Partial observation of timed systems

Patricia Bouyer

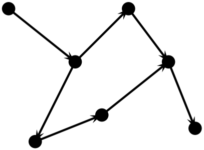
LSV – CNRS & ENS de Cachan – France

GDV'05 – July 2005

Model-checking

Does the system satisfy the property?

Modelling

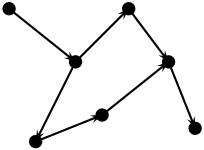


φ

Model-checking

Does the system satisfy the property?

Modelling



$\models$

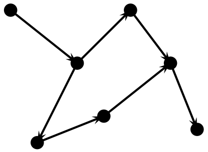
φ

Model-checking Algorithm

Controller synthesis

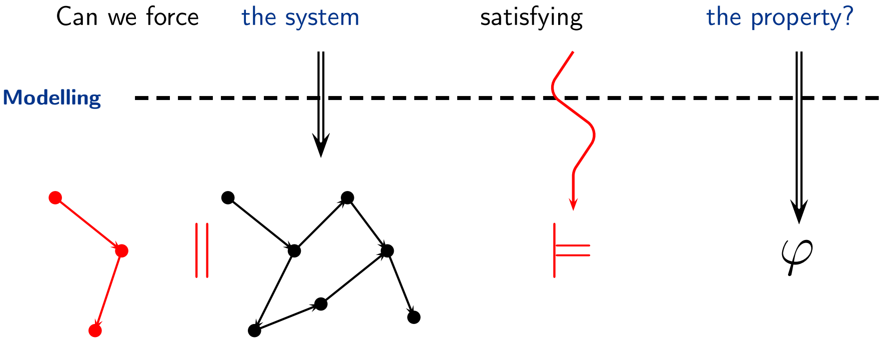
Can we force the system satisfying the property?

Modelling



φ

Controller synthesis



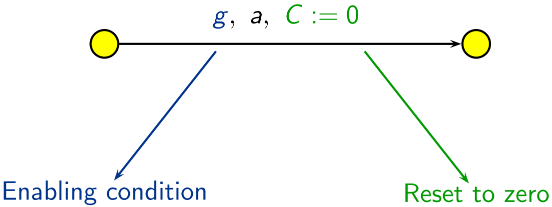
Controller synthesis

Outline

1. Introduction
2. Control synthesis games
3. Control under partial observation
4. Fault diagnosis
5. Conclusion and further developments

Timed automata [Alur & Dill 90's]

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



- An enabling condition (or guard) is:

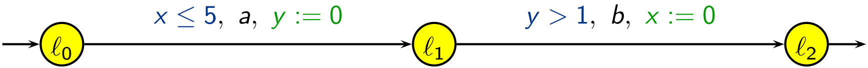
$$g \quad ::= \quad x \sim c \quad | \quad g \wedge g$$

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

- An invariant in each location

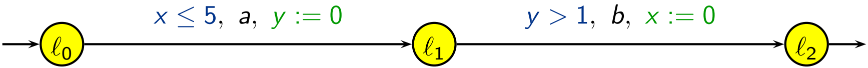
Timed automata (example)

x, y : clocks



Timed automata (example)

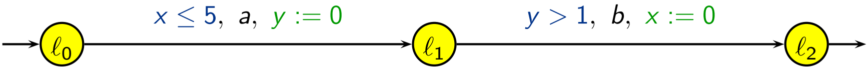
x, y : clocks



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

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 automata (example)

x, y : clocks

$$\to \ell_0 \xrightarrow{\quad x \leq 5,\ a,\ y := 0 \quad} \ell_1 \xrightarrow{\quad y > 1,\ b,\ x := 0 \quad} \ell_2 \to$$

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

(clock) valuation

➜ timed word $(a, 4.1)(b, 5.5)$

Fundamental result [Alur & Dill 90's]

Theorem

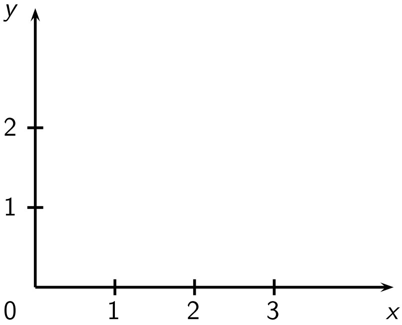
Emptiness of timed automata is decidable and PSPACE-complete.

Fundamental result [Alur & Dill 90's]

Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

The region abstraction



Equivalence of finite index

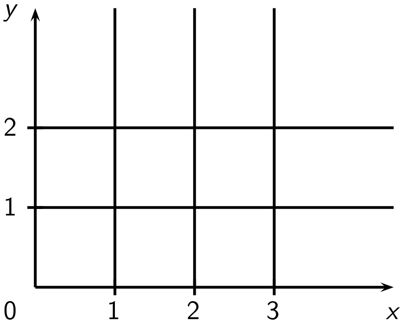
Fundamental result

[Alur & Dill 90's]

Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

The region abstraction



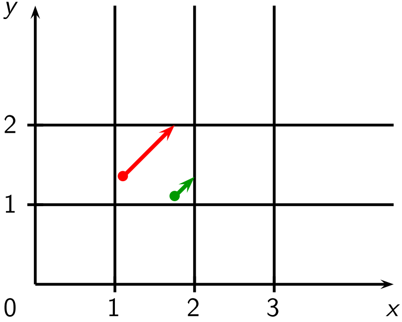
Equivalence of finite index

Fundamental result [Alur & Dill 90's]

Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

The region abstraction



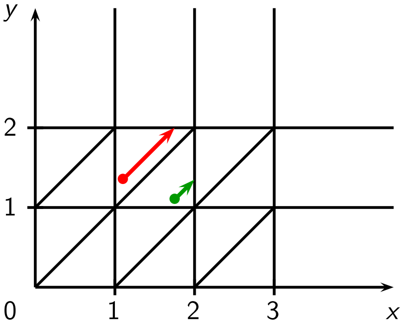
Equivalence of finite index

Fundamental result [Alur & Dill 90's]

Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

The region abstraction



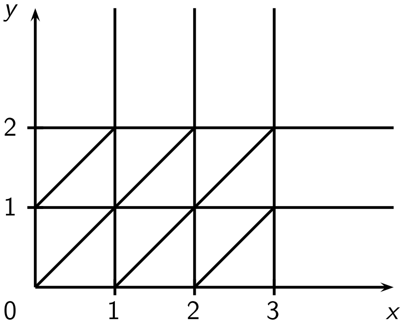
Equivalence of finite index

Fundamental result [Alur & Dill 90's]

Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

The region abstraction



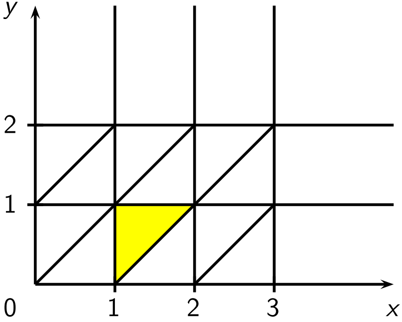
Equivalence of finite index

Fundamental result **[Alur & Dill 90's]**

Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

The region abstraction



Equivalence of finite index

region defined by

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

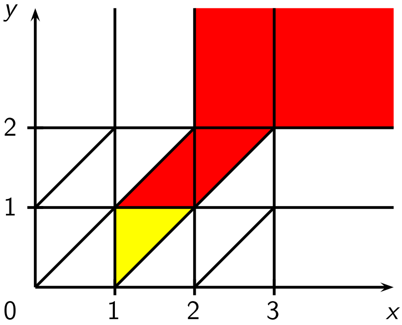
$\{x\} < \{y\}$

Fundamental result [Alur & Dill 90's]

Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

The region abstraction



Equivalence of finite index

region defined by

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

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

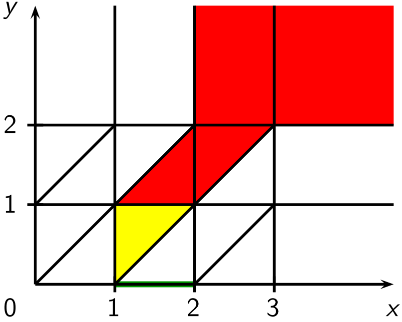
delay successors

Fundamental result [Alur & Dill 90's]

Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

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

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]
- ...

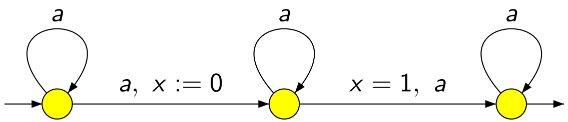
A model not far from undecidability

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:

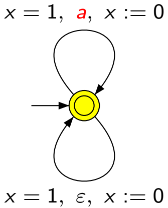


Power of ε-transitions

[Bérard, Diekert, Gastin, Petit 1998]

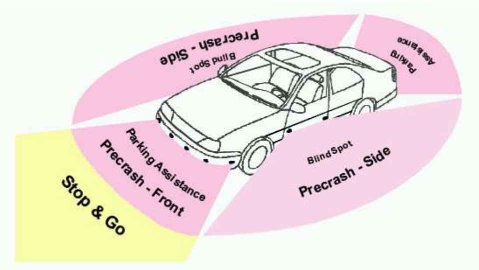
Proposition

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



Outline

An example, the car periphery supervision



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

Control synthesis 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

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

  - 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 runs 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 runs in parallel with $\mathcal{P}$

It should not be too powerful!

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:

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

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

It should not be too powerful!

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

Strategies and controllers

- A **strategy** is a partial function

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

  - needs to satisfy some *continuity* property:

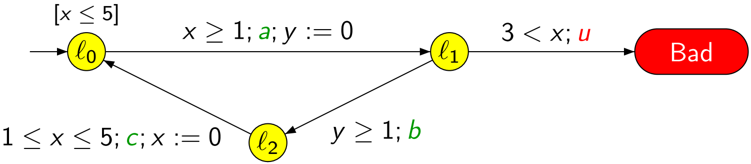
$$f(\rho) = \lambda \implies \exists t > 0,\ \forall 0 \leq t' < t,\ f(\rho \xrightarrow{\delta(t')}) = \lambda$$

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

It should not be too powerful!

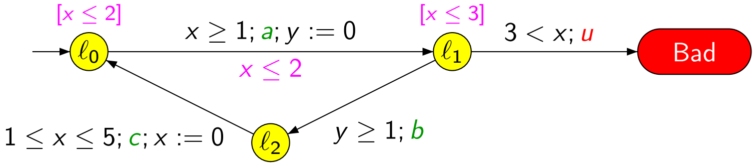
  - needs to be *non-restricting* for uncontrollable actions
  - needs to be *non-blocking*: if there is no deadlock in the original plant, there will be no deadlock in the controlled system

An example



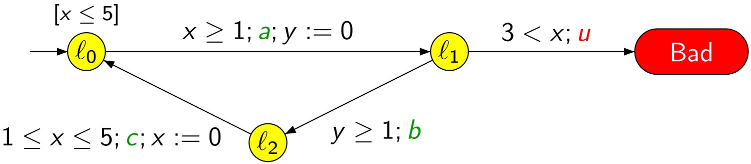
Aim: control the system in such a way that Bad state is avoided.

An example



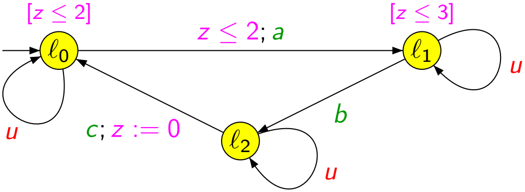
Aim: control the system in such a way that Bad state is avoided.

An example

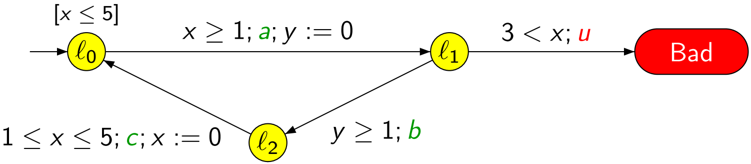


Aim: control the system in such a way that Bad state is avoided.

A controller:

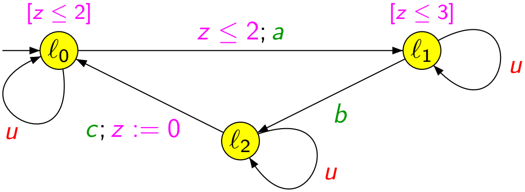


An example



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

A controller:



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

Decidability and complexity

- The **attractor** of a zone-definable set is computable.
- Winning states of safety and reachability games are computable.
- Winning strategies can be computed and are polyhedral.
- Winning strategies can be state-based.

Theorem [Henzinger, Kopke 1999]

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

Computing winning states

- 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

- 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 (Pred_δ):

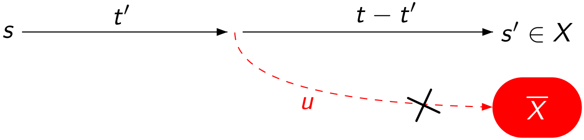
$$s \xrightarrow{\quad\quad t \quad\quad} s' \in X$$

Computing winning states

- 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 (Pred_δ):

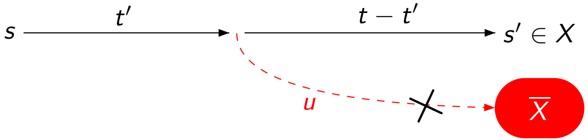


Computing winning states

- 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 (Pred_δ):



- **winning states:** greatest fixed point of

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

Further objectives

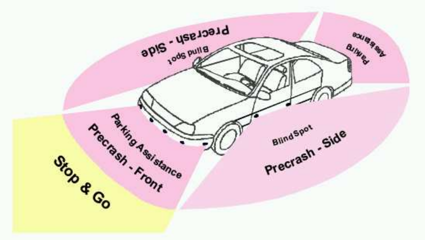
- TCTL objectives **[Faella, La Torre, Murano 2002]**
- CTL, LTL objectives **[Faella, La Torre, Murano 2002]**
- general symmetric parity games **[de Alfaro, Faella, Henzinger, Majumdar, Stoelinga 2003]**
- external specifications given by timed automata **[D'Souza, Madhusudan 2002]**

➜ use theory of classical untimed games

Outline

Why partial observation?

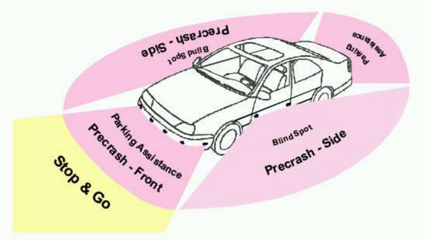
Example (The car periphery supervision)



Environment is seen through **sensors**.

Why partial observation?

Example (The car periphery supervision)

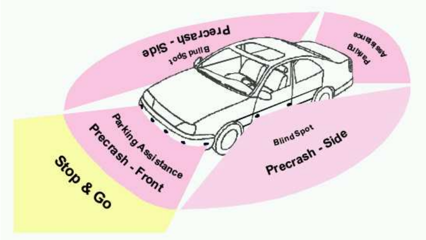


Environment is seen through **sensors**.

- some actions are non-controllable

Why partial observation?

Example (The car periphery supervision)



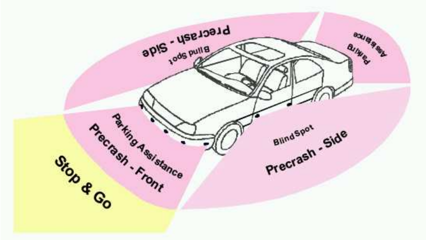
Environment is seen through **sensors**.

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

[Partial observation]

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 observation]

Stumbling blocks:

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

Control under partial observation

Theorem [Bouyer, D'Souza, Madhusudan, Petit 2003]

Safety and reachability control under partial observation is undecidable.

Control under partial observation

Theorem **[Bouyer, D'Souza, Madhusudan, Petit 2003]**

Safety and reachability control under partial observation is undecidable.

➜ by reduction of universality problem for timed automata

Control under partial observation

Theorem **[Bouyer, D'Souza, Madhusudan, Petit 2003]**

Safety and 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.

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

Control under partial observation

Theorem **[Bouyer, D'Souza, Madhusudan, Petit 2003]**

Safety and 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
- $(\delta_0, t_0)(a_0, t'_0)(\delta_1, t_1)(a_1, t'_1)\ldots$ is accepted by $\mathcal{P}$ iff $t_i = t'_i$ for every i and $(a_0, t_0)(a_1, t_1)\ldots$ is accepted by $\mathcal{A}$ along the path $\delta_0\delta_1\ldots$

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.

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

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 $\mathcal{A}$

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 $\mathcal{A}$

NB: this undecidability result seems robust...

Fixing resources

[Bouyer, D'Souza, Madhusudan, Petit 2003]

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

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

Fixing resources

[Bouyer, D'Souza, Madhusudan, Petit 2003]

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

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

With fixed resources, control of simple winning objectives becomes decidable (and 2EXPTIME-complete).

Fixing resources

[Bouyer, D'Souza, Madhusudan, Petit 2003]

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

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

With fixed resources, control of simple winning objectives becomes decidable (and 2EXPTIME-complete).

Control under partial observation is a difficult problem

Fixing resources

[Bouyer, D'Souza, Madhusudan, Petit 2003]

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

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

With fixed resources, control of simple winning objectives becomes decidable (and 2EXPTIME-complete).

Control under partial observation
is a difficult problem

➜ We focus on a simpler problem, where partial observation is crucial

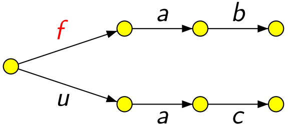
Outline

Principle of fault diagnosis

[Sampath, Sengupta, Lafortune, Sinnamohideen, Teneketzis 1995]

Principle: "observe the behavior of a plant, and tell if something wrong has happened"

System:

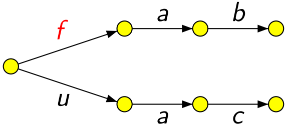


Principle of fault diagnosis

[Sampath, Sengupta, Lafortune, Sinnamohideen, Teneketzis 1995]

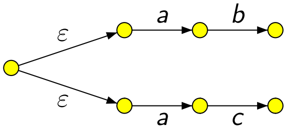
Principle: "observe the behavior of a plant, and tell if something wrong has happened"

System:



$\Sigma_o = \{a, b, c\} \quad \Sigma_u = \{f, u\}$

Sensors:

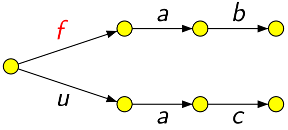


Principle of fault diagnosis

[Sampath, Sengupta, Lafortune, Sinnamohideen, Teneketzis 1995]

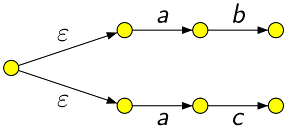
Principle: **"observe the behavior of a plant, and tell if something wrong has happened"**

System:



$\Sigma_o = \{a, b, c\}$ $\Sigma_u = \{f, u\}$

Sensors:



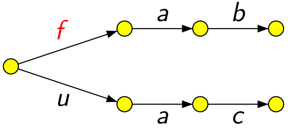
Observation: «ab» or «ac»

Principle of fault diagnosis

[Sampath, Sengupta, Lafortune, Sinnamohideen, Teneketzis 1995]

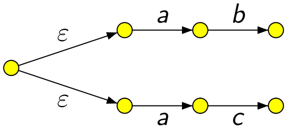
Principle: "observe the behavior of a plant, and tell if something wrong has happened"

System:



$\Sigma_o = \{a, b, c\}$ $\Sigma_u = \{f, u\}$

Sensors:



Observation: «ab» or «ac»

Did a fault occur?

The timed framework

- Plant = timed automaton
- Σ_o observable events, and Σ_u unobservable events

The timed framework

- Plant = timed automaton
- Σ_o observable events, and Σ_u unobservable events

Pb: Given an observation (timed word over Σ_o), did a fault occur?

The timed framework

- Plant = timed automaton
- Σ_o observable events, and Σ_u unobservable events

Pb: Given an observation (timed word over Σ_o), did a fault occur?
Aim: answer within Δ units of time

The timed framework

- Plant = timed automaton
- Σ_o observable events, and Σ_u unobservable events

Pb: Given an observation (timed word over Σ_o), did a fault occur?
Aim: answer within Δ units of time

Example: $\Sigma_o = \{a, b\}$ $\Sigma_u = \{f\}$

- Execution of the plant: $w = (a, 1)(f, 3.1)(b, 4.5)$
- Observation: $\pi(w) = (a, 1)(b, 4.5)$

The timed framework

- Plant = timed automaton
- Σ_o observable events, and Σ_u unobservable events

Pb: Given an observation (timed word over Σ_o), did a fault occur?
Aim: answer within Δ units of time

Example: $\Sigma_o = \{a, b\}$ $\Sigma_u = \{f\}$

- Execution of the plant: $w = (a, 1)(f, 3.1)(b, 4.5)$
- Observation: $\pi(w) = (a, 1)(b, 4.5)$

1-diagnoser:	has to announce fault on $\pi(w)$
2-diagnoser:	can announce fault on $\pi(w)$
	may announce nothing on $\pi(w)$

Δ-diagnosis

A Δ-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \rightarrow \{0, 1\}$ such that:

Δ-diagnosis

A Δ-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \rightarrow \{0,1\}$ such that:

- for every non-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 0$

Δ-diagnosis

A Δ-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \rightarrow \{0, 1\}$ such that:

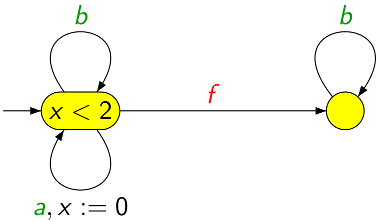
- for every non-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 0$
- for every Δ-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 1$

Δ-diagnosis

A Δ-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \to \{0,1\}$ such that:

- for every non-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 0$
- for every Δ-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 1$

Example



This system is 2-diagnosable... but not 1-diagnosable because $(f,0)(b,1)$ and $(b,1)$ raise the same observation.

Δ-diagnosis

A Δ-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \rightarrow \{0,1\}$ such that:

- for every non-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 0$
- for every Δ-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 1$

A solution [Tripakis02]: state estimation

➜ the Δ-diagnosis problem is PSPACE-complete

Δ-diagnosis

A Δ-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \rightarrow \{0, 1\}$ such that:

- for every non-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 0$
- for every Δ-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 1$

A solution [Tripakis02]: state estimation
➜ the Δ-diagnosis problem is PSPACE-complete

Limit of this approach:

- expensive (in theory) if we want to run it online
- not close enough to controller synthesis

Δ-diagnosis

A Δ-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \to \{0, 1\}$ such that:

- for every non-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 0$
- for every Δ-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 1$

A solution [Tripakis02]: state estimation

➜ the Δ-diagnosis problem is PSPACE-complete

Limit of this approach:

- expensive (in theory) if we want to run it online
- not close enough to controller synthesis

➜ **Our aim:** build a deterministic diagnoser $\mathcal{O}$...

Δ-diagnosis

A Δ-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \rightarrow \{0,1\}$ such that:

- for every non-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 0$
- for every Δ-faulty execution ρ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 1$

A solution [Tripakis02]: state estimation

➜ the Δ-diagnosis problem is PSPACE-complete

Limit of this approach:

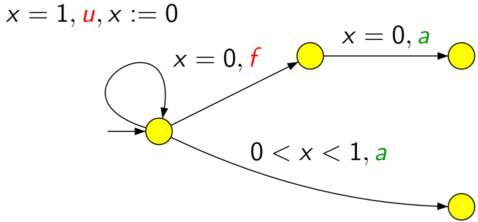
- expensive (in theory) if we want to run it online
- not close enough to controller synthesis

➜ **Our aim:** build a deterministic diagnoser $\mathcal{O}$...

$$L_{\Delta f}(\mathcal{P}) \subseteq L(\mathcal{O}) \subseteq L_{\neg f}(\mathcal{P})^c$$

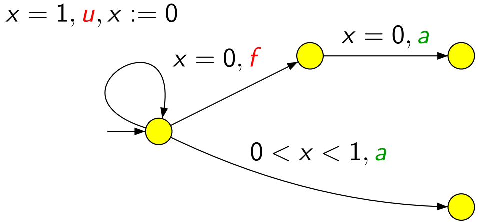
Diagnosis with deterministic timed automata

- less general than previous diagnosis



Diagnosis with deterministic timed automata

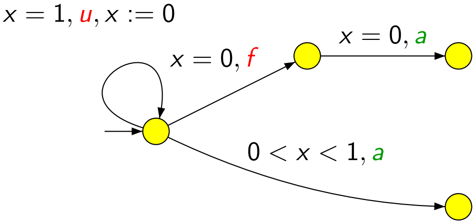
- less general than previous diagnosis



- the diagnosis problem with deterministic timed automata (DTA) is not solved yet

Diagnosis with deterministic timed automata

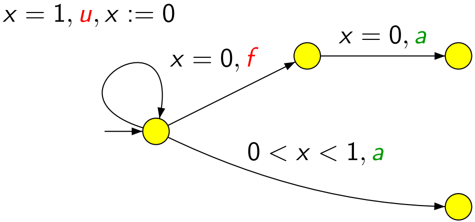
- less general than previous diagnosis



- the diagnosis problem with deterministic timed automata (DTA) is not solved yet
- the "precise" diagnosis problem and the "asap" diagnosis problem with DTA are undecidable **[Chevalier 2004]**

Diagnosis with deterministic timed automata

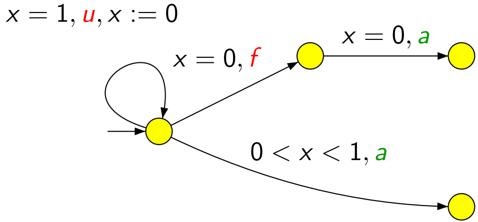
- less general than previous diagnosis



- the diagnosis problem with deterministic timed automata (DTA) is not solved yet
- the "precise" diagnosis problem and the "asap" diagnosis problem with DTA are undecidable **[Chevalier 2004]**
- restriction to bounded resources $\mu = (X, m, \max)$

Diagnosis with deterministic timed automata

- less general than previous diagnosis



- the diagnosis problem with deterministic timed automata (DTA) is not solved yet
- the "precise" diagnosis problem and the "asap" diagnosis problem with DTA are undecidable **[Chevalier 2004]**
- restriction to bounded resources $\mu = (X, m, \max)$

Theorem **[Bouyer, Chevalier, D'Souza 2005]**

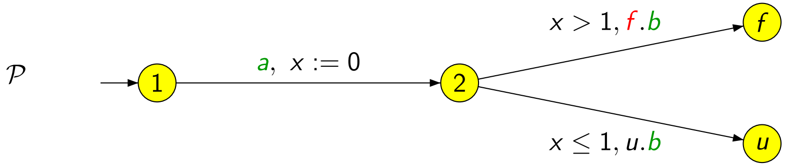
Δ-diagnosis of timed systems with DTA_μ is 2EXPTIME-complete.

Observation as a game

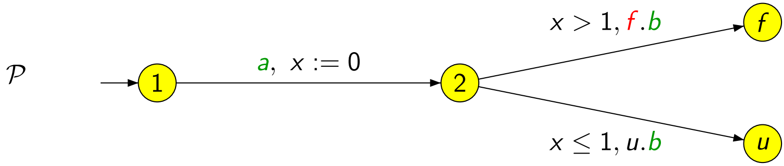
We will transform the diagnosis problem into a two-player safety game:

- one player is the observer $\square$
- the other player is the environment $\bigcirc$

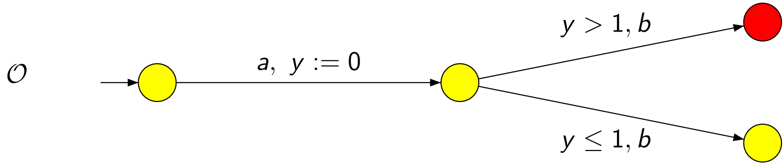
The plant is Δ-DTA$_\mu$-diagnosable iff $\square$ has a winning strategy

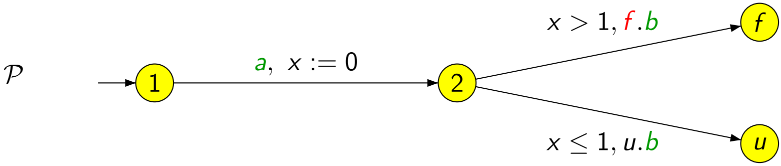


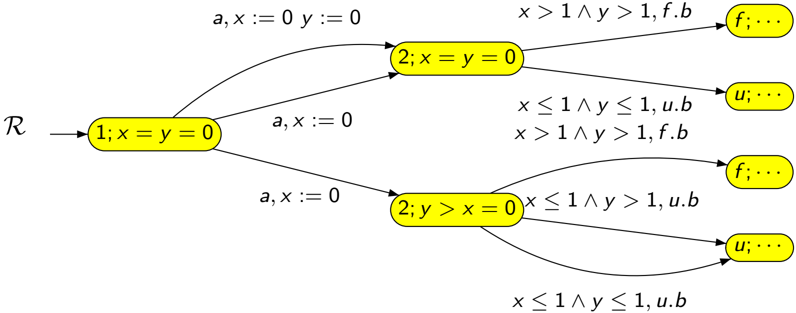
Is there an observer for the plant with one clock and constants 0 and 1?

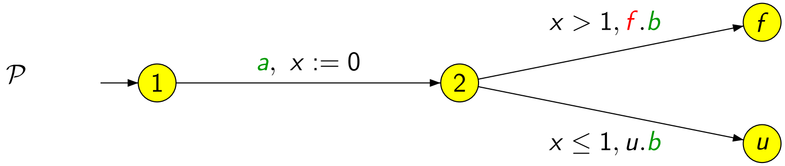


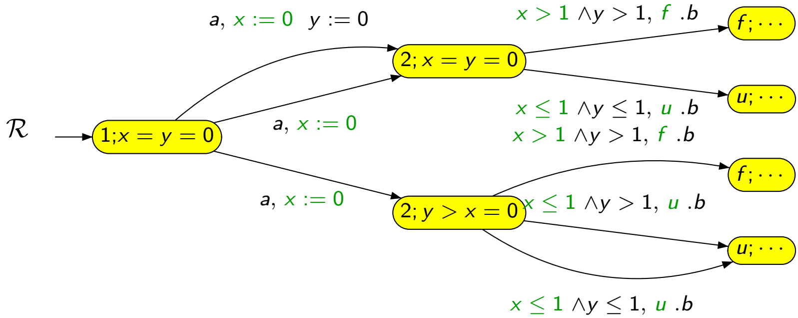
Is there an observer for the plant with one clock and constants 0 and 1?

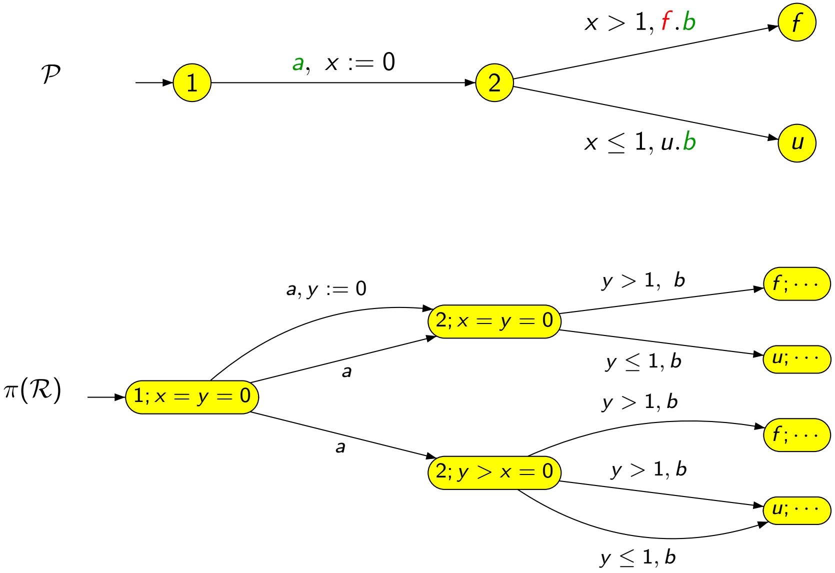


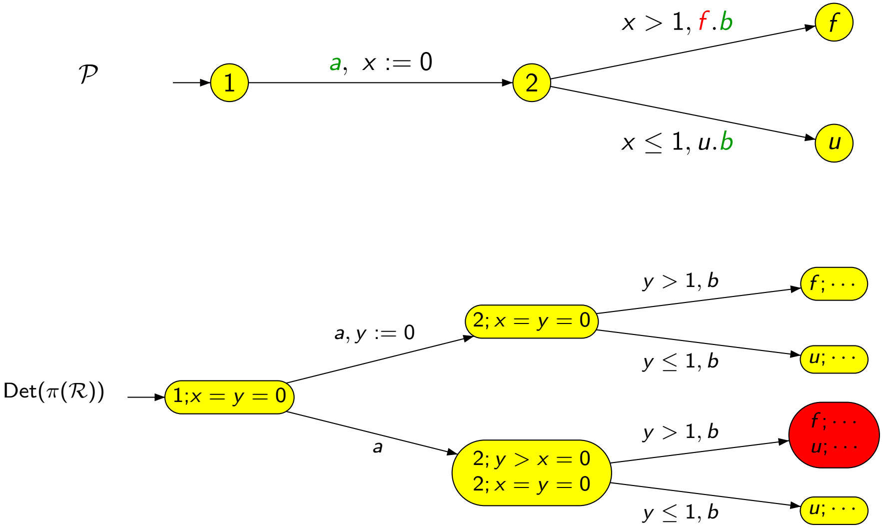

$\mathcal{P}$
1
$a, \ x := 0$
2
$x > 1, f.b$
$x \leq 1, u.b$
f
u

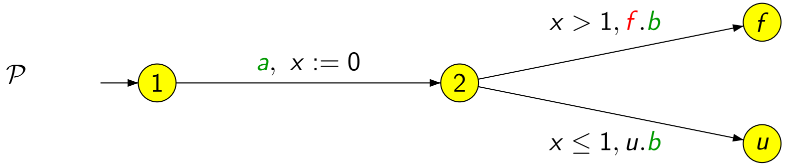

$\mathcal{R}$
$1; x = y = 0$
$a, x := 0 \ y := 0$
$a, x := 0$
$a, x := 0$
$2; x = y = 0$
$2; y > x = 0$
$x > 1 \wedge y > 1, f.b$
$x \leq 1 \wedge y \leq 1, u.b$
$x > 1 \wedge y > 1, f.b$
$x \leq 1 \wedge y > 1, u.b$
$x \leq 1 \wedge y \leq 1, u.b$
$f; \cdots$
$u; \cdots$
$f; \cdots$
$u; \cdots$

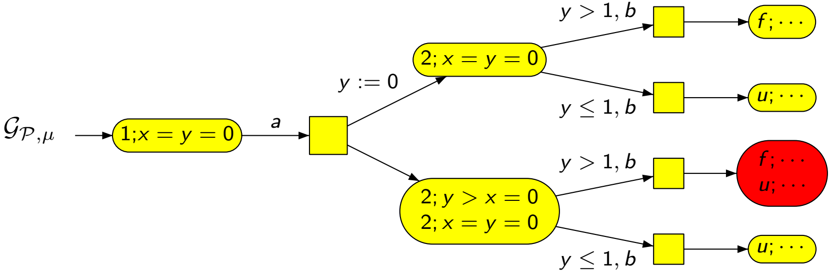

$\mathcal{P}$
1
$a,\ x := 0$
2
$x > 1, f.b$
f
$x \leq 1, u.b$
u


$\mathcal{R}$
$1; x = y = 0$
$a,\ x := 0\ \ y := 0$
$a,\ x := 0$
$a,\ x := 0$
$2; x = y = 0$
$2; y > x = 0$
$x > 1 \wedge y > 1,\ f\ .b$
$f; \cdots$
$x \leq 1 \wedge y \leq 1,\ u\ .b$
$u; \cdots$
$x > 1 \wedge y > 1,\ f\ .b$
$f; \cdots$
$x \leq 1 \wedge y > 1,\ u\ .b$
$u; \cdots$
$x \leq 1 \wedge y \leq 1,\ u\ .b$


$\mathcal{P}$
1
$a,\ x := 0$
2
$x > 1, f.b$
f
$x \leq 1, u.b$
u
$\pi(\mathcal{R})$
$1; x = y = 0$
$a, y := 0$
a
a
$2; x = y = 0$
$2; y > x = 0$
$y > 1,\ b$
$y \leq 1, b$
$y > 1, b$
$y > 1, b$
$y \leq 1, b$
$f; \cdots$
$u; \cdots$
$f; \cdots$
$u; \cdots$


$\mathcal{P}$
1
$a, \ x := 0$
2
$x > 1, f.b$
f
$x \leq 1, u.b$
u
$\mathrm{Det}(\pi(\mathcal{R}))$
$1; x = y = 0$
$a, y := 0$
$2; x = y = 0$
$y > 1, b$
$f; \cdots$
$y \leq 1, b$
$u; \cdots$
a
$2; y > x = 0$
$2; x = y = 0$
$y > 1, b$
$f; \cdots$
$u; \cdots$
$y \leq 1, b$
$u; \cdots$


$\mathcal{P}$
1
$a, \ x := 0$
2
$x > 1, f.b$
$x \leq 1, u.b$
f
u


$\mathcal{G}_{\mathcal{P},\mu}$
1;$x = y = 0$
a
$y := 0$
2; $x = y = 0$
2; $y > x = 0$
2; $x = y = 0$
$y > 1, b$
$y \leq 1, b$
$y > 1, b$
$y \leq 1, b$
$f; \cdots$
$u; \cdots$
$f; \cdots$
$u; \cdots$
$u; \cdots$

Diagnosis by DTA$_\mu$

Proposition

$\square$ has a winning strategy in $\mathcal{G}_{\mathcal{P},\mu}$ iff there is a diagnoser for $\mathcal{P}$ in DTA$_\mu$.

➜ Δ-DTA$_\mu$-diagnosability is in 2EXPTIME

Δ-DTA$_\mu$-observability if 2EXPTIME-hard

$\longrightarrow$ By reduction of the acceptance of
an Alternating Turing Machine using exponential space

Δ-DTA$_\mu$-observability if 2EXPTIME-hard

$\rightarrow$ By reduction of the acceptance of
an Alternating Turing Machine using exponential space

- The plant plays "a"'s.
- The diagnoser reads these "a"'s and plays a sequence of configurations.
- The plant verifies that this sequence is correct.

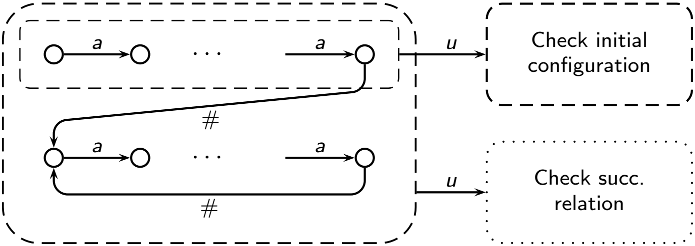
Δ-DTA$_\mu$-observability if 2EXPTIME-hard

$\longrightarrow$ By reduction of the acceptance of
an Alternating Turing Machine using exponential space

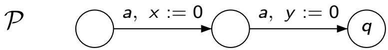
- The plant plays "a"'s.
- The diagnoser reads these "a"'s and plays a sequence of configurations.
- The plant verifies that this sequence is correct.

NB: the plant non-deterministically chooses one test

Shape of the plant



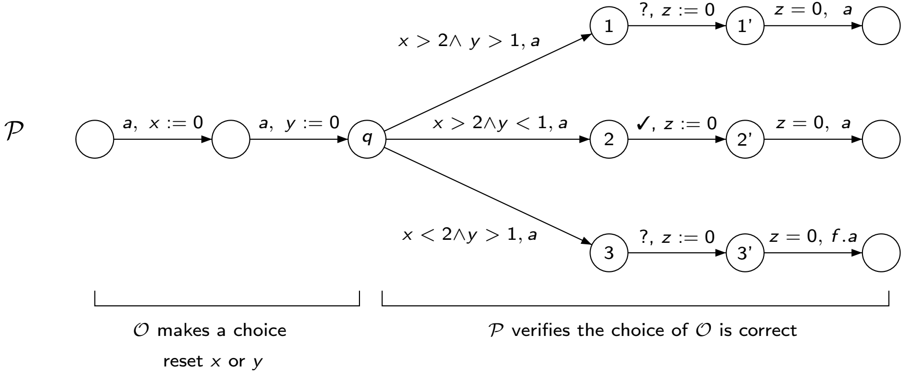
$\mathcal{O}$ has 1 clock.



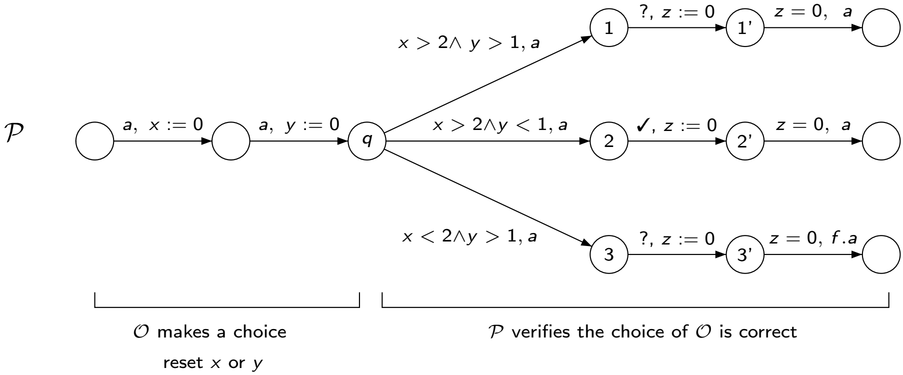
$\mathcal{O}$ makes a choice

reset x or y

$\mathcal{O}$ has 1 clock.

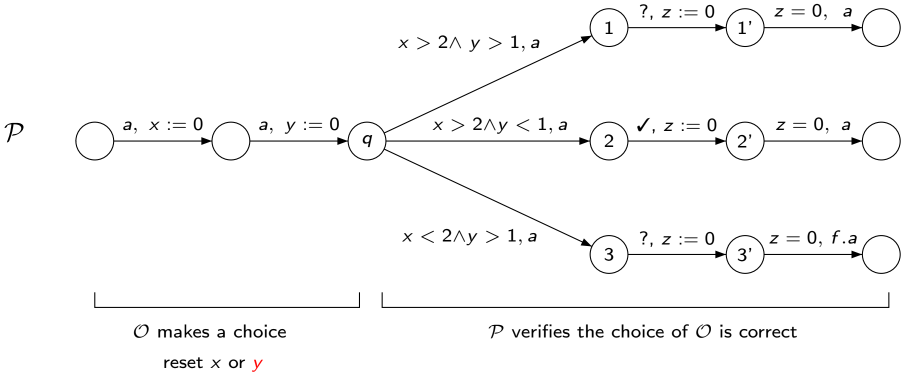


$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

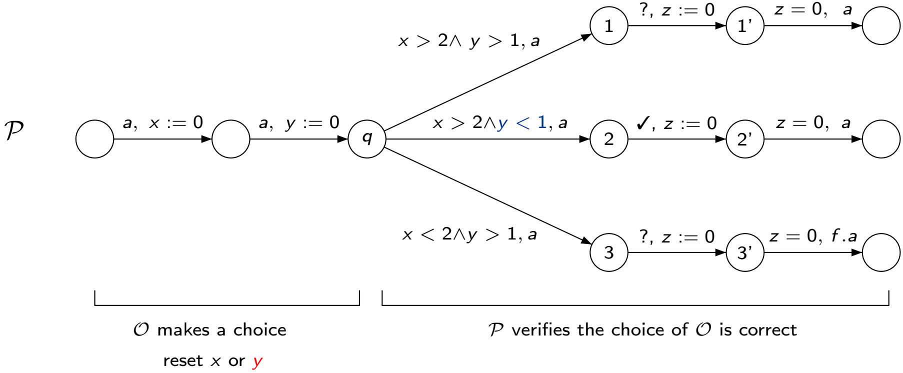
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible

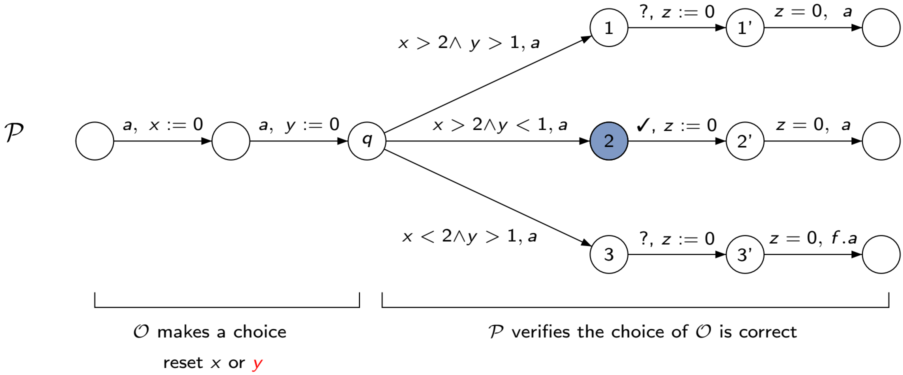
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible

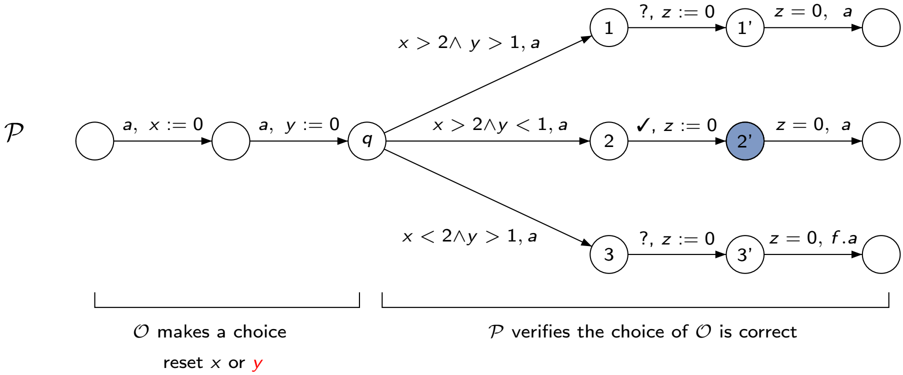
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible

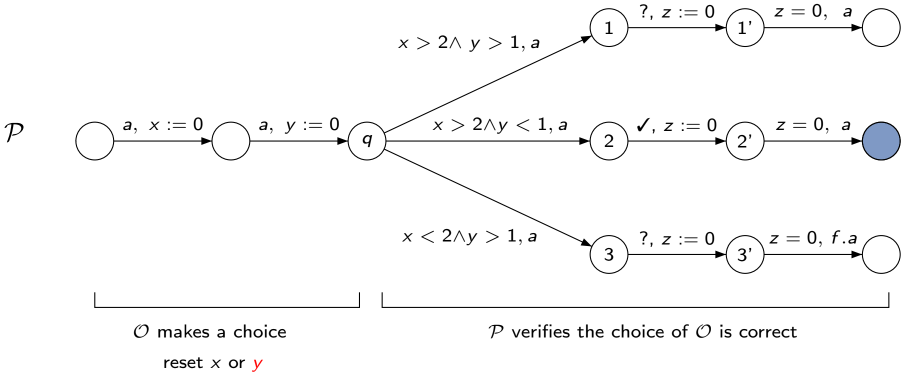
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible

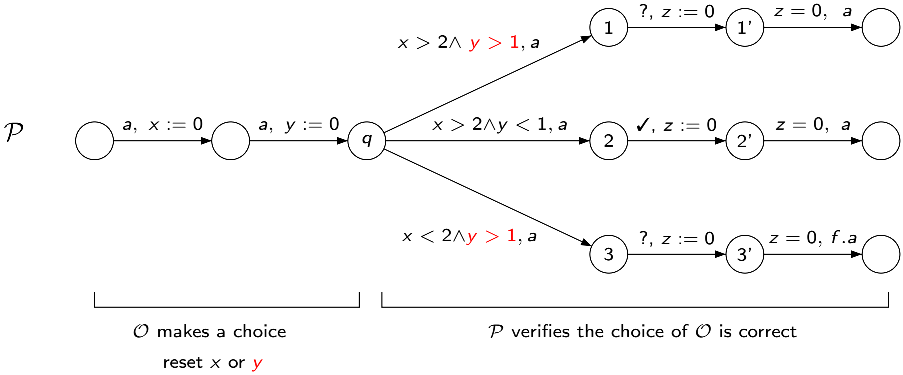
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible

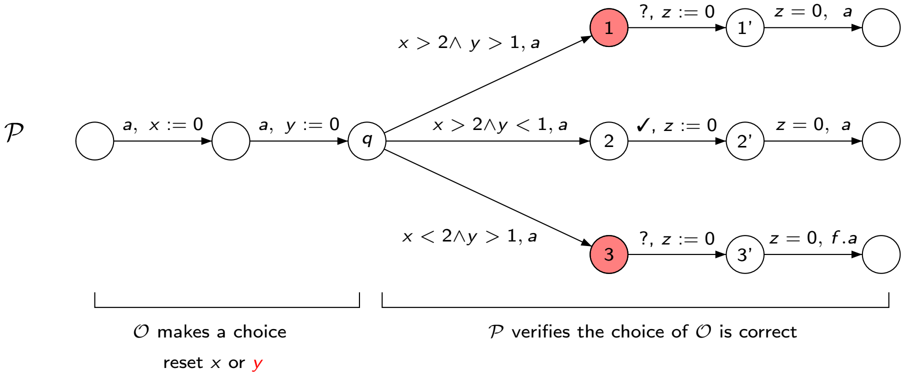
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible

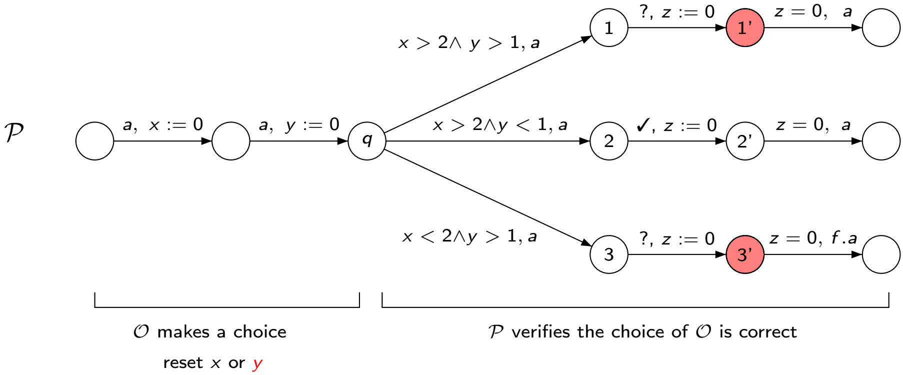
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible

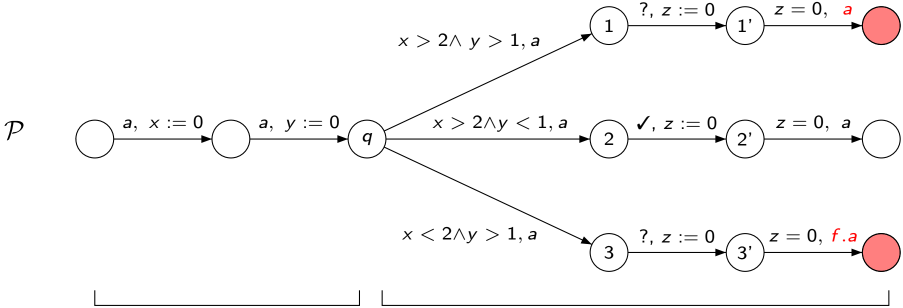
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible

$\mathcal{O}$ has 1 clock.



$\mathcal{O}$ makes a choice

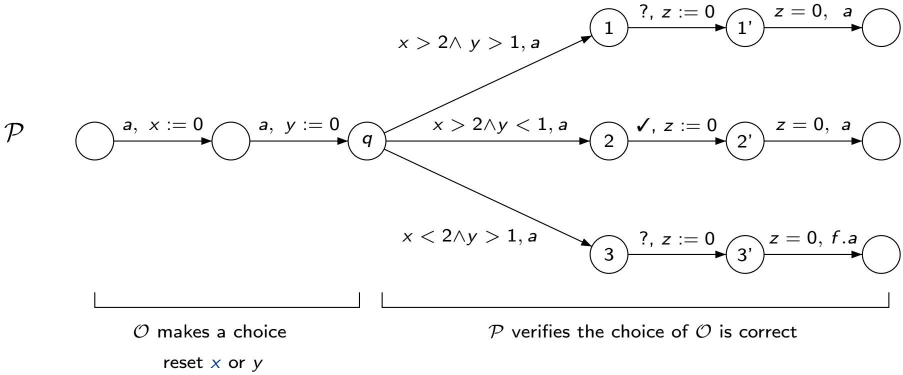
reset x or y

$\mathcal{P}$ verifies the choice of $\mathcal{O}$ is correct

$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible

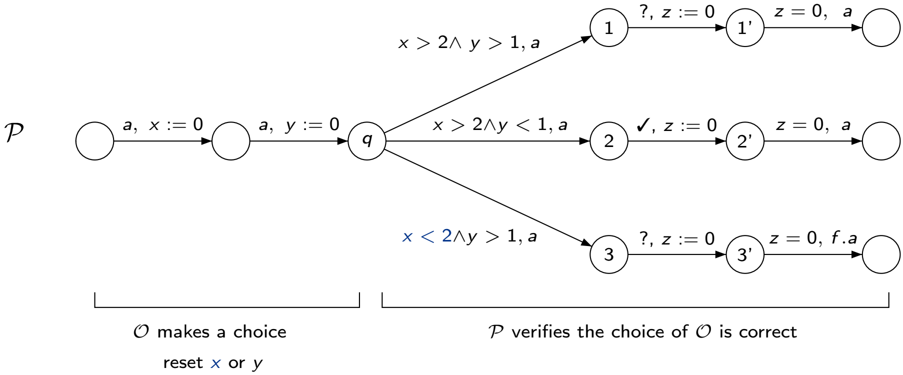
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible
- if $\mathcal{O}$ "remembers" x, diagnosis is possible

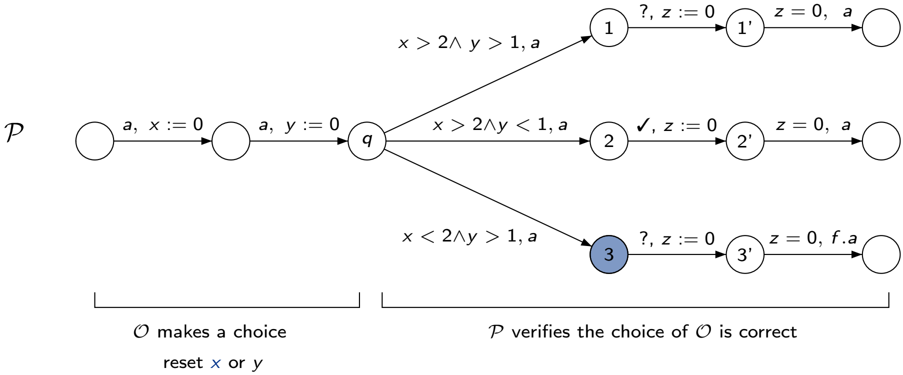
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible
- if $\mathcal{O}$ "remembers" x, diagnosis is possible

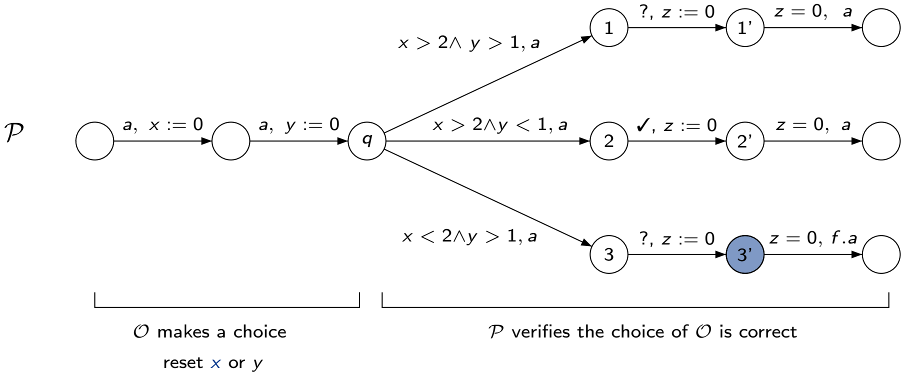
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible
- if $\mathcal{O}$ "remembers" x, diagnosis is possible

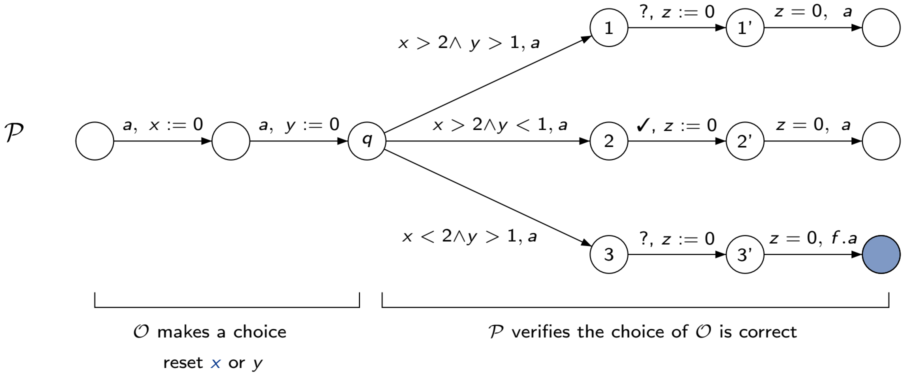
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible
- if $\mathcal{O}$ "remembers" x, diagnosis is possible

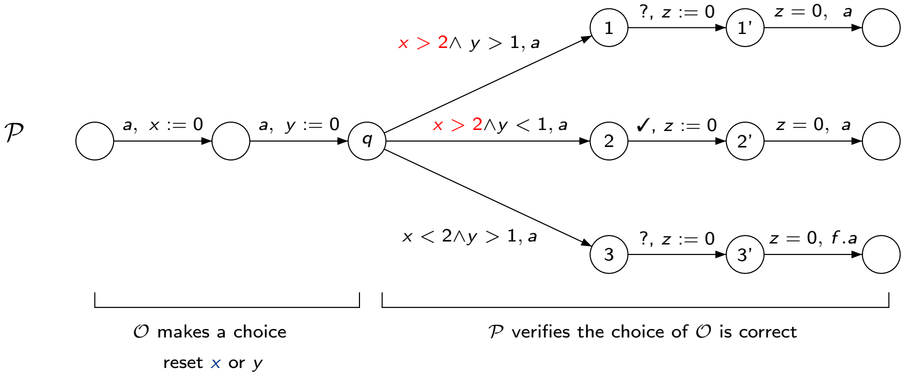
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible
- if $\mathcal{O}$ "remembers" x, diagnosis is possible

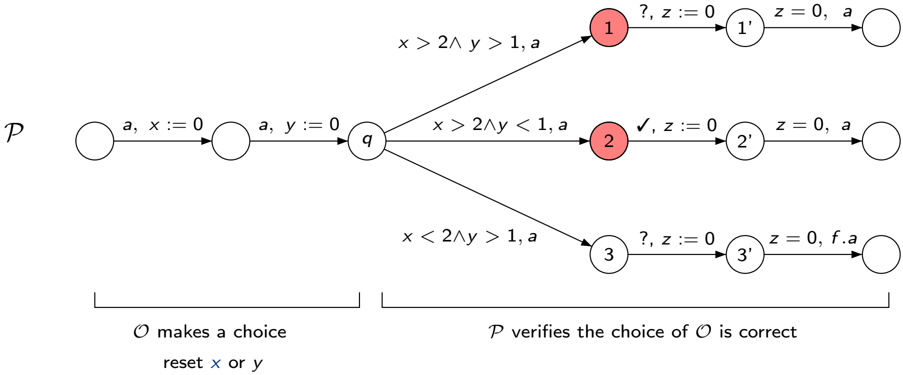
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible
- if $\mathcal{O}$ "remembers" x, diagnosis is possible

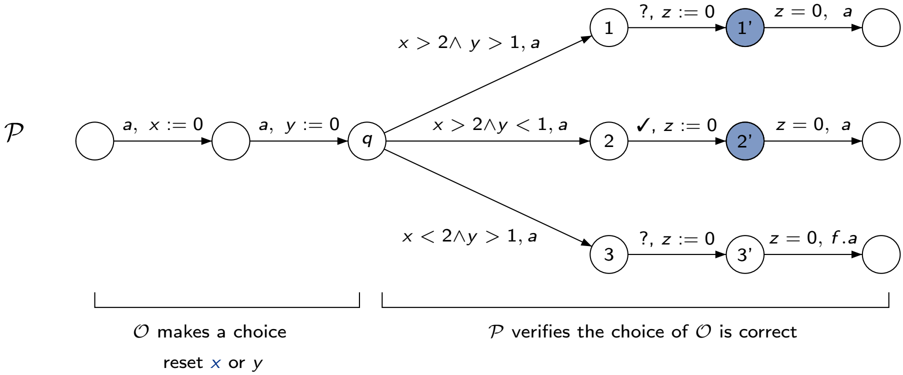
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible
- if $\mathcal{O}$ "remembers" x, diagnosis is possible

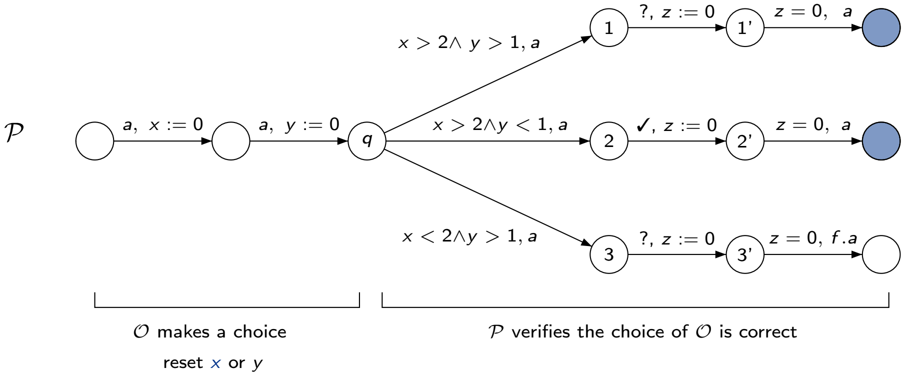
$\mathcal{O}$ has 1 clock.



$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible
- if $\mathcal{O}$ "remembers" x, diagnosis is possible

$\mathcal{O}$ has 1 clock.

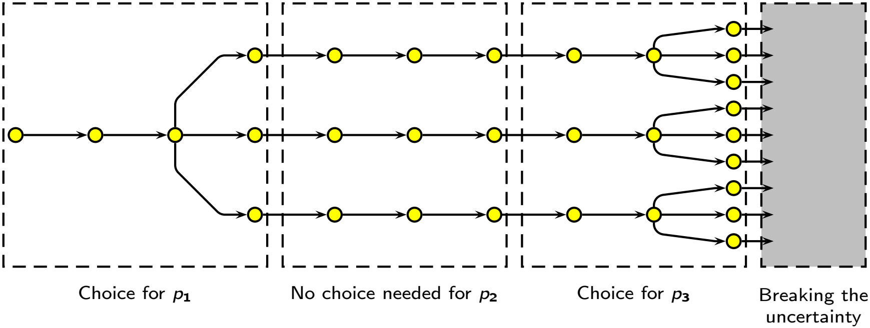


$\mathcal{P}$ can force $\mathcal{O}$ "remember" x:

- if $\mathcal{O}$ "remembers" y, diagnosis is impossible
- if $\mathcal{O}$ "remembers" x, diagnosis is possible

An example of encoding for a 3SAT formula

Formula $p_1 \vee \neg p_3$**:**



Diagnosis by event-recording timed automata

- one clock x_a per event a
- clock x_a is reset when a occurs

Diagnosis by event-recording timed automata

- one clock x_a per event a
- clock x_a is reset when a occurs

Property

- Event-recording timed automata are determinizable
  [Alur, Fix, Henzinger 1994]
- Event-recording timed automata are *input-determined*
  [D'Souza, Tabareau 2004]

Diagnosis by event-recording timed automata

- one clock x_a per event a
- clock x_a is reset when a occurs

Property

- Event-recording timed automata are determinizable
  [Alur, Fix, Henzinger 1994]
- Event-recording timed automata are *input-determined*
  [D'Souza, Tabareau 2004]

➜ Diagnosis (with bounded resources) becomes PSPACE-complete
[BCD05]

Outline

1. Introduction
2. Control synthesis games
3. Control under partial observation
4. Fault diagnosis
5. **Conclusion and further developments**

Conclusion & further developments

Conclusion

- Partial observation adds complexity to control problems
- Even fault diagnosis is difficult
- **Related domains:** conformance testing, monitoring. . .

Conclusion & further developments

Conclusion

- Partial observation adds complexity to control problems
- Even fault diagnosis is difficult
- **Related domains:** conformance testing, monitoring. . .

Further developments

- Algorithms for control under partial observation
  e.g. forward zone-based algorithm (*cf* Emmanuel's talk)
- Fault diagnosis with DTA/ERA
- Get rid of some resources or the Δ parameter
- Control under partial observation for other classes of systems
  (*e.g.* o-minimal hybrid games)

Bibliography I

[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 Surprise in Timed Games**. CONCUR'03 (LNCS 2761).

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

[BCD05] Bouyer, Chevalier, D'Souza. **Fault Diagnosis using Timed Automata**. FoSSaCS'05 (LNCS 3441).

[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).

[Che04] Chevalier. **Détection d'erreurs dans les systèmes temporisés**. Master Thesis, 2004.

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

Bibliography II

[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.

[SSL+95] Sampath, Sengupta, Lafortune, Sinnamohideen, Teneketzis. **Diagnosability of Discrete Event Systems**. IEEE Transactions Automatic Control 40(9), 1995.

[SSL+96] Sampath, Sengupta, Lafortune, Sinnamohideen, Teneketzis. **Failure Diagnosis using Discrete Event Systems**. IEEE Transactions Control Systems Technology 4(2), 1996.

[Tri02] Tripakis. **Fault Diagnosis for Timed Automata**. FTRTFT'02 (LNCS 2469).

[Tri03] Tripakis. **Folk Theorems on the Determinization and Minimization of Timed Automata**. FORMATS'03 (LNCS 2791).

Acknowledgments: Fabrice Chevalier for providing some of the slides