Synthesis of Timed Systems

Patricia Bouyer

LSV – CNRS & ENS de Cachan

Spring School GAMES
Model-checking

Does the system satisfy the property?
Model-checking

Does the system satisfy the property?

Modelling

Model-checking Algorithm
Controller synthesis

Can we force the system satisfying the property?

Modelling
Controller synthesis

Can we force the system satisfying the property?

Modelling

Controller synthesis
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

© Society of Automotive Engineers Inc.
1. Preliminaries on timed systems
   - The model of timed automata
   - The region abstraction
   - Everything is not that nice!
   - Symbolic manipulation of timed automata

2. On the semantics of timed games

3. Control synthesis games
   - Framework of these games
   - Simple control objectives
   - Control for external specifications
   - Partial observability

4. Troubles with dense-time control
   - Sampling time control
   - Implementability of controllers

5. Conclusion and current 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)

\( x, y : \text{clocks} \)

\[
\begin{array}{c}
\ell_0 \xrightarrow{\delta(4.1)} \ell_0 \xrightarrow{a} \ell_1 \xrightarrow{\delta(1.4)} \ell_1 \xrightarrow{b} \ell_2 \\
x \quad 0 \quad 4.1 \quad 4.1 \quad 5.5 \quad 0 \\
y \quad 0 \quad 4.1 \quad 0 \quad 1.4 \quad 1.4
\end{array}
\]

(clock) valuation
**Timed automata (example)**

\[ x, y : \text{clocks} \]

\[ x \leq 5, \ a, \ y := 0 \]

\[ y > 1, \ b, \ x := 0 \]

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

\[ x \quad 0 \quad 4.1 \quad 4.1 \]

\[ y \quad 0 \quad 4.1 \quad 0 \]

\[ \boxed{5.5} \quad 0 \]

\[ \boxed{1.4} \quad 1.4 \]

\[ (\text{clock}) \text{ valuation} \]

\[ \rightarrow \text{timed word } (a, 4.1)(b, 5.5) \]
Timed automata semantics

- $\mathcal{A} = (\Sigma, L, X, \rightarrow)$ 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

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

\[ b, \ y := 0 \]

\[ y < 1, \ b, \ y := 0 \]
Timed languages

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

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

Discrete-time: 
\[ L_{\text{discrete}} = \emptyset \]
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 cannot be applied
Verification

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

- **Problem:** the set of configurations is infinite
  - ➔ classical methods cannot 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 cannot 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 cannot 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

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

\[ \{x\} < \{y\} \]

“compatibility” between regions and constraints

“compatibility” between regions and time elapsing

→ a bisimulation property
The region abstraction

Equivalence of finite index

- region defined by
  \[ l_x = ]1; 2[, \quad l_y = ]0; 1[ \]
  \{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 \textit{bisimulation} property
The region automaton

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

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

- \( R'' \subseteq g \)
- \([C \leftarrow 0]R'' \subseteq R'\)

\[ \Rightarrow \text{time-abstract bisimulation} \]

\[ \mathcal{L}(\text{reg. aut.}) = \text{UNTIME}(\mathcal{L}(\text{timed aut.})) \]

where \( \text{UNTIME}((a_1, t_1)(a_2, t_2)\ldots) = a_1 a_2 \ldots \)
Time-abstract bisimulation
Time-abstract bisimulation

∀

∃

Spring School GAMES Synthesis of Timed Systems 14 / 80
Time-abstract bisimulation

∀

∃

∀ \delta(d) > 0

δ(d)

∀

∃
Time-abstract bisimulation

∀ \ a → b
∃ \ a → b

∀ \ d > 0 → d
∃ \ d' > 0 → d'
Preliminaries on timed systems

The region abstraction

Time-abstract bisimulation

\( \forall \) a \( \exists \) a for \( \forall d > 0 \) \( \delta(d) \) and \( \exists d' > 0 \) \( \delta(d') \)

\((\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \xrightarrow{a_2, t_2} (\ell_2, v_2) \xrightarrow{a_3, t_3} \ldots\)
Time-abstract bisimulation

\[
\forall \ a \ \exists \ a \\
\exists \ d > 0 \ \exists \ d' > 0
\]

with \( v_i \in R_i \) for all \( i \).


**Time-abstract bisimulation**

∀a \rightarrow \exists a

∀d > 0 \rightarrow \exists d' > 0

\(\delta(d)\) \rightarrow \delta(d')

---

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

\((\ell_0, R_0)\) \xrightarrow{a_1} (\ell_1, R_1) \xrightarrow{a_2} (\ell_2, R_2) \xrightarrow{a_3} \ldots

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 cannot be reduced to finite automata...
A model not far from undecidability

Properties

- Universality is undecidable
- Inclusion is undecidable
- Determinizability is undecidable
- Complementability is undecidable
- ...

[Alur & Dill 90’s]
[Alur & Dill 90’s]
[Tripakis 2003]
[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 \((x\text{ and } y)\):

- **Incrementation:**
  
  \((p)\): \(x := x + 1; \text{ goto (q)}\)

- **Decrementation:**
  
  \((p)\): if \(x > 0\) then \(x := x - 1; \text{ goto (q)}\) 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.

\[ b_i, \ x := 0 \]
\[ x < 1, \ c, \ x := 0 \]
\[ x = 1, \ \neg c \]
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 $\varepsilon$-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.

$$x = 1, \ a, \ x := 0$$

$$x = 1, \ \varepsilon, \ x := 0$$
On \textit{zeno} behaviours

\textbf{Definition}

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

\(\rightarrow\) infinitely many discrete events within a bounded amount of time

\textbf{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 \((a_1, t_1)(a_2, t_2)\ldots\) 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**

\[
(a, \frac{1}{2}) (a, \frac{3}{4}) (a, \frac{7}{8}) \ldots (a, 1 - \frac{1}{2^n}) \ldots
\]

**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

\[ \varphi ::= x \sim c \mid x - y \sim c \mid \varphi \land \varphi \]

**Example**

The constraint \((x_1 \geq 3) \land (x_2 \leq 5) \land (x_1 - x_2 \leq 4)\) defines the zone:
A symbolic representation for timed systems

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

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

**Example**

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

$$\{v + t \mid v \models \varphi \text{ and } t \geq 0\} \text{ is a zone}$$

$$\{[Y \leftarrow 0]v \mid v \models \varphi\} \text{ is a zone}$$

$$\{v - t \mid v \models \varphi \text{ and } t \geq 0\} \text{ is a zone}$$

...
Outline

1 Preliminaries on timed systems
   • The model of timed automata
   • The region abstraction
   • Everything is not that nice!
   • Symbolic manipulation of timed automata

2 On the semantics of timed games

3 Control synthesis games
   • Framework of these games
   • Simple control objectives
   • Control for external specifications
   • Partial observability

4 Troubles with dense-time control
   • Sampling time control
   • Implementability of controllers

5 Conclusion and current developments
Smiley
in the last ten years, a prolific literature
On the semantics of timed games

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 $(\delta_i, a_i)$
  - Semantics with surprise
    - each player $i$ chooses either a delay $\delta_i$, or an action $a_i$
  - Semantics without surprise
- the next move is determined by the following rules:
  - if $\delta_i < \delta_j$, wait $\delta_i$ and do $a_i$
  - if $\delta_i = \delta_j$, wait $\delta_i$ and do either $a_i$ or $a_j$ (non-deterministic choice)
- classical $\omega$-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)**

\[ x > 1, a_2, x := 0 \]

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

\[
\begin{align*}
    &i \quad [x \leq 0] \\
    &x = 0, a_1 \\
    &x = 0, a_1
\end{align*}
\]

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)

\[ a_2, \ x := 0 \]

\[ 0 < x < 1, \ a_1 \]

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

\[ 0 < x < 1, \ a_1 \]
**Proposition**

“Real” strategies based on regions are not sufficient...

**Example**

![Diagram showing a transition labeled $0 < x < 1, a_1$ from state $p$ to state $q$.]
Proposition

“Real” strategies based on regions are not sufficient...

Example

\[0 < x < 1, a_1\]
**Proposition**

“Real” strategies based on regions are not sufficient...

**Example**

\[
0 < x < 1, a_1
\]
**Proposition**

“Real” strategies based on regions are not sufficient...

**Example**

\[ 0 < x < 1, a_1 \]

Winning states

Winning strategy
Proposition

“Real” strategies based on regions are not sufficient...

Example

\[ 0 < x < 1, a_1 \]

Winning states

Winning strategy

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
1 Preliminaries on timed systems
   - The model of timed automata
   - The region abstraction
   - Everything is not that nice!
   - Symbolic manipulation of timed automata

2 On the semantics of timed games

3 Control synthesis games
   - Framework of these games
   - Simple control objectives
   - Control for external specifications
   - Partial observability

4 Troubles with dense-time control
   - Sampling time control
   - Implementability of controllers

5 Conclusion and current developments
Framework of these games

- some actions are controllable $\Sigma_c$
- some actions are uncontrollable $\Sigma_u$
- 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

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

\( \lambda \): time elapsing
Strategies and controllers

- A **strategy** is a partial function

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

- needs to satisfy some *continuity* property:

\[ f(\rho) = \lambda \implies \exists t > 0, \forall 0 \leq t' < t, \ f(\rho \xrightarrow{\delta(t')} ) = \lambda \]
Strategies and controllers

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

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

  - 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 will run in parallel with \( \mathcal{P} \)

  How powerful can it be?
A **strategy** is a partial function

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

- needs to satisfy some *continuity* property:

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

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
  \[ f : \text{Runs}(\mathcal{P}) \rightarrow \Sigma_c \cup \{\lambda\} \]
  \[ \lambda : \text{time elapsing} \]
  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 will run in parallel with \( \mathcal{P} \)

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

  needs to be *non-restricting* for uncontrollable actions
Control synthesis games

Framework of these games

Strategies and controllers

A **strategy** is a partial function

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

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

- 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 will run in parallel with \(P\)

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 $P \parallel C$

specifications (or winning conditions)
  - **Internal specifications**: conditions on the states of the plant
    - **safety**: the controlled system avoids bad states
    - **reachability**: $C$ 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 $S$
    - representing desired behaviours
      \[ L(\mathcal{P} \parallel \mathcal{C}) \subseteq L(S) \]
    - representing undesired behaviours
      \[ L(\mathcal{P} \parallel \mathcal{C}) \cap L(S) = \emptyset \]
Winning a timed game

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

- Specifications (or winning conditions)
  - **Internal specifications**: conditions on the states of the plant
    - **safety**: the controlled system avoids bad states
    - **reachability**: $C$ enforces a good state
    - ... 
  - **External specifications**: given by a timed automaton $S$
    - representing desired behaviours
      \[ L(P \parallel C) \subseteq L(S) \]
    - representing undesired behaviours
      \[ L(P \parallel C) \cap L(S) = \emptyset \]

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

\[ x \leq 5 \]

\[ l_0 \rightarrow x \geq 1; a; y := 0 \rightarrow l_1 \rightarrow 3 < x; u \rightarrow \text{Bad} \]

\[ 1 \leq x \leq 5; \ c; x := 0 \rightarrow l_2 \rightarrow y \geq 1; b \]

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

**A controller:**

\[ z \leq 2 \]

\[ l_0 \rightarrow z \leq 2; a \rightarrow l_1 \rightarrow [z \leq 3] \]

\[ u \rightarrow l_1 \rightarrow u \]

\[ c; z := 0 \rightarrow l_2 \rightarrow b \]

\[ u \rightarrow l_2 \rightarrow u \]
An example

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

A controller:

A winning strategy: \[
\begin{align*}
& f(l_0, x < 1) = \lambda \\
& f(l_0, x = 1) = a \\
& f(l_1, x < 2) = \lambda \\
& f(l_1, x = 2) = b \\
& f(l_2, x = 2) = c
\end{align*}
\]
Computing winning states

- $\text{Pred}^a(X) = \{s \mid s \xrightarrow{a} s' \text{ with } s' \in X\}$

- controllable and uncontrollable discrete predecessors:

  $\text{cPred}(X) = \bigcup_{c \in \Sigma_c} \text{Pred}^c(X)$

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

- \( \text{Pred}^a(X) = \{ s \mid s \xrightarrow{a} s' \text{ with } s' \in X \} \)
- Controllable and uncontrollable discrete predecessors:
  \[
  \begin{align*}
  \text{cPred}(X) &= \bigcup_{c \in \Sigma_c} \text{Pred}^c(X) \\
  \text{uPred}(X) &= \bigcup_{u \in \Sigma_u} \text{Pred}^u(X)
  \end{align*}
  \]
- Time controllable predecessor of \( X \):
  \[
  s \quad \xrightarrow{t} \quad s' \in X
  \]
Computing winning states

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

\[
\begin{align*}
\text{cPred}(X) &= \bigcup_{c \in \Sigma_c} \text{Pred}^c(X) \\
\text{uPred}(X) &= \bigcup_{u \in \Sigma_u} \text{Pred}^u(X)
\end{align*}
\]

- time controllable predecessor of \( X \):

\[
\begin{array}{c}
s \xrightarrow{t'} s' \in X \\
\text{Pred}(X) = \{ s \mid s \xrightarrow{a} s' \text{ with } s' \in X \}
\end{array}
\]
Computing winning states

- \( \text{Pred}^a(X) = \{ s \mid s \xrightarrow{a} s' \text{ with } s' \in X \} \)
- Controllable and uncontrollable discrete predecessors:
  \[
  \begin{align*}
  \text{cPred}(X) &= \bigcup_{c \in \Sigma_c} \text{Pred}^c(X) \\
  \text{uPred}(X) &= \bigcup_{u \in \Sigma_u} \text{Pred}^u(X)
  \end{align*}
  \]
- Time controllable predecessor of \( X \):
  \[
  s \xrightarrow{t'} t \xrightarrow{t-t'} s' \in X
  \]
- \( \text{Pred}_\delta(X, Y) = \{ s \mid \exists t \geq 0, s \xrightarrow{t} s', s' \in X \text{ and } \text{Post}_{[0,t]}(s) \subseteq \overline{Y} \} \)

Where \( \text{Post}_{[0,t]}(s) = \{ s' \mid \exists 0 \leq t' \leq t, s \xrightarrow{t'} s' \} \)
Computing winning states

\[ \pi(X) = \text{Pred}_\delta(X \cap \text{cPred}(X), \text{uPred}($X$)) \]

**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,

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

- $u\text{Pred}(\ell, R)$ is a finite set of regions
- $c\text{Pred}(\ell, R)$ is a finite set of regions
- $\text{Pred}_\delta((\ell, R), \cup_{i \in I}(\ell_i, R_i))$ 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,

- $u\text{Pred}(\ell, R)$ is a finite set of regions
- $c\text{Pred}(\ell, R)$ is a finite set of regions
- $\text{Pred}_{\delta}((\ell, R), \cup_{i \in I}(\ell_i, R_i))$ 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 $\lambda$, 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 $\lambda$, and the second one on which the strategy is defined as being $c$.

$\Rightarrow$ 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 $P$ 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 if and only if $M$ accepts $w$
External specifications

**External specifications**: given by a timed automaton $S$

- representing desired behaviours

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

- representing undesired behaviours

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

Theorem

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

[D’Souza, Madhusudan 2002]
Several undecidability results

[Di'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

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 $\Sigma$
Several undecidability results

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

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

- take $A$ a timed automaton over $\Sigma$
- take $P$ universal plant over $\Sigma$

[D’Souza, Madhusudan 2002]
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 $\Sigma$
- take $\mathcal{P}$ universal plant over $\Sigma$
- assume all actions of $\Sigma$ are uncontrollable

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

**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 $\Sigma$
- take $\mathcal{P}$ universal plant over $\Sigma$
- assume all actions of $\Sigma$ are **uncontrollable**

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

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

[D’Souza, Madhusudan 2002]
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)

[Đ’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 $\Sigma$
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 $A$ a timed automaton over $\Sigma$
- take $P$ universal plant over $\Sigma$
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 $\Sigma$
- take $\mathcal{P}$ universal plant over $\Sigma$
- assume all actions of $\Sigma$ are **controllable**
Several undecidability results (cont’d)

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 $\Sigma$
- take $\mathcal{P}$ universal plant over $\Sigma$
- assume all actions of $\Sigma$ are controllable

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

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.

Difficulties:
- $\varepsilon$-transitions cannot be removed from timed automata.
- Timed automata cannot be determinized.
Timed framework: what’s specific?

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

![Diagram showing plant and controller with clocks]

- **Unreadable clocks**
- **Readable clocks (can not be reset by the controller)**
- **Clocks belonging to the controller**
**Timed framework: what’s specific?**

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

![Diagram showing plant and controller with clocks]

- Unreadable clocks
- Readable clocks (can not be reset by the controller)
- Clocks belonging to the controller

- Which constants clocks can be compared with? \( \sim \) \((X, m, \text{max})\)

\[
x \sim c \implies c \in \mathbb{Z} / m \text{ and } |c| \leq \text{max}
\]
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? \( \sim (X, m, \text{max}) \)

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

Resources
Timed framework: what’s specific?

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

  ![Diagram showing plant and controller with clocks]

  - Unreadable clocks
  - Readable clocks (can not be reset by the controller)
  - Clocks belonging to the controller

- Which constants clocks can be compared with? \( \sim (X, m, \text{max}) \)

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

**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)

![Diagram showing plant and controller with clocks]

- Unreadable clocks
- Readable clocks (can not be reset by the controller)
- Clocks belonging to the controller

- Which constants clocks can be compared with? \( \sim (X, m, \text{max}) \)

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

\( \rightarrow \) **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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Desired behaviors</td>
<td>Undesired behaviors</td>
</tr>
<tr>
<td>Non-fixed</td>
<td>Decidable [DM02]</td>
<td>Undecidable [DM02]</td>
</tr>
</tbody>
</table>
# Summary of previous results

## Full observability hypothesis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Desired behaviors</td>
<td>Undesired behaviors</td>
</tr>
<tr>
<td>Fixed</td>
<td>Decidable [WTH91, AMPS98]</td>
<td>Undecidable [DM02]</td>
</tr>
<tr>
<td>Non-fixed</td>
<td>Decidable [DM02]</td>
<td>Undecidable [DM02]</td>
</tr>
</tbody>
</table>

## Partial observability hypothesis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Desired behaviors</td>
<td>Undesired behaviors</td>
</tr>
<tr>
<td>Fixed</td>
<td>?</td>
<td>Undecidable</td>
</tr>
<tr>
<td>Non-fixed</td>
<td>?</td>
<td>Undecidable</td>
</tr>
</tbody>
</table>
Notations

• some actions are controllable $\Sigma_c$

• some actions are uncontrollable $\Sigma_u$
  • some uncontrollable actions are observable $\Sigma^o_u$
  • some uncontrollable actions are non-observable $\Sigma^n_u$

• a **plant** $\mathcal{P}$ is a DTA over $\Sigma_c \cup \Sigma^o_u \cup \Sigma^n_u$

• a **controller** $\mathcal{C}$ is a DTA over $\Sigma_c \cup \Sigma^o_u$

• the **controlled system** is $\mathcal{P} \parallel \mathcal{C}$ where synchronization is only over observable and controllable events
### Full observability hypothesis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Desired behaviors</td>
</tr>
<tr>
<td>Non-fixed</td>
<td>Decidable [DM02]</td>
<td>Undecidable [DM02]</td>
</tr>
</tbody>
</table>

### Partial observability hypothesis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Desired behaviors</td>
</tr>
<tr>
<td>Fixed</td>
<td>?</td>
<td>Undecidable</td>
</tr>
<tr>
<td>Non-fixed</td>
<td>Undecidable [BDMP03]</td>
<td>Undecidable</td>
</tr>
</tbody>
</table>

**Remark:** reachability and safety control problems are undecidable!
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.

$g, a, C := 0 \xrightarrow{\ell} \ell'$ is replaced by

$\ell \xrightarrow{\ell, g, a, C := 0, \ell'}, z := 0 \xrightarrow{g \land z = 0, a, C := 0} \ell'$
Reachability control under partial observability

Theorem

Reachability control under partial observation is undecidable.

→ by reduction of universality problem for timed automata

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

\[
\ell, g, a, C := 0 \quad \xrightarrow{} \quad \ell',
\]

is replaced by

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

Thus,

\begin{itemize}
  \item \( P \) is a deterministic timed automaton, thus a plant
  \item \((\delta_0, t_0)(a_0, t'_0)(\delta_1, t_1)(a_1, t'_1)\ldots\) is accepted by \( P \) iff \( t_i = t'_i \) for every \( i \) and \((a_0, t_0)(a_1, t_1)\ldots\) is accepted by \( A \) along the path \( \delta_0\delta_1\ldots \)
\end{itemize}

We note \( \Delta = \{(\ell, g, a, C := 0, \ell') \mid \text{transition of } A\} \) 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 $\mathcal{A}$ 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

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 $\mathcal{A}$
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 another $c$ 1 t.u. later.

\[ b_i, \ x := 0 \quad x < 1, \ c, \ x := 0 \quad x < 1 \quad x > 1, \ c \quad x \geq 1, \neg c \]
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.

There is a controller to avoid red states iff the two-counter machine halts.
### Full observability hypothesis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-fixed</td>
<td>Decidable [DM02]</td>
<td>Undecidable [DM02]</td>
<td>Undecidable [DM02]</td>
</tr>
</tbody>
</table>

### Partial observability hypothesis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed</td>
<td>Decidable [BDMP03]</td>
<td>Undecidable</td>
<td>Decidable [BDMP03]</td>
</tr>
<tr>
<td>Non-fixed</td>
<td>Undecidable</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
</tbody>
</table>
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:** \( \mu = (X, m, \max) \)

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

A controller will be an automaton over *symbolic alphabet*

\[ \Gamma = \mathcal{G}(\mu) \times (\Sigma_c \cup \Sigma_o) \times 2^X \]

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

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

- \(0 < x < 1 \land y = 1\) is an atomic constraint
- \(1 < x < 2 \land y > 2\) is an atomic constraint
- \(x = 0 \land y \geq 1\) is *not* an atomic constraint

A transition of the controller is thus of the form

\[ \ell \xrightarrow{0 < x < 1 \land y > 2, a, x := 0} \ell' \]
Timed games

On the same symbolic alphabet \( \Gamma \) as the controller:

-
Timed games

On the same symbolic alphabet $\Gamma$ as the controller:

Player C
Timed games

On the same symbolic alphabet $\Gamma$ as the controller:
Timed games

On the same symbolic alphabet $\Gamma$ as the controller:

![Diagram showing game states and transitions]
Timed games

On the same symbolic alphabet $\Gamma$ as the controller:

Player E
**Timed games**

On the same symbolic alphabet $\Gamma$ as the controller:

![Diagram showing a game graph with Player C highlighted.]
Timed games

On the same symbolic alphabet $\Gamma$ as the controller:
Timed games

On the same symbolic alphabet $\Gamma$ as the controller:
Timed games

On the same symbolic alphabet $\Gamma$ as the controller:

whether the play $\gamma$ is winning or not depends on the synchronization of the plant with $\gamma$:

$$L(P \parallel \gamma) \cap L(S) = \emptyset$$
Timed games

On the same symbolic alphabet $\Gamma$ as the controller:

- Whether the play $\gamma$ is winning or not depends on the synchronization of the plant with $\gamma$:

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

- 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 $\Gamma$
Untimed games

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

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

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

is regular (where $\Gamma$ is the symbolic alphabet for the controller).
Untimed games

- **Construction of the arena:** universal automaton $U_\Gamma$ over $\Gamma$
  - additional condition for non-restricting hypothesis

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

- **Construction of the arena:** universal automaton $\mathcal{U}_\Gamma$ over $\Gamma$
  - 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
  \[
  \{ \gamma \in \Gamma^\infty \mid L(\mathcal{P} \parallel \gamma) \cap L(S) = \emptyset \}
  \]
  is regular (where $\Gamma$ is the symbolic alphabet for the controller).
Untimed games

- **Construction of the arena:** universal automaton $\mathcal{U}_{\Gamma}$ over $\Gamma$
  - additional condition for non-restricting hypothesis
  - what about non-blocking hypothesis?
  - we need a more involved construction with information on $P$
    (projection of the region automaton of $P \parallel \mathcal{U}_{\Gamma}$ onto observable actions and clocks)

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

- **Construction of the arena:** universal automaton $U_{\Gamma}$ over $\Gamma$
  - additional condition for non-restricting hypothesis
  - what about non-blocking hypothesis?
    - we need a more involved construction with information on $P$
      (projection of the region automaton of $P \parallel U_{\Gamma}$ onto observable actions and clocks)

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

  ➔ classical untimed game

  ➔ the problem is $2\text{EXPTIME}$
Untimed games

- **Construction of the arena:** universal automaton $\mathcal{U}_\Gamma$ over $\Gamma$
  - 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
  \[
  \{ \gamma \in \Gamma^\infty \mid L(\mathcal{P} \parallel \gamma) \cap L(S) = \emptyset \}\]
  is regular (where $\Gamma$ 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

Spring School GAMES Synthesis of Timed Systems 62 / 80
Troubles with dense-time control

Outline

1 Preliminaries on timed systems
   • The model of timed automata
   • The region abstraction
   • Everything is not that nice!
   • Symbolic manipulation of timed automata

2 On the semantics of timed games

3 Control synthesis games
   • Framework of these games
   • Simple control objectives
   • Control for external specifications
   • Partial observability

4 Troubles with dense-time control
   • Sampling time control
   • Implementability of controllers

5 Conclusion and current developments
Zenoness...

- is often avoided (by assuming the plant is strictly non-zeno) \[\text{[AMPS98, ...]}\]
- is incorporated in the winning condition \[\text{[dAFH+03]}\]

Is that sufficient?
Problems with dense-time control

\[ x := 0, 1 \]
\[ y := 0 \]
\[ [x \leq 2] \]

\[ x = 1 \quad x := 0 \]
\[ y := 0 \]
\[ z := 0 \]
\[ y = 1 \]
\[ z > 0 \]
\[ y := 0 \]

Spring School GAMES
Synthesis of Timed Systems
Problems with dense-time control

\begin{align*}
\ell_0 : & \quad x := 0, 1, \quad y := 0 \\
\ell_1 : & \quad x = 1, x := 0, \quad a \\
\ell_2 : & \quad y := 0 \\
\ell_3 : & \quad z := 0 \\
\text{Bad} : & \quad u; x > 1 \\
\end{align*}

\[ x \leq 2 \land x \leq 1 \]

\[ [x \leq 1] \]

Spring School GAMES

Synthesis of Timed Systems

65 / 80
Problems with dense-time control

- $\delta_i$: time in $\ell_2$ during loop $i$
- The controller must ensure: $\sum_{i=1}^{+\infty} \delta_i < 1 - x_0$
Problems with dense-time control

- $x := ]0, 1[$
- $y := 0$

- $\delta_i : \text{time in } \ell_2 \text{ during loop } i$
- The controller must ensure: $\sum_{i=1}^{+\infty} \delta_i < 1 - x_0$

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

[Cassez, Henzinger, Raskin 2002]
Sampling-time control

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

[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

**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:

\[ (x - y) \cdot n \]
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:

- $(x - y) \cdot n$
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:***

![Diagram of encoding](image)

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:

- \((x - y) \cdot n\) if \( x \geq y\)
- \([1 - (y - x)] \cdot n\) 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

\[ x < 1, y < 1 \]

\[ x = 1, y = 1 \]

\[ x := 0, y := 0 \]
Zero testing widget

\[ x < 1, y < 1 \]

\[ x = 1, y = 1 \]

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

Idling widget

\[ x = 1, y < 1 \]

\[ x := 0 \]

\[ x < 1, y = 1 \]

\[ y := 0 \]

\[ x = 1, y = 1 \]

\[ x, y := 0 \]
Normalization and incrementation widget

\[ x < 1, y < 1 \]

\[ x = 1, y < 1; x := 0 \]

\[ y < 1 \]

\[ y = 1; y := 0 \]

\[ x > 1 \]

\[ x \leq 1; y := 0 \]

\[ x = 1, y = 1; x, y := 0 \]
The two-counter machine has an halting computation iff there is a sampled time controller for the above systems.
Notion of implementability

An **implementable** controller $C$

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

[De Wulf, Doyen, Raskin 2004]
Notion of implementability

An **implementable** controller $C$

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

$\implies$ defines a TTS $\llbracket C \rrbracket_\Delta$ (where $\Delta$ is a parameter)

[De Wulf, Doyen, Raskin 2004]
Notion of implementability

An **implementable** controller $C$
- has finite precision (digital clock)
- may delay responses and communications (relaxes synchrony hypothesis)

$\Rightarrow$ defines a TTS $\llbracket C \rrbracket_\Delta$ (where $\Delta$ is a parameter)

**Proposition**

If $\Delta_1 \geq \Delta_2$ and $\llbracket C \rrbracket_{\Delta_1}$ controls $\mathcal{P}$ to avoid bad states, then $\llbracket C \rrbracket_{\Delta_2}$ controls $\mathcal{P}$ to avoid bad states.
Notion of implementability

An **implementable** controller $C$
- has finite precision (digital clock)
- may delay responses and communications (relaxes synchrony hypothesis)

$\Rightarrow$ defines a TTS $\llbracket C \rrbracket_{\Delta}$ (where $\Delta$ is a parameter)

**Proposition**

If $\Delta_1 \geq \Delta_2$ and $\llbracket C \rrbracket_{\Delta_1}$ controls $\mathcal{P}$ to avoid bad states, then $\llbracket C \rrbracket_{\Delta_2}$ controls $\mathcal{P}$ to avoid bad states.

**Proposition**

If $\llbracket C \rrbracket_{\Delta}$ controls $\mathcal{P}$, then $C$ can be implemented on a sufficiently fast hardware.

[De Wulf, Doyen, Raskin 2004]
Notion of implementability

An **implementable** controller $C$

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

$\Rightarrow$ defines a TTS $\llbracket C \rrbracket_\Delta$ (where $\Delta$ is a parameter)

**Proposition**

If $\Delta_1 \geq \Delta_2$ and $\llbracket C \rrbracket_{\Delta_1}$ controls $\mathcal{P}$ to avoid bad states, then $\llbracket C \rrbracket_{\Delta_2}$ controls $\mathcal{P}$ to avoid bad states.

**Proposition**

If $\llbracket C \rrbracket_\Delta$ controls $\mathcal{P}$, then $C$ can be implemented on a sufficiently fast hardware.

$\Rightarrow$ it is sufficient to study the $\Delta$-enlarged semantics
An example: standard semantics

[De Wulf, Doyen, Markey, Raskin 2004]
An example: standard semantics

\[ \begin{align*}
  x &= 1 \\
  y &= 0 \\
  x &\leq 2 \\
  y &\geq 2 \\
  y &= 0
\end{align*} \]

[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 $\Delta > 0$

\[ \text{[De Wulf, Doyen, Markey, Raskin 2004]} \]
An example with $\Delta > 0$

$[\text{De Wulf, Doyen, Markey, Raskin 2004}]$
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 $\Delta > 0$

$\Delta > 0$

$\begin{align*}
\Delta &> 0 \\
\Delta &= \frac{b}{c} \\
x &\in [1-\Delta; 1+\Delta] \\
y &:= 0 \\
x &\leq 2+\Delta \\
y &\geq 2-\Delta \\
y &:= 0
\end{align*}$

[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 $\Delta > 0$

\[ \begin{align*}
\text{Spring School GAMES} & \quad \text{Synthesis of Timed Systems} \\
\end{align*} \]
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 $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

\[ x \in [1-\Delta; 1+\Delta] \]

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

$\begin{align*}
 x \in [1-\Delta; 1+\Delta] \\
 y &:= 0 \\
 x &\leq 2 + \Delta \\
 y &\geq 2 - \Delta \\
 y &:= 0
\end{align*}$

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]

Mathematical expressions:

\[ x \in [1-\Delta; 1+\Delta] \]

\[ y := 0 \]

\[ x \leq 2 + \Delta \]

\[ y \geq 2 - \Delta \]

\[ y := 0 \]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

$\begin{align*}
\forall x \in [1-\Delta; 1+\Delta] \\
y := 0 \\
x := 0 \\
y \geq 2 - \Delta \\
y := 0
\end{align*}$

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

\[ \begin{align*}
\text{Implementability of controllers} \\
\text{Spring School GAMES} \\
\text{De Wulf, Doyen, Markey, Raskin 2004}
\end{align*} \]

\[
\begin{align*}
x \in [1-\Delta; 1+\Delta] \\
y := 0 \\
x \leq 2 + \Delta \\
y \geq 2 - \Delta \\
y := 0
\end{align*}
\]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
An example with $\Delta$ very small

[De Wulf, Doyen, Markey, Raskin 2004]
Deciding implementability

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

$$\llbracket A^\Delta \rrbracket \cap \text{Bad} = \emptyset$$

Theorem

Implementability is decidable for timed automata.

(Using an extension of region automaton construction)

[De Wulf, Doyen, Markey, Raskin 2004]
Outline

1 Preliminaries on timed systems
   • The model of timed automata
   • The region abstraction
   • Everything is not that nice!
   • Symbolic manipulation of timed automata

2 On the semantics of timed games

3 Control synthesis games
   • Framework of these games
   • Simple control objectives
   • Control for external specifications
   • Partial observability

4 Troubles with dense-time control
   • Sampling time control
   • Implementability of controllers

5 Conclusion and current developments
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

- 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
Conclusion and current developments

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

$c_i$: controllable action

$u$: uncontrollable action

\[
\ell_0 \xrightarrow{x \leq 2; c_1; y := 0} \ell_1 \quad \text{cost}(\ell_0) = 5
\]

\[
\ell_1 \xrightarrow{y = 0} \ell_2 \quad \text{cost}(\ell_2) = 10
\]

\[
\ell_2 \xrightarrow{x \geq 2; c_2; \text{cost} = 1} W
\]

\[
\ell_3 \xrightarrow{x \geq 2; c_2; \text{cost} = 7}
\]

\[
\ell_3 \xrightarrow{u} \ell_1
\]

Question: what is the optimal price we can ensure in state $\ell_0$?
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 \(\ell_0\)?

\[5t + 10(2 - t) + 1\]
Synthesis of optimal controllers [LMM02, ABM04, BCFL04, BCFL05]

$c_i$: controllable action
$u$: uncontrollable action

$\ell_0$: $x \leq 2; c_1; y := 0$
$\ell_1$: $y = 0$
$\ell_2$: $x \geq 2; c_2; \text{cost} = 1$
$\ell_3$: $x \geq 2; c_2; \text{cost} = 7$
$W$: $\text{cost}(\ell_2) = 10$
$\text{cost}(\ell_3) = 1$

Question: what is the optimal price we can ensure in state $\ell_0$?

$5t + 10(2 - t) + 1, 5t + (2 - t) + 7$
Synthesis of optimal controllers [LMM02, ABM04, BCFL04, BCFL05]

$c_i$: controllable action

$u$: uncontrollable action

\[
\begin{align*}
\text{cost}(\ell_0) &= 5 \\
\text{cost}(\ell_1) &= \begin{cases} 
5t + 10(2 - t) + 1 & , \quad 5t + (2 - t) + 7 
\end{cases} \\
\text{cost}(\ell_2) &= 10 \\
\text{cost}(\ell_3) &= 7 \\
\text{cost}(W) &= 1 \\
x \geq 2; \ c_2; \ \text{cost} = 1
\end{align*}
\]

Question: what is the optimal price we can ensure in state $\ell_0$?

$$\max \left( 5t + 10(2 - t) + 1, \ 5t + (2 - t) + 7 \right)$$
Conclusion and current developments

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

$c_i$: controllable action
$u$: uncontrollable action

$\ell_0$: cost($\ell_0$) = 5
$\ell_1$: cost($\ell_1$) = 10
$\ell_2$: $x \geq 2$; $c_2$; cost = 1
$\ell_3$: $x \geq 2$; $c_2$; cost = 7
$W$: cost($\ell_3$) = 1

Question: what is the optimal price we can ensure in state $\ell_0$?

$$\inf_{0 \leq t \leq 2} \max (5t + 10(2 - t) + 1, 5t + (2 - t) + 7) = 14 + \frac{1}{3}$$
**Synthesis of optimal controllers** [LMM02, ABM04, BCFL04, BCFL05]

- $c_1$: controllable action
- $u$: uncontrollable action

![Diagram of state transitions](image)

- $\ell_0$: Initial state
  - $x \leq 2; c_1; y := 0$
  - $\text{cost}(\ell_0) = 5$

- $\ell_1$: Transition state
  - $y = 0$

- $\ell_2$: Transition state
  - $x \geq 2; c_2; \text{cost} = 1$
  - $\text{cost}(\ell_2) = 10$

- $\ell_3$: Transition state
  - $x \geq 2; c_2; \text{cost} = 7$
  - $\text{cost}(\ell_3) = 1$

**Question:** what is the optimal price we can ensure in state $\ell_0$?

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

**Strategy:** wait in $\ell_0$, and when $t = \frac{4}{3}$, go to $\ell_1$
Conclusion and current developments

Synthesis of optimal controllers \([LMM02, ABM04, BCFL04, BCFL05]\)

\(c_i\): controllable action
\(u\): uncontrollable action

\[
\ell_0 \xrightarrow{x \leq 2; c_1; y := 0} \ell_1 \xrightarrow{u} \ell_2 \xrightarrow{x \geq 2; c_2; \text{cost} = 1} W
\]

\[
\ell_0 \xrightarrow{\text{cost}(\ell_0) = 5} \ell_1 \xrightarrow{[y = 0]} \ell_2 \xrightarrow{\text{cost}(\ell_2) = 10} \ell_3 \xrightarrow{\text{cost}(\ell_3) = 1} W
\]

\(x \geq 2; c_2; \text{cost} = 7\)

**Question:** what is the optimal price we can ensure in state \(\ell_0\)?

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

\(\rightarrow\text{strategy: wait in } \ell_0, \text{and when } t = \frac{4}{3}, \text{go to } \ell_1\)

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

Bibliography 1


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


Conclusion and current developments

Bibliography II


Conclusion and current developments

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).


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


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