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?
Model-checking

Does the system satisfy the property?
Controller synthesis

Can we force the system satisfying the property?

Modelling
Controller synthesis

Can we force the system satisfying the property?

Modelling

Controller synthesis
Outline

1. Introduction

2. Control synthesis games

3. Control under partial observation

4. Fault diagnosis

5. Conclusion and further developments
Timed automata

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

  \[ g, \ a, \ C := 0 \]

- An enabling condition (or **guard**) is:

  \[ g \ ::= \ x \sim c \mid g \land g \]

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

- An invariant in each location
Timed automata (example)

$x, y : \text{clocks}$

$\ell_0 \xrightarrow{x \leq 5, \ a, \ y := 0} \ell_1 \xrightarrow{y > 1, \ b, \ x := 0} \ell_2$
Timed automata (example)

$x, y : \text{clocks}$

$x \leq 5, \ a, \ y := 0$

$y > 1, \ b, \ x := 0$
Timed automata (example)

$x, y : $ clocks

$x \leq 5, a, y := 0$

$y > 1, b, x := 0$

(clock) valuation
Timed automata (example)

$x, y : \text{clocks}$

$x \leq 5, a, y := 0$

$y > 1, b, x := 0$

$\ell_0 \xrightarrow{\delta(4.1)} \ell_0 \xrightarrow{a} \ell_1 \xrightarrow{\delta(1.4)} \ell_1 \xrightarrow{b} \ell_2$

$(\text{clock}) \text{ valuation}$

$\rightarrow \text{ timed word } (a, 4.1)(b, 5.5)$
**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

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
Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

The region abstraction

Equivalence of finite index
Fundamental result

Theorem

Emptiness of timed automata is decidable and PSPACE-complete.

The region abstraction

Equivalence of finite index

region defined by

\[ l_x = ]1; 2[, l_y = ]0; 1[ \]

\{x\} < \{y\}
**Fundamental result**

[Alur & Dill 90’s]

The emptiness of timed automata is decidable and PSPACE-complete.

**The region abstraction**

The region defined by

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

\{x\} < \{y\}

Delay successors
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 $\varepsilon$-transitions

[Bérard, Diekert, Gastin, Petit 1998]

**Proposition**

- $\varepsilon$-transitions cannot be removed in timed automata.
- Timed automata with $\varepsilon$-transitions are strictly more expressive than timed automata without $\varepsilon$-transitions.

\[
x = 1, \ a, \ x := 0 \]
\[
x = 1, \ \varepsilon, \ x := 0
\]
Outline

1 Introduction

2 Control synthesis games

3 Control under partial observation

4 Fault diagnosis

5 Conclusion and further developments
An example, the car periphery supervision

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

© Society of Automotive Engineers Inc.
Control synthesis games

Environment against controller
(Non-symmetrical game)

- some actions are controllable $\Sigma_c$
- some actions are uncontrollable $\Sigma_u$
- player “environment” can:
  - interrupt time elapsing,
  - enforce zeno behaviours
  - ...

- a plant $P$ 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

\[ f : \text{Runs}(\mathcal{P}) \rightarrow \Sigma_c \cup \{\lambda\} \quad \lambda : \text{time elapsing} \]
A strategy is a partial function

\[ f : Runs(\mathcal{P}) \rightarrow \Sigma_c \cup \{\lambda\} \quad \lambda : \text{time elapsing} \]

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 strategy is a partial function

\[ f : \text{Runs}(P) \longrightarrow \Sigma_c \cup \{\lambda\} \]

\( \lambda \): 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 \( P \)
A strategy is a partial function

\[ f : \text{Runs}(\mathcal{P}) \rightarrow \Sigma_c \cup \{\lambda\} \]

\[ \lambda : \text{time elapsing} \]

- 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
  \[ f : \text{Runs}(\mathcal{P}) \rightarrow \Sigma_c \cup \{\lambda\} \]
  \[ \lambda : \text{time elapsing} \]
  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
A strategy is a partial function

\[ f : \text{Runs}(\mathcal{P}) \rightarrow \Sigma_c \cup \{ \lambda \} \]

\[ \lambda : \text{time elapsing} \]

- 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: 
\[
\begin{cases}
  f(\ell_0, x < 1) = \lambda \\
  f(\ell_0, x = 1) = a \\
  f(\ell_1, x < 2) = \lambda \\
  f(\ell_1, x = 2) = b \\
  f(\ell_2, x = 2) = c
\end{cases}
\]
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:

\[ c\text{Pred}(X) = \bigcup_{c \in \Sigma_c} \text{Pred}^c(X) \]

\[ u\text{Pred}(X) = \bigcup_{u \in \Sigma_u} \text{Pred}^u(X) \]
Computing winning states

- controllable and uncontrollable discrete predecessors:

\[
\text{cPred}(X) = \bigcup_{c \in \Sigma_c} \text{Pred}^c(X) \quad \text{uPred}(X) = \bigcup_{u \in \Sigma_u} \text{Pred}^u(X)
\]

- time controllable predecessor of \( X \) (\( \text{Pred}_\delta \)):

\[
s \longrightarrow_{t} s' \in X
\]
Computing winning states

- controllable and uncontrollable discrete predecessors:

\[
c\text{Pred}(X) = \bigcup_{c \in \Sigma_c} \text{Pred}^c(X) \quad \text{and} \quad u\text{Pred}(X) = \bigcup_{u \in \Sigma_u} \text{Pred}^u(X)
\]

- time controllable predecessor of \( X \) (\( \text{Pred}_\delta \)):

\[
s \quad \overset{t'}{\longrightarrow} \quad t - t' \quad \overset{\bar{X}}{\longrightarrow} \quad s' \in X
\]
Computing winning states

- controllable and uncontrollable discrete predecessors:
  \[ \text{cPred}(X) = \bigcup_{c \in \Sigma_c} \text{Pred}^c(X) \]
  \[ \text{uPred}(X) = \bigcup_{u \in \Sigma_u} \text{Pred}^u(X) \]

- time controllable predecessor of \( X \) (\( \text{Pred}_\delta \)):
  \[ s \xrightarrow{t'} t - t' \xrightarrow{u} s' \in X \]

- winning states: greatest fixed point of
  \[ \pi(X) = \text{Pred}_\delta(X \cap \text{cPred}(X), \text{uPred}(\overline{X})) \]
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

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

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

Environment is seen through sensors.

[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

Stumbling blocks:
- $\varepsilon$-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

$\ell \xrightarrow{(\ell, g, a, C := 0, \ell'), z := 0} \cdot \xrightarrow{g \land z = 0, a, C := 0} \ell'$
Safety and reachability control under partial observation is undecidable.

\[\text{by reduction of universality problem for timed automata}\]

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

\[
\begin{array}{l}
\ell \xrightarrow{g, a, C := 0} \ell' \\
\text{is replaced by}
\end{array}
\]

\[
\begin{array}{l}
\ell \xrightarrow{(\ell, g, a, C := 0, \ell'), z := 0} g \land z = 0, a, C := 0
\end{array}
\]

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 \(\Delta = \{(\ell, g, a, C := 0, \ell')\ \text{transition of } \mathcal{A}\}\) and make all actions from \(\Delta\) non-observable.
Take $\mathcal{A}$ a (complete) timed automaton. Construct $\mathcal{P}$ as follows.

There exists a controller $\mathcal{C}$ which enforces non-final states of $\mathcal{P}$
iff
$\mathcal{A}$ is not universal.
Take $A$ a (complete) timed automaton. Construct $P$ as follows.

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

There exists a controller $C$ which enforces non-final states of $P$ iff $A$ is not universal.

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

$P \parallel \gamma$ represents all the possible runs for $\gamma$ with transitions in $A$. 

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

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

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 $\gamma$ with transitions in $\mathcal{A}$

NB: this undecidability result seems robust...
Fixing resources

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

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

[Bouyer, D’Souza, Madhusudan, Petit 2003]
Fixing resources

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

\textbf{Resources: } \mu = (X, m, \text{max})

\[ x \sim c \quad \Rightarrow \quad c \in \frac{\mathbb{Z}}{m} \text{ and } |c| \leq \text{max} \]

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

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

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

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

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

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

Resources: \( \mu = (X, m, \text{max}) \)

\[ x \sim c \implies c \in \frac{\mathbb{Z}}{m} \quad \text{and} \quad |c| \leq \text{max} \]

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

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

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

System:
Principle of fault diagnosis

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**

**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:**

**Observation:** «ab» or «ac»
Fault diagnosis

Principle of fault diagnosis

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

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

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

**System:**

\[ \begin{align*}
  & f \\
  & u \\
  & a \\
  & b \\
  & c \\
\end{align*} \]

**Sensors:**

\[ \begin{align*}
  & \varepsilon \\
  & a \\
  & b \\
  & c \\
\end{align*} \]

**Observation:** «ab» or «ac»

Did a fault occur?
The timed framework

- Plant = timed automaton
- $\Sigma_o$ observable events, and $\Sigma_u$ unobservable events
The timed framework

- Plant = timed automaton
- \( \Sigma_0 \) observable events, and \( \Sigma_u \) unobservable events

**Pb:** Given an observation (timed word over \( \Sigma_0 \)), did a fault occur?
The timed framework

- Plant = timed automaton
- \( \Sigma_o \) observable events, and \( \Sigma_u \) unobservable events

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

**Aim:** answer within \( \Delta \) units of time
The timed framework

- Plant = timed automaton
- $\Sigma_o$ observable events, and $\Sigma_u$ unobservable events

**Pb:** Given an observation (timed word over $\Sigma_o$), did a fault occur?

**Aim:** answer within $\Delta$ 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
- $\Sigma_o$ observable events, and $\Sigma_u$ unobservable events

**Pb:** Given an observation (timed word over $\Sigma_o$), did a fault occur?
**Aim:** answer within $\Delta$ 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 $\pi(w)$
A $\Delta$-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \rightarrow \{0, 1\}$ such that:
A $\Delta$-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_o) \rightarrow \{0, 1\}$ such that:

- for every non-faulty execution $\rho$ 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 $\rho$ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 0$
- for every Δ-faulty execution $\rho$ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 1$
A Δ-diagnoser for $\mathcal{P}$ is a function $D : TW(\Sigma_\circ) \to \{0, 1\}$ such that:
- for every non-faulty execution $\rho$ of $\mathcal{P}$, $D(\pi_{\Sigma_\circ}(\rho)) = 0$
- for every Δ-faulty execution $\rho$ of $\mathcal{P}$, $D(\pi_{\Sigma_\circ}(\rho)) = 1$

**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 $D : TW(\Sigma_\circ) \rightarrow \{0, 1\}$ such that:

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

A solution [Tripakis02]: state estimation

⇒ the Δ-diagnosis problem is PSPACE-complete
Δ-diagnosis

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

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

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
A $\Delta$-diagnoser for $P$ is a function $D : TW(\Sigma_\circ) \rightarrow \{0, 1\}$ such that:

- for every non-faulty execution $\rho$ of $P$, $D(\pi_{\Sigma_\circ}(\rho)) = 0$
- for every $\Delta$-faulty execution $\rho$ of $P$, $D(\pi_{\Sigma_\circ}(\rho)) = 1$

A solution [Tripakis02]: state estimation

$\Rightarrow$ the $\Delta$-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

$\Rightarrow$ Our aim: build a deterministic diagnoser $O$...
**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 $\rho$ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 0$
- for every $\Delta$-faulty execution $\rho$ of $\mathcal{P}$, $D(\pi_{\Sigma_o}(\rho)) = 1$

A solution **[Tripakis02]**: state estimation  
→ the $\Delta$-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
  
  \[ x = 1, u, x := 0 \]

\[
\begin{align*}
  & x = 0, f \\
  & 0 < x < 1, a
\end{align*}
\]

\[
\begin{align*}
  & x = 0, a
\end{align*}
\]
Diagnosis with deterministic timed automata

- less general than previous diagnosis
  \[ x = 1, u, x := 0 \]
  \[ x = 0, f \]
  \[ x = 0, a \]
  \[ 0 < x < 1, a \]

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

- less general than previous diagnosis
  \[ x = 1, u, x := 0 \]

- 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
  \[ x = 1, u, x := 0 \]

- 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

- restriction to bounded resources \[ \mu = (X, m, \text{max}) \]
Diagnosis with deterministic timed automata

- less general than previous diagnosis
  \[ x = 1, u, x := 0 \]

- 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

- restriction to bounded resources \[ \mu = (X, m, \text{max}) \]

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

\[ \Delta \text{-diagnosis of timed systems with } DTA_\mu \text{ is } 2\text{EXPTIME-complete.} \]
We will transform the diagnosis problem into a two-player safety game:

- one player is the observer □
- the other player is the environment ○

The plant is $\Delta$-DTA$_\mu$-diagnosable iff □ 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?
Fault diagnosis

\[
P \quad \begin{array}{c}
1 \quad a, \ x := 0 \quad 2 \\
\end{array}
\]

\[
\begin{align*}
x & > 1, f.b \\
x & \leq 1, u.b
\end{align*}
\]

\[
R \quad \begin{array}{c}
1; x = y = 0 \quad 2; x = y = 0 \\
\end{array}
\]

\[
\begin{align*}
& a, x := 0 \quad y := 0 \\
& 2; x = y = 0 \\
& a, x := 0 \\
& a, x := 0
\end{align*}
\]

\[
\begin{align*}
& x > 1 \land y > 1, f.b \\
& x \leq 1 \land y \leq 1, u.b \\
& x > 1 \land y > 1, f.b \\
& x \leq 1 \land y > 1, u.b \\
& x \leq 1 \land y \leq 1, u.b
\end{align*}
\]
Fault diagnosis

\[ P \]

\[ \begin{align*}
  \text{1; } & x := 0 \quad a, \quad x > 1, f \cdot b \\
  \text{2; } & y > x = 0 \quad x \leq 1, u \cdot b
\end{align*} \]

\[ R \]

\[ \begin{align*}
  \text{1; } & x = y = 0 \quad a, \ x := 0, \ y := 0 \quad x > 1 \land y > 1, f \cdot b \\
  \text{2; } & x = y = 0 \quad a, \ x := 0 \quad x \leq 1 \land y \leq 1, u \cdot b \\
  \text{2; } & y > x = 0 \quad x \leq 1 \land y > 1, u \cdot b \\
\end{align*} \]
\[
\begin{align*}
\mathcal{P} & \quad x > 1, f.b \\
1 & \quad a, x := 0 \quad \rightarrow \quad 2 \quad \rightarrow \quad f \\
2 & \quad x \leq 1, u.b \\
\pi(\mathcal{R}) & \\
1; x = y = 0 & \quad y > 1, b \\
a, y := 0 & \quad \rightarrow \quad 2; x = y = 0 \\
2; x = y = 0 & \quad y \leq 1, b \\
a & \quad \rightarrow \quad 2; y > x = 0 \\
2; y > x = 0 & \quad y > 1, b \\
a & \quad \rightarrow \quad 2; y > x = 0 \\
2; y > x = 0 & \quad y \leq 1, b \\
\end{align*}
\]
\[ P \]

\[ a, x := 0 \]

\[ f \]

\[ x > 1, f \cdot b \]

\[ u \]

\[ x \leq 1, u \cdot b \]

\[ \text{Det}(\pi(\mathcal{R})) \]

\[ 1; x = y = 0 \]

\[ a, y := 0 \]

\[ 2; x = y = 0 \]

\[ y > 1, b \]

\[ f; \ldots \]

\[ y \leq 1, b \]

\[ u; \ldots \]

\[ 2; y \geq x = 0 \]

\[ y > 1, b \]

\[ f; \ldots \]

\[ y \leq 1, b \]

\[ u; \ldots \]

\[ 2; x = y = 0 \]

\[ y \leq 1, b \]

\[ u; \ldots \]
Fault diagnosis

\[ \mathcal{P} \]

1. \( a, \ x := 0 \)
   2. \( x > 1, \ f.b \)
   3. \( x \leq 1, \ u.b \)

\[ G_{\mathcal{P}, \mu} \]

1; \( x = y = 0 \)
   \( y := 0 \)
   2; \( x = y = 0 \)
   3. \( y > 1, b \)
   4. \( y \leq 1, b \)
   5. \( y > 1, b \)
   6. \( y \leq 1, b \)
   7. \( f; \cdots \)
   8. \( u; \cdots \)
   9. \( f; \cdots \)
   10. \( u; \cdots \)
   11. \( u; \cdots \)
Proposition

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

$\Rightarrow$ $\Delta$-$\text{DTA}_\mu$-diagnosability is in $2\text{EXPTIME}$
Δ-DTA_μ-observability if 2EXPSPACE-hard

→ By reduction of the acceptance of an Alternating Turing Machine using exponential space
Δ-DTAμ-observability if 2EXPTIME-hard

→ 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.
△-DTA$_\mu$-observability if 2EXPTIME-hard

→ 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
Fault diagnosis

Shape of the plant

Check initial configuration

Check succ. relation
$\mathcal{O}$ has 1 clock.

\begin{itemize}
  \item $a, \ x := 0$
  \item $a, \ y := 0$
\end{itemize}

$\mathcal{O}$ makes a choice

reset $x$ or $y$
$O$ has 1 clock.

$P$ verifies the choice of $O$ is correct.

$O$ makes a choice
reset $x$ or $y$
\( \mathcal{O} \) has 1 clock.

\[ a, x := 0 \quad a, y := 0 \quad x > 2 \land y > 1, a \quad x > 2 \land y < 1, a \quad \checkmark, z := 0 \quad z = 0, a \]

\( \mathcal{P} \) can force \( \mathcal{O} \) “remember” \( x \):
\( \mathcal{O} \) has 1 clock.

\[ a, \ x := 0 \quad a, \ y := 0 \]

\[ x > 2 \land y > 1, \ a \]

\[ x > 2 \land y < 1, \ a \]

\[ x < 2 \land y > 1, \ a \]

\[ ?, \ z := 0 \quad \checkmark, \ z := 0 \quad ? , \ z := 0 \]

\[ z = 0, \ a \quad z = 0, \ a \quad z = 0, \ a \]

\( \mathcal{P} \) 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
\( \mathcal{O} \) has 1 clock.

\[ a, \ x := 0 \quad \text{\&} \quad a, \ y := 0 \]

\( \mathcal{P} \)

\( \mathcal{O} \) 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.

\[ P \\
\begin{array}{c}
\tau, x := 0 \\
\end{array} \quad \begin{array}{c}
\tau, y := 0 \\
\end{array} \\quad \begin{array}{c}
x > 2 \land y > 1, \tau \\
x > 2 \land y < 1, \tau \\
x < 2 \land y > 1, \tau \\
\end{array} \quad \begin{array}{c}
\checkmark, z := 0 \\
\checkmark, z := 0 \\
?, z := 0 \\
\end{array} \quad \begin{array}{c}
z = 0, \tau \\
z = 0, \tau \\
z = 0, f.\tau \\
\end{array} \]

\( \mathcal{O} \) makes a choice
- reset \( x \) or \( y \)
- \( P \) verifies the choice of \( \mathcal{O} \) is correct

\( P \) can force \( \mathcal{O} \) “remember” \( x \):
- if \( \mathcal{O} \) “remembers” \( y \), diagnosis is impossible
\( \mathcal{O} \) has 1 clock.

\[ x > 2 \land y > 1, a \]

\[ x < 2 \land y > 1, a \]

\[ x > 2 \land y < 1, a \]

\( \mathcal{P} \)

\[ a, x := 0 \]

\[ a, y := 0 \]

\( \mathcal{O} \) 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.

\(\mathcal{P}\)

\[\begin{align*}
& a, \ x := 0 \\
& a, \ y := 0
\end{align*}\]

\(\mathcal{O}\) 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.

\( \mathcal{P} \)

\( a, x := 0 \quad a, y := 0 \)

\( x > 2 \land y > 1, a \quad x > 2 \land y < 1, a \quad \checkmark, z := 0 \quad z = 0, a \quad ?, z := 0 \quad z = 0, f.a \)

\( \mathcal{O} \) 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.

\[
\begin{align*}
\mathcal{P} & \quad a, \ x := 0 \quad a, \ y := 0 \\
\mathcal{Q} & \quad x > 2 \land y > 1, \ a \\
\mathcal{P} & \quad a, \ y := 0 \\
\mathcal{Q} & \quad x > 2 \land y < 1, \ a \\
\mathcal{P} & \quad \checkmark, \ z := 0 \\
\mathcal{Q} & \quad z = 0, \ a \\
\mathcal{P} & \quad a, \ x := 0 \\
\mathcal{Q} & \quad x < 2 \land y > 1, \ a \\
\mathcal{P} & \quad ?, \ z := 0 \\
\mathcal{Q} & \quad z = 0, \ f.a
\end{align*}
\]

\( \mathcal{O} \) 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.

\[ a, x := 0 \]

\[ a, y := 0 \]

\[ x > 2 \land y > 1, a \]

\[ x > 2 \land y < 1, a \]

\[ x < 2 \land y > 1, a \]

\[ ?, z := 0 \]

\[ z = 0, a \]

\[ ? \]

\[ z = 0, a \]

\[ \checkmark, z := 0 \]

\[ z = 0, a \]

\[ ?, z := 0 \]

\[ z = 0, f.a \]

\( \mathcal{P} \) makes a choice

\( \mathcal{O} \) makes a choice

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

\( \mathcal{P} \) can force \( \mathcal{O} \) “remember” \( x \):

- if \( \mathcal{O} \) “remembers” \( y \), diagnosis is impossible
\( O \) has 1 clock.

\[
\begin{align*}
\text{if } x > 2 \land y > 1, a \\
\text{if } x < 2 \land y > 1, a \\
\text{if } x > 2 \land y < 1, a \\
\text{\( ? \), } z := 0 &\quad \text{\( z = 0, a \)} \\
\text{\( \checkmark \), } z := 0 &\quad \text{\( z = 0, a \)} \\
\text{\( ? \), } z := 0 &\quad \text{\( z = 0, f.a \)}
\end{align*}
\]

\( P \) can force \( O \) “remember” \( x \):
- if \( O \) “remembers” \( y \), diagnosis is impossible
- if \( O \) “remembers” \( x \), diagnosis is possible
\( O \) has 1 clock.

\[ a, x := 0 \quad a, y := 0 \]

\( P \) verifies the choice of \( O \) is correct

\( P \) can force \( O \) “remember” \( x \):
- if \( O \) “remembers” \( y \), diagnosis is impossible
- if \( O \) “remembers” \( x \), diagnosis is possible
\( \mathcal{O} \) has 1 clock.

\[ a, \ x := 0 \rightarrow \ \mathcal{P} \]

\[ \text{\( x > 2 \land y > 1, a \)} \]

\[ a, \ y := 0 \rightarrow \ \mathcal{P} \]

\[ \text{\( x > 2 \land y < 1, a \)} \]

\[ \mathcal{P} \text{ verifies the choice of } \mathcal{O} \text{ is correct} \]

\[ \mathcal{O} \text{ makes a choice} \]

\[ \text{reset } x \text{ or } y \]

\( \mathcal{P} \) can force \( \mathcal{O} \) “remember” \( x \):

- if \( \mathcal{O} \) “remembers” \( y \), diagnosis is impossible
- if \( \mathcal{O} \) “remembers” \( x \), diagnosis is possible
\(O\) has 1 clock.

\[x > 2 \land y > 1, a\]

\[x < 2 \land y > 1, a\]

\(P\)

\[a, x := 0\]

\(a, y := 0\)

\(q\)

\[x > 2 \land y < 1, a\]

\(\checkmark, z := 0\)

\[z = 0, a\]

\[?, z := 0\]

\[z = 0, a\]

\[z = 0, f.a\]

\(O\) makes a choice

reset \(x\) or \(y\)

\(P\) verifies the choice of \(O\) is correct

\(P\) can force \(O\) “remember” \(x\):

- if \(O\) “remembers” \(y\), diagnosis is impossible
- if \(O\) “remembers” \(x\), diagnosis is possible
\( \mathcal{O} \) has 1 clock.

\[ x > 2 \land y > 1, a \]

\( \mathcal{P} \)

\[ a, x := 0 \quad a, y := 0 \]

\( \mathcal{O} \) 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
- if \( \mathcal{O} \) “remembers” \( x \), diagnosis is possible
\( \mathcal{O} \) has 1 clock.

\( \mathcal{P} \)

\( a, \ x := 0 \quad a, \ y := 0 \quad x > 2 \land y < 1, \ a \)

\( x > 2 \land y > 1, \ a \)

\( x < 2 \land y > 1, \ a \)

\( \mathcal{O} \) 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
- if \( \mathcal{O} \) “remembers” \( x \), diagnosis is possible
\( O \) has 1 clock.

\[ a, x := 0 \quad a, y := 0 \]

\( P \) can force \( O \) “remember” \( x \):
- if \( O \) “remembers” \( y \), diagnosis is impossible
- if \( O \) “remembers” \( x \), diagnosis is possible
\( O \) has 1 clock.

\[
x > 2 \land y > 1, a
\]

\[
x < 2 \land y > 1, a
\]

\[ P \] can force \( O \) “remember” \( x \):
- if \( O \) “remembers” \( y \), diagnosis is impossible
- if \( O \) “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** \( p_1 \lor \neg p_3: \)

Diagram showing the encoding process with choices and breaking the uncertainty.
Diagnosis by event-recording timed automata

- one clock $x_a$ per event $a$
- clock $x_a$ is reset when $a$ occurs
Fault diagnosis

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

- 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 $\Delta$ parameter
- Control under partial observation for other classes of systems
  *(e.g. o-minimal hybrid games)*
Conclusion and further developments

Bibliography I


Conclusion and further developments

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).


[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