $\frac{1}{n}$ will be a sampling rate for the control problem

Encoding of the value of a counter:

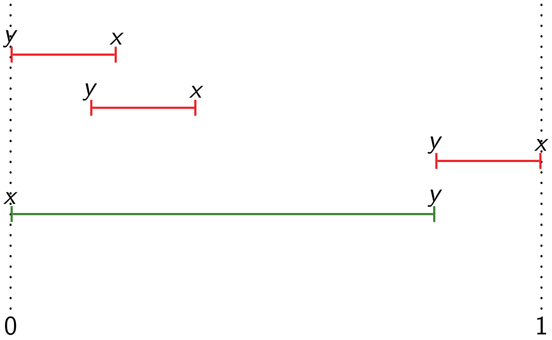


The value of counter c is:

- $(x - y).n$

Idea: if n is the length of a halting computation for the machine, then $\frac{1}{n}$ will be a sampling rate for the control problem

Encoding of the value of a counter:

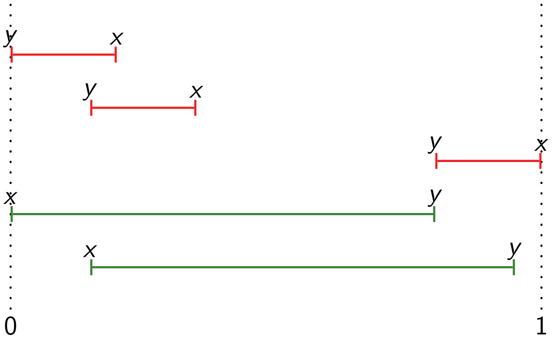


The value of counter c is:

- $(x - y).n$ if $x \geq y$
- $[1 - (y - x)].n$ if $y > x$

Idea: if n is the length of a halting computation for the machine, then $\frac{1}{n}$ will be a sampling rate for the control problem

Encoding of the value of a counter:

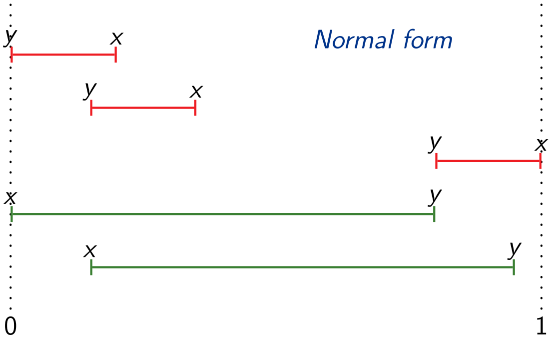


The value of counter c is:

- $(x - y).n$ if $x \geq y$
- $[1 - (y - x)].n$ if $y > x$

Idea: if n is the length of a halting computation for the machine, then $\frac{1}{n}$ will be a sampling rate for the control problem

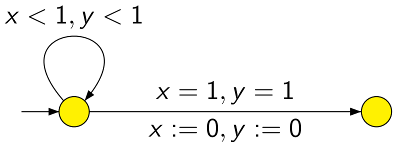
Encoding of the value of a counter:



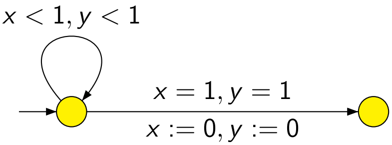
The value of counter c is:

- $(x - y).n$ if $x \geq y$
- $[1 - (y - x)].n$ if $y > x$

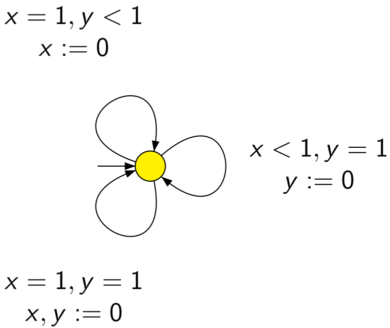
- Zero testing widget



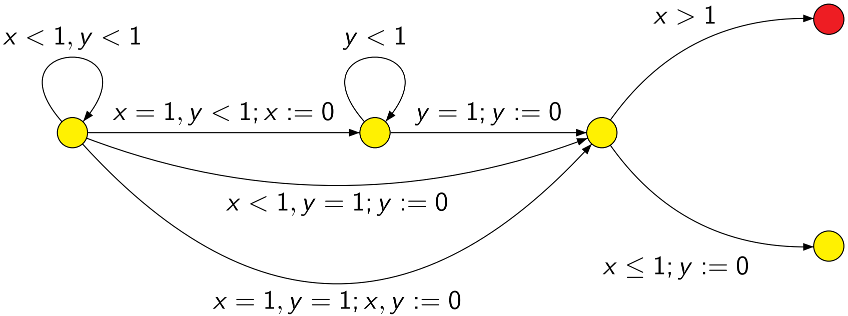
- Zero testing widget



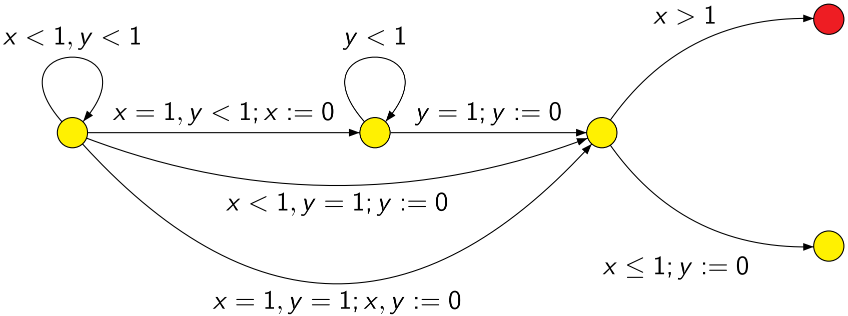
- Idling widget



- Normalization and incrementation widget



- Normalization and incrementation widget



The two-counter machine has an halting computation
iff
there is a sampled time controller for the above systems

Notion of implementability

[De Wulf, Doyen, Raskin 2004]

An **implementable** controller $\mathcal{C}$

- has finite precision (digital clock)
- may delay responses and communications (relaxes synchrony hypothesis)

Notion of implementability

[De Wulf, Doyen, Raskin 2004]

An **implementable** controller $\mathcal{C}$

- has finite precision (digital clock)
- may delay responses and communications
  (relaxes synchrony hypothesis)

➜ defines a TTS $[\![\mathcal{C}]\!]_\Delta$ (where Δ is a parameter)

Notion of implementability

[De Wulf, Doyen, Raskin 2004]

An **implementable** controller $\mathcal{C}$

- has finite precision (digital clock)
- may delay responses and communications
  (relaxes synchrony hypothesis)

➜ defines a TTS $[\![\mathcal{C}]\!]_\Delta$ (where Δ is a parameter)

Proposition

If $\Delta_1 \geq \Delta_2$ and $[\![\mathcal{C}]\!]_{\Delta_1}$ controls $\mathcal{P}$ to avoid bad states, then $[\![\mathcal{C}]\!]_{\Delta_2}$ controls $\mathcal{P}$ to avoid bad states.

Notion of implementability

[De Wulf, Doyen, Raskin 2004]

An **implementable** controller $\mathcal{C}$

- has finite precision (digital clock)
- may delay responses and communications
  (relaxes synchrony hypothesis)

➜ defines a TTS $[\![\mathcal{C}]\!]_\Delta$ (where Δ is a parameter)

Proposition

If $\Delta_1 \geq \Delta_2$ and $[\![\mathcal{C}]\!]_{\Delta_1}$ controls $\mathcal{P}$ to avoid bad states, then $[\![\mathcal{C}]\!]_{\Delta_2}$ controls $\mathcal{P}$ to avoid bad states.

Proposition

If $[\![\mathcal{C}]\!]_\Delta$ controls $\mathcal{P}$, then $\mathcal{C}$ can be implemented on a sufficiently fast hardware.

Notion of implementability

[De Wulf, Doyen, Raskin 2004]

An **implementable** controller $\mathcal{C}$

- has finite precision (digital clock)
- may delay responses and communications (relaxes synchrony hypothesis)

➜ defines a TTS $[\![\mathcal{C}]\!]_\Delta$ (where Δ is a parameter)

Proposition

If $\Delta_1 \geq \Delta_2$ and $[\![\mathcal{C}]\!]_{\Delta_1}$ controls $\mathcal{P}$ to avoid bad states, then $[\![\mathcal{C}]\!]_{\Delta_2}$ controls $\mathcal{P}$ to avoid bad states.

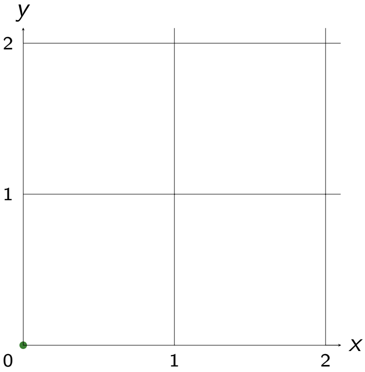
Proposition

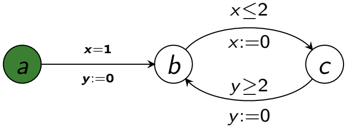
If $[\![\mathcal{C}]\!]_\Delta$ controls $\mathcal{P}$, then $\mathcal{C}$ can be implemented on a sufficiently fast hardware.

➜ it is sufficient to study the Δ-enlarged semantics

An example: standard semantics

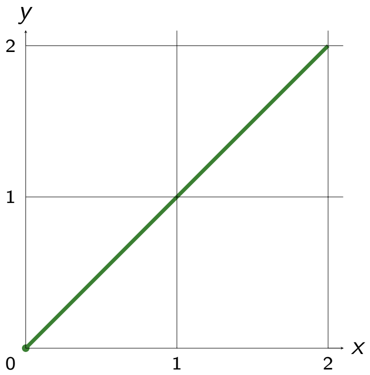
[De Wulf, Doyen, Markey, Raskin 2004]

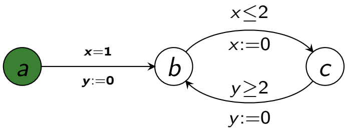




An example: standard semantics

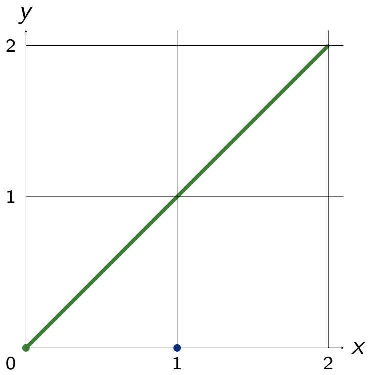
[De Wulf, Doyen, Markey, Raskin 2004]

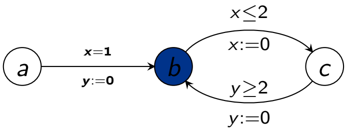




An example: standard semantics

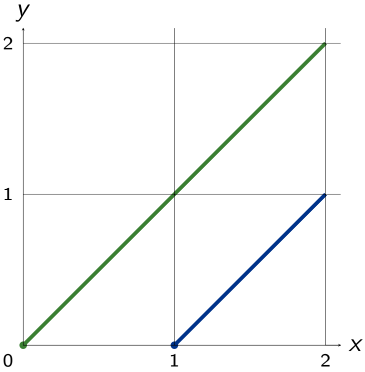
[De Wulf, Doyen, Markey, Raskin 2004]

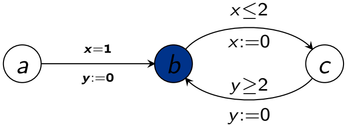




An example: standard semantics

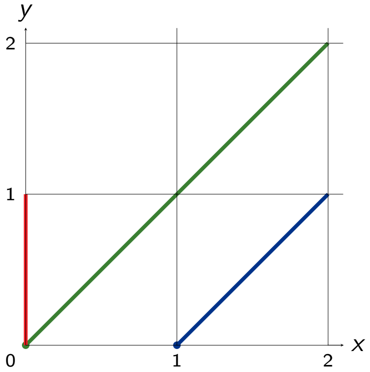
[De Wulf, Doyen, Markey, Raskin 2004]

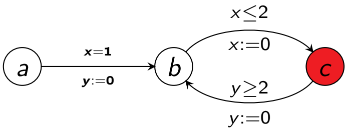




An example: standard semantics

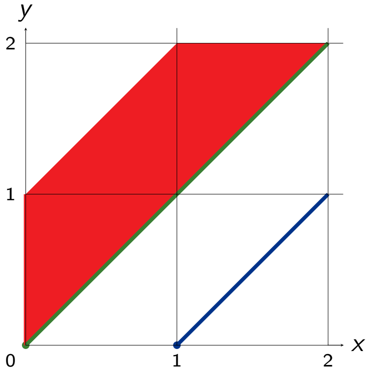
[De Wulf, Doyen, Markey, Raskin 2004]

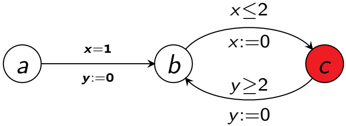




An example: standard semantics

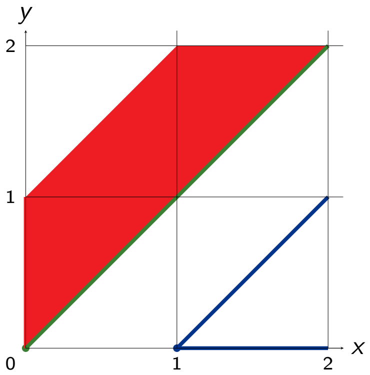
[De Wulf, Doyen, Markey, Raskin 2004]

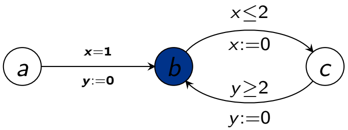




An example: standard semantics

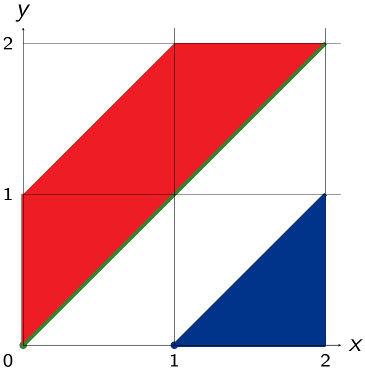
[De Wulf, Doyen, Markey, Raskin 2004]

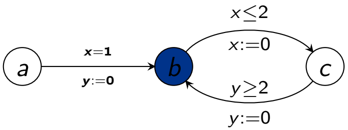




An example: standard semantics

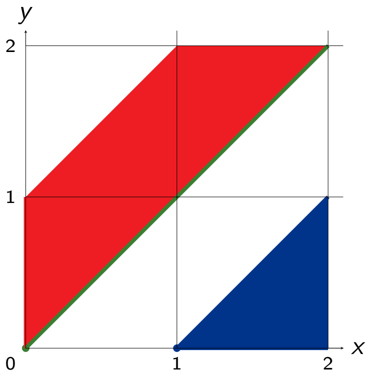
[De Wulf, Doyen, Markey, Raskin 2004]

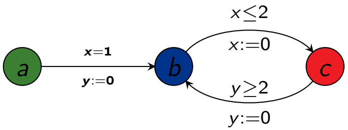




An example: standard semantics

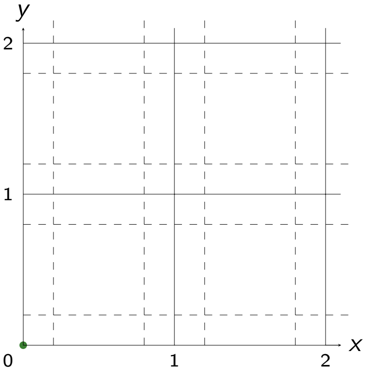
[De Wulf, Doyen, Markey, Raskin 2004]

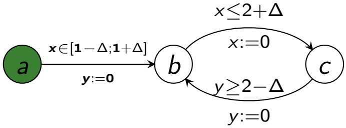




An example with $\Delta > 0$

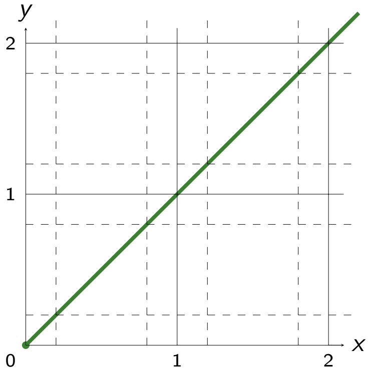
[De Wulf, Doyen, Markey, Raskin 2004]

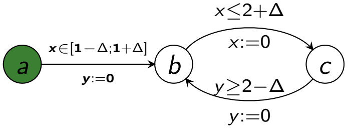




An example with $\Delta > 0$

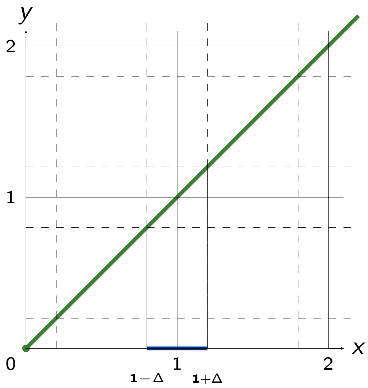
[De Wulf, Doyen, Markey, Raskin 2004]

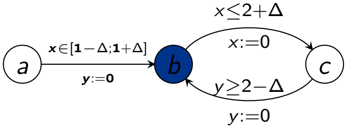




An example with $\Delta > 0$

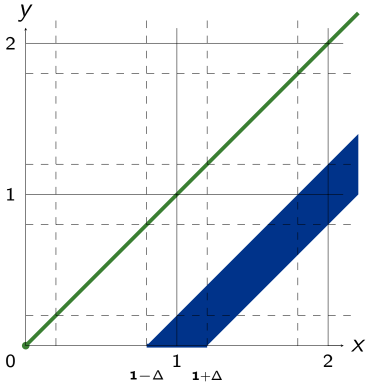
[De Wulf, Doyen, Markey, Raskin 2004]

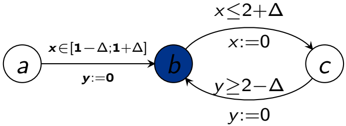




An example with $\Delta > 0$

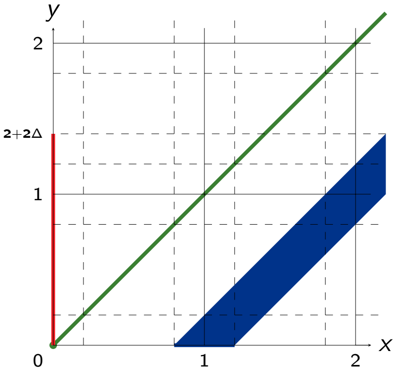
[De Wulf, Doyen, Markey, Raskin 2004]

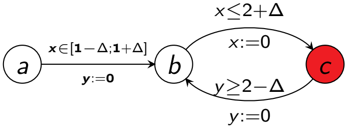




An example with $\Delta > 0$

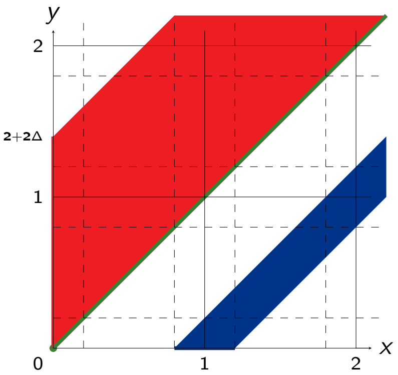
[De Wulf, Doyen, Markey, Raskin 2004]

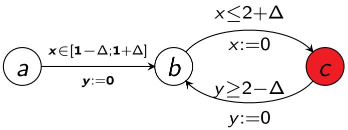




An example with $\Delta > 0$

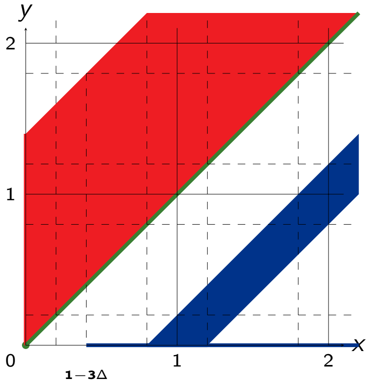
[De Wulf, Doyen, Markey, Raskin 2004]

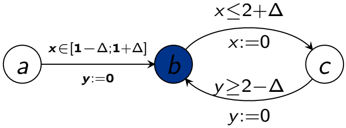




An example with $\Delta > 0$

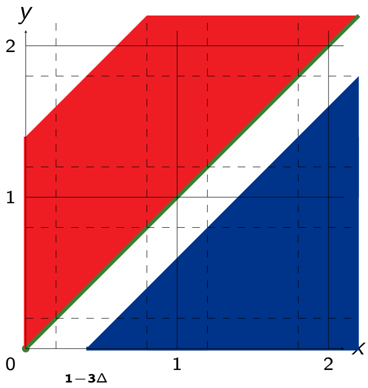
[De Wulf, Doyen, Markey, Raskin 2004]

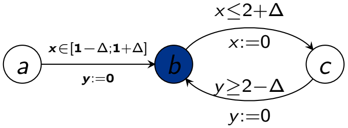




An example with $\Delta > 0$

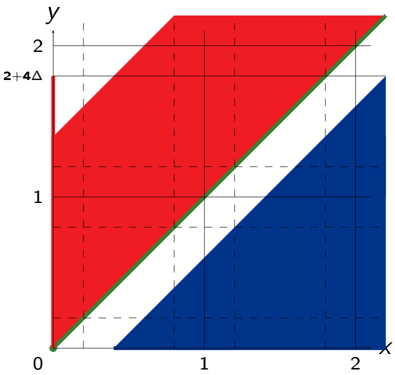
[De Wulf, Doyen, Markey, Raskin 2004]

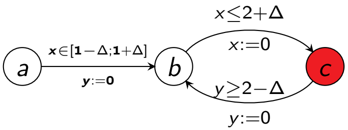




An example with $\Delta > 0$

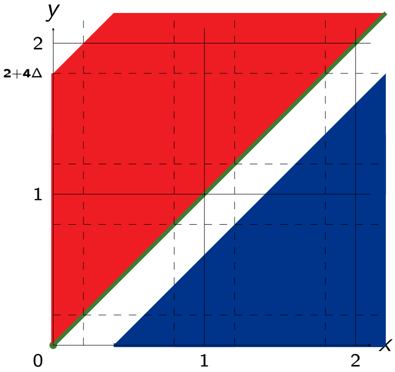
[De Wulf, Doyen, Markey, Raskin 2004]

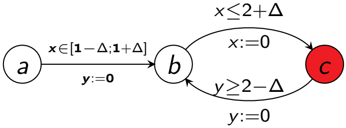




An example with $\Delta > 0$

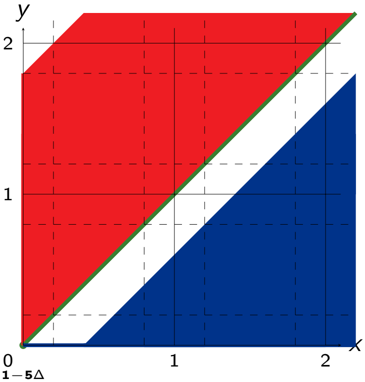
[De Wulf, Doyen, Markey, Raskin 2004]

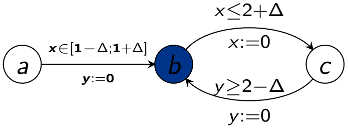




An example with $\Delta > 0$

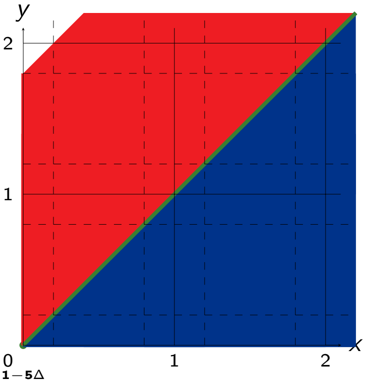
[De Wulf, Doyen, Markey, Raskin 2004]

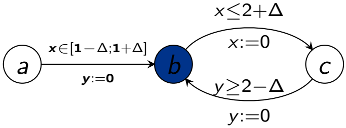




An example with $\Delta > 0$

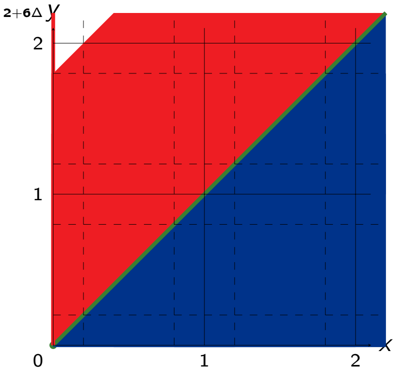
[De Wulf, Doyen, Markey, Raskin 2004]

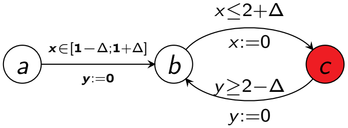




An example with $\Delta > 0$

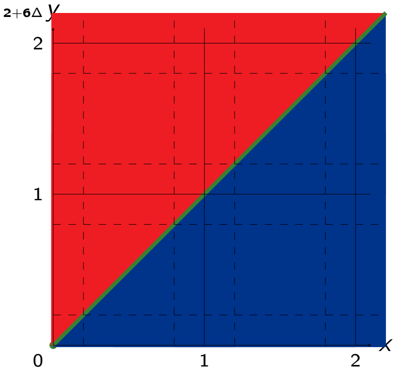
[De Wulf, Doyen, Markey, Raskin 2004]

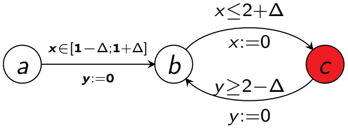




An example with $\Delta > 0$

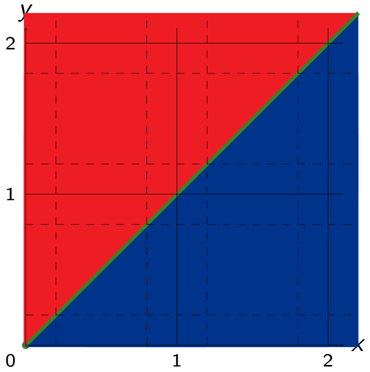
[De Wulf, Doyen, Markey, Raskin 2004]

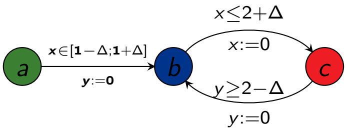




An example with $\Delta > 0$

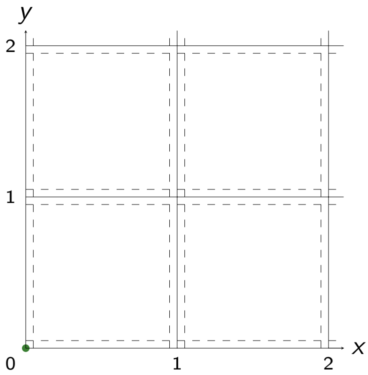
[De Wulf, Doyen, Markey, Raskin 2004]

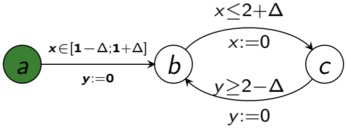




An example with Δ very small

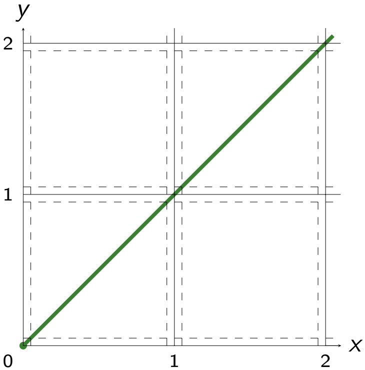
[De Wulf, Doyen, Markey, Raskin 2004]

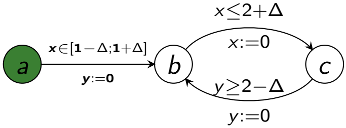




An example with Δ very small

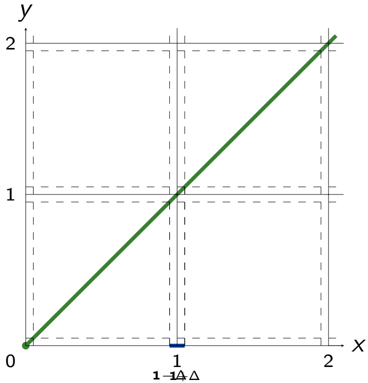
[De Wulf, Doyen, Markey, Raskin 2004]

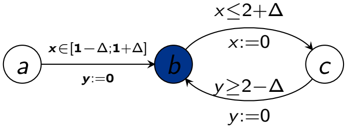




An example with Δ very small

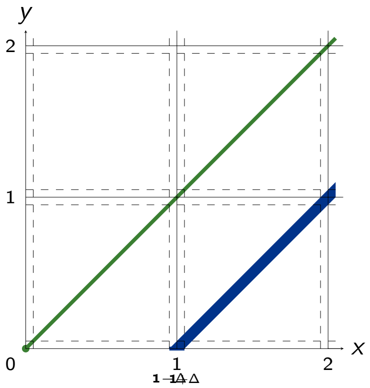
[De Wulf, Doyen, Markey, Raskin 2004]

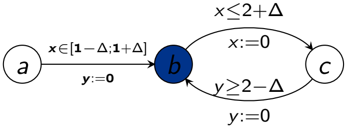




An example with Δ very small

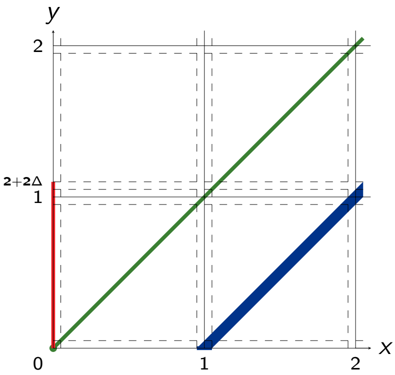
[De Wulf, Doyen, Markey, Raskin 2004]

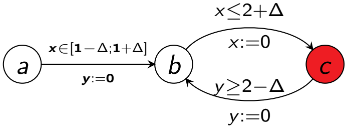




An example with Δ very small

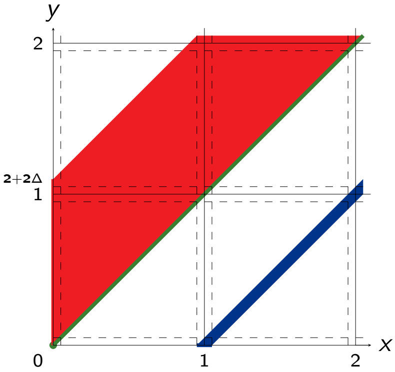
[De Wulf, Doyen, Markey, Raskin 2004]

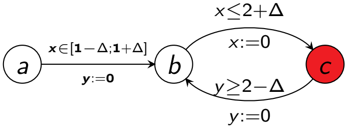




An example with Δ very small

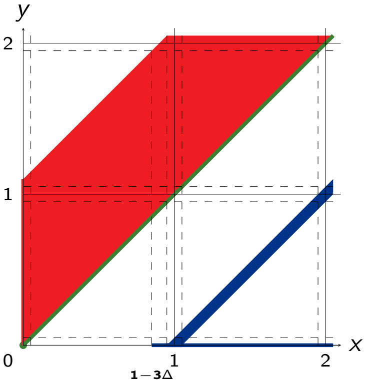
[De Wulf, Doyen, Markey, Raskin 2004]

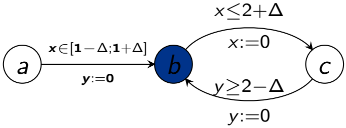




An example with Δ very small

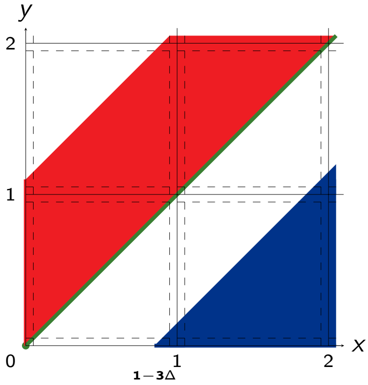
[De Wulf, Doyen, Markey, Raskin 2004]

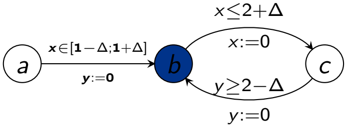




An example with Δ very small

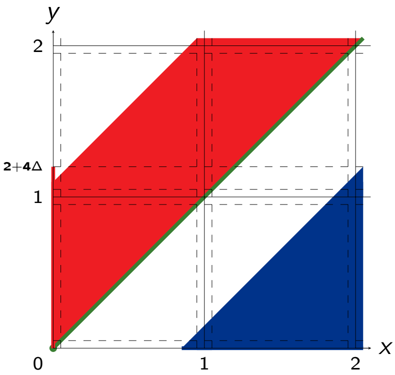
[De Wulf, Doyen, Markey, Raskin 2004]

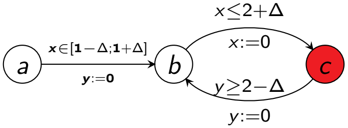




An example with Δ very small

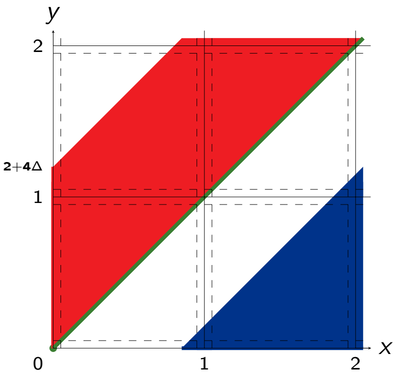
[De Wulf, Doyen, Markey, Raskin 2004]

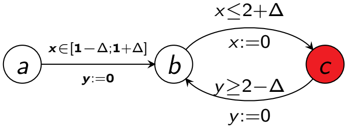




An example with Δ very small

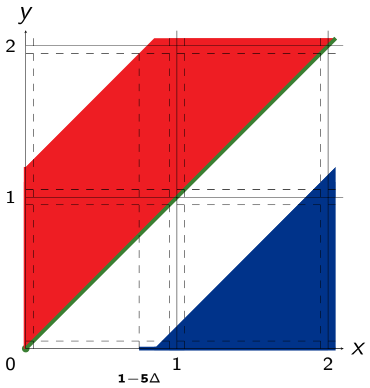
[De Wulf, Doyen, Markey, Raskin 2004]

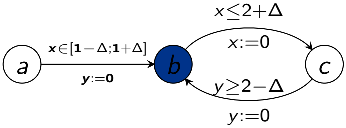




An example with Δ very small

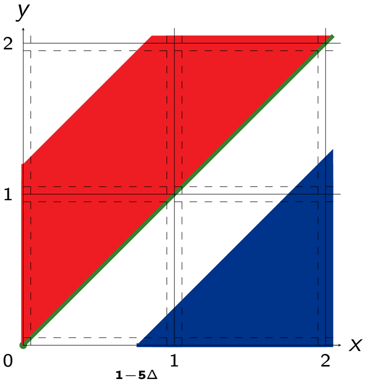
[De Wulf, Doyen, Markey, Raskin 2004]

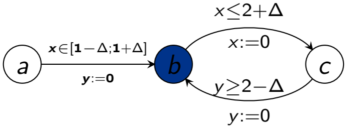




An example with Δ very small

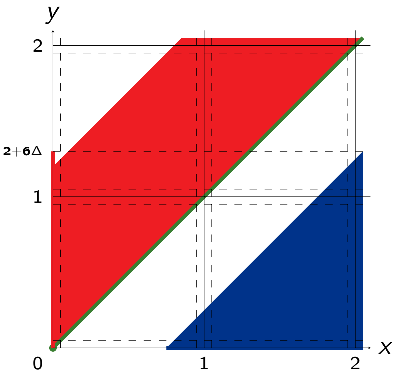
[De Wulf, Doyen, Markey, Raskin 2004]

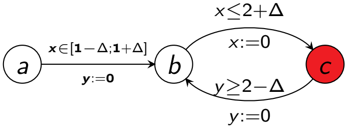




An example with Δ very small

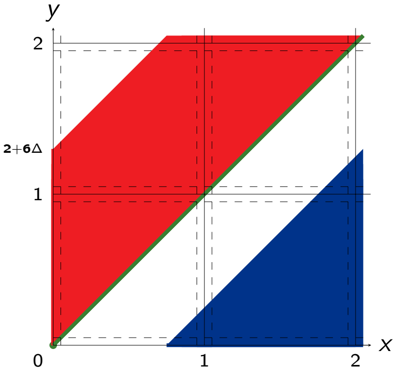
[De Wulf, Doyen, Markey, Raskin 2004]

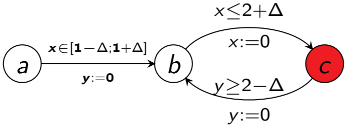




An example with Δ very small

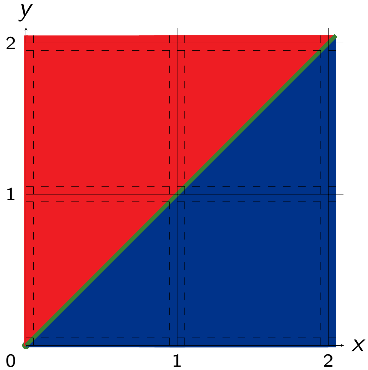
[De Wulf, Doyen, Markey, Raskin 2004]

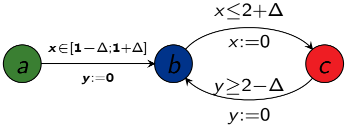




An example with Δ very small

[De Wulf, Doyen, Markey, Raskin 2004]





Deciding implementability

[De Wulf, Doyen, Markey, Raskin 2004]

Implementability problem: given a timed automaton A and a set of bad states “Bad”, does there exists $\Delta > 0$ such that

$$[\![A^\Delta]\!] \cap \mathrm{Bad} = \emptyset$$

Theorem

Implementability is decidable for timed automata.

(Using an extension of region automaton construction)

Outline

Conclusion & current developments

Conclusion

- Much literature about timed control/games these last ten years
- Structural properties of winning strategies highly depend on semantics which is chosen
- We have presented here "control timed games", a framework suitable to model open systems, and several (un)decidability results

Conclusion & current developments

Conclusion

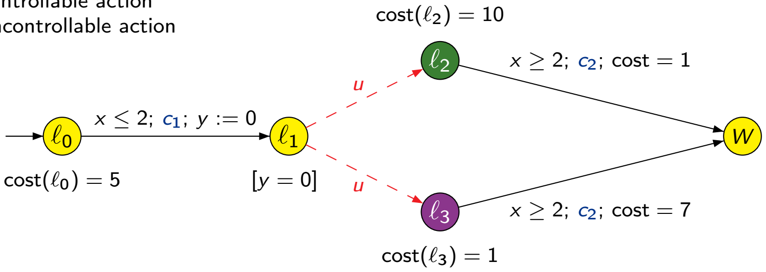
- Much literature about timed control/games these last ten years
- Structural properties of winning strategies highly depend on semantics which is chosen
- We have presented here "control timed games", a framework suitable to model open systems, and several (un)decidability results

Current developments

- Synthesis of optimal controllers
  - time-optimal controllers **[Asarin, Maler 1999]**
  - cost-optimal controllers (see after)
- Synthesis of implementable controllers
- Better understand partial observability
  - Concentrate on fault diagnosis
  - Link with testing

Synthesis of optimal controllers [LMM02,ABM04,BCFL04,BCFL05]

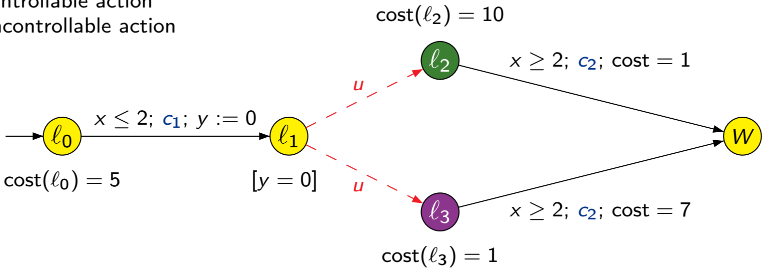
c_i: controllable action
u: uncontrollable action



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

Synthesis of optimal controllers [LMM02,ABM04,BCFL04,BCFL05]

c_i: controllable action
u: uncontrollable action

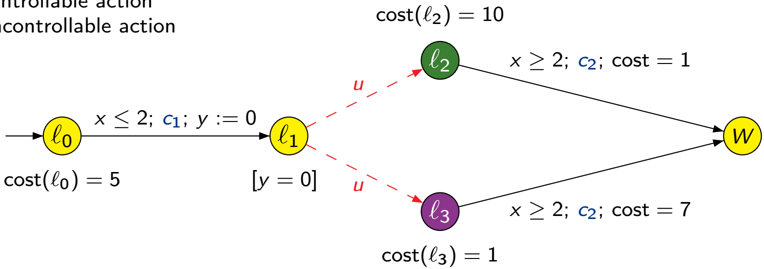


Question: what is the optimal price we can ensure in state ℓ_0?

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

Synthesis of optimal controllers [LMM02,ABM04,BCFL04,BCFL05]

c_i: controllable action
u: uncontrollable action

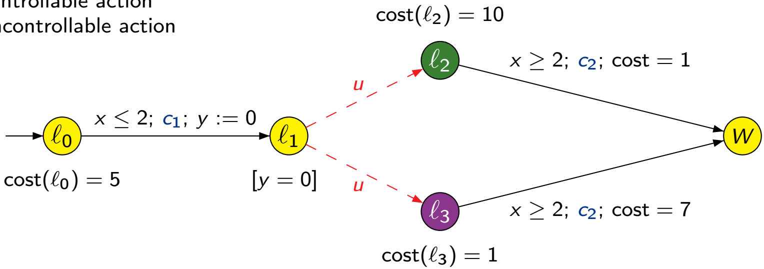


Question: what is the optimal price we can ensure in state ℓ_0?

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

Synthesis of optimal controllers [LMM02,ABM04,BCFL04,BCFL05]

c_i: controllable action
u: uncontrollable action

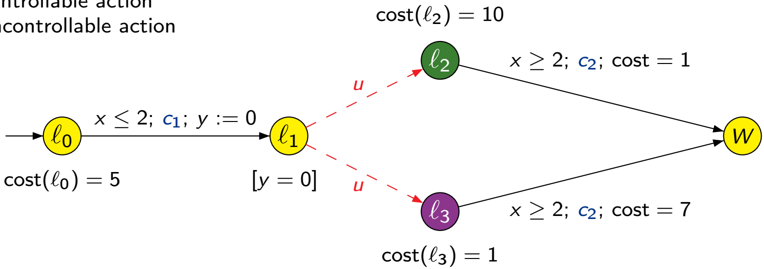


Question: what is the optimal price we can ensure in state ℓ_0?

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

Synthesis of optimal controllers [LMM02,ABM04,BCFL04,BCFL05]

c_i: controllable action
u: uncontrollable action

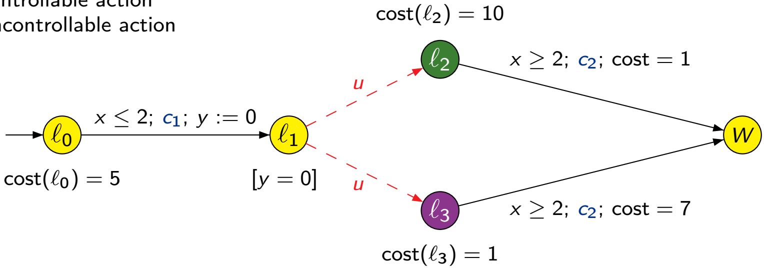


Question: what is the optimal price we can ensure in state ℓ_0?

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

Synthesis of optimal controllers [LMM02,ABM04,BCFL04,BCFL05]

c_i: controllable action
u: uncontrollable action

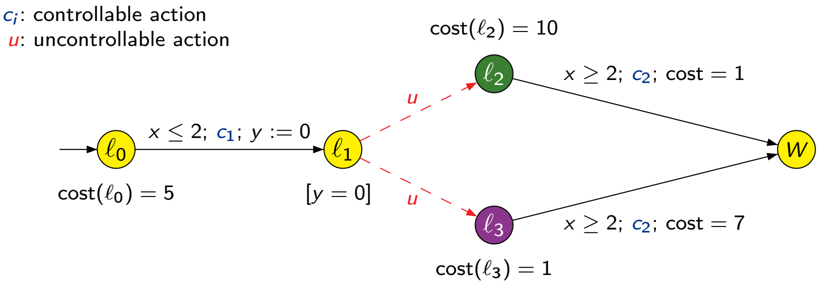


Question: what is the optimal price we can ensure in state ℓ_0?

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

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

Synthesis of optimal controllers [LMM02,ABM04,BCFL04,BCFL05]



Question: what is the optimal price 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

- region partitioning is not sufficient
- optimal winning strategies may need memory
- computability with no assumption on cost is an open problem

Bibliography I

[ABM04] Alur, Bernadsky, Madhusudan. Optimal Reachability in Weighted Timed Games. ICALP'04 (LNCS 3142).

[AD90] Alur, Dill. Automata for Modeling Real-Time Systems. ICALP'90 (LNCS 443).

[AD94] Alur, Dill. A Theory of Timed Automata. TCS 126(2), 1994.

[dAFH+03] de Alfaro, Faella, Henzinger, Majumdar, Stoelinga. The Element of Surpris in Timed Games. CONCUR'03 (LNCS 2761).

[AM99] Asarin, Maler. As Soon as Possible: Time Optimal Control for Timed Automata. HSCC'99 (LNCS 1569).

[AM04] Alur, Madhusudan. Decision Problems for Timed Automata. SFM-04:RT (LNCS 3142).

[AMP94] Asarin, Maler, Pnueli. Symbolic Controller Synthesis for Discrete and Timed Systems. Hybrid Systems'94 (LNCS 999).

[AMPS98] Asarin, Maler, Pnueli, Sifakis. Controller Synthesis fot Timed Automata. Symp. System Structure and Control'98.

[BCFL04] Bouyer, Cassez, Fleury, Larsen. Optimal Strategies in Priced Timed Game Automata. FSTTCS'04 (LNCS 3328).

Bibliography II

[BCFL05] Bouyer, Cassez, Fleury, Larsen. **Synthesis of Optimal Strategies Using HyTech**. GDV'04 (ENTCS 119), 2005.

[BDGP98] Bérard, Diekert, Gastin, Petit. **Characterization of the Expressive Power of Silent Transitions in Timed Automata.** Fundamenta Informaticae 36(2–3), 1998.

[BDMP03] Bouyer, D'Souza, Madhusudan, Petit. **Timed Control with Partial Observability**. CAV'03 (LNCS 2725).

[Bouyer04] Bouyer. **Forward analysis of updatable timed automata**. Formal Methods in System Design 24(3),2004.

[CHR02] Cassez, Henzinger, Raskin. **A Comparison of Control Problems for Timed and Hybrid Systems**. HSCC'02 (LNCS 2289).

[DDMR04] De Wulf, Doyen, Markey, Raskin. **Robustness and Implementability of Timed Automata**. FORMATS+FTRTFT'04 (LNCS 3253).

[DDR04] De Wulf, Doyen, Raskin. **Almost ASAP Semantics: From Timed Models to Timed Implementations**. HSCC'04 (LNCS 2993).

[Dill89] Dill. **Timing Assumptions and Verification of Finite-State Concurrent Systems.** Aut. Verif. Methods for Fin. State Sys. (LNCS 1989).

[DM02] D'Souza, Madhusudan. **Timed Control Synthesis for External Specifications**. STACS'02 (LNCS 2285).

Bibliography III

[FLM02a] Faella, La Torre, Murano. **Dense Real-Time Games**. LICS'02.

[FLM02b] Faella, La Torre, Murano. **Automata-Theoretic Decision of Timed Games**. VMCAI'02 (LNCS 2294).

[HK99] Henzinger, Kopke. **Discrete-Time Control for Rectangular Hybrid Automata**. TCS 221, 1999.

[LMM02] La Torre, Mukhopadhyay, Murano. **Optimal-Reachability and Control for Acyclic Weighted Timed Automata**. TCS'02.

[Minsky67] Minsky. **Computation: Finite and Infinite Machines.** 1967.

[MPS95] Maler, Pnueli, Sifakis. **On the Synthesis of Discrete Controllers for Timed Systems**. STACS'95 (LNCS 900).

[WTH91] Wong-Toi, Hoffmann. **The Control of Dense Real-Time Discrete Event Systems**. CDC'91.

Acknowledgments: ACI Cortos, Franck Cassez, Fabrice Chevalier, and Nicolas Markey