Third Lecture: Basics of Timed Controller Synthesis

Jean-François Raskin
Université Libre de Bruxelles
Belgium

Artist2 Asian Summer School - Shanghai - July 2008
Goals of the talk

• **Introduction** to basic game technics to solve the controller synthesis problem

• **Timed games** and symbolic technics (sketches)

• Show that the **implementability** of controller models is an **important issue**
Goals of the talk

• **Introduction** to basic game technics to solve the controller synthesis problem

• **Timed games** and symbolic technics (sketches)

• Show that the **implementability** of controller **models** is an **important issue**

Give relevant pointers to literature
Context

- Make a model of the environment

**Environment**

- Make clear the control objective: **Bad**

- Make a model of your control strategy: **ControllerMod**

- Verify:

  Does Environment || ControllerMod avoid **Bad**?
Context

• Make a model of the environment

  Environment

• Make clear the control objective:

  Make the synthesis

• Make a model of your control strategy:

  ControllerMod

• Verify:

  Does Environment || ControllerMod avoid Bad?

• Good, but after? Is my controller implementable?
The synthesis problem
The synthesis problem

? || Env |= \phi
The synthesis problem

\[ ? \parallel \text{Env} \models \phi \]

Cont
The synthesis problem

\[ \text{Env} \models \phi \]

Cont

Using algorithmic methods
The synthesis problem

Specialize process A into C such that

\[ A \geq C \text{ and } C \parallel B \models \phi \]

So, C must refine A and control B to enforce \( \phi \)
Basic technics: finite state case
Are transition systems adequate for synthesis?

- For the verification problem, the semantics of processes is usually given by transition systems.
- When we consider the transition system for $A \parallel B$, we lose the information about the components.
Are transition systems adequate for synthesis?

- For the verification problem, the semantics of processes is usually given by **transition systems**
- When we consider the transition system for \( A \parallel B \), we loose the information about the **components**

So, we need richer models where **identities** of processes are explicit: **two-player game structures**
Two-player game structures
Rounded positions belong to Player I
Rounded positions belong to Player 1

Square positions belong to Player 2
A game is played as follows: in each **round**, the game is in a **position**, if the game is in a rounded position, Player I resolves the **choice** for the next state, if the game is in a square position, Player 2 resolves the choice. The game is played for an **infinite number of rounds**.
Play : 0000
Play: 0000 0100
Play: 0000 0100 0101
Play: 0000 0100 0101 1101
Two-player Game Structure

A two-player game structure is a tuple

\[ G = \langle Q_1, Q_2, \iota, \delta \rangle \]

where:

- \( Q_1 \) and \( Q_2 \) are two (finite and) disjoint sets of positions
- \( \iota \in Q_1 \cup Q_2 \) is the initial position of the game
- \( \delta \subseteq (Q_1 \cup Q_2) \times (Q_1 \cup Q_2) \) is the transition relation of the game

We assume that

\[ \forall q \in Q_1 \cup Q_2 : \exists q' \in Q_1 \cup Q_2 : \delta(q, q') \]
Let $G = \langle Q_1, Q_2, \iota, \delta \rangle$,

$\omega = q_0q_1 \ldots q_n \ldots$ is a play in $G$ if
Plays, Prefixes of Plays

Let \( G = \langle Q_1, Q_2, \iota, \delta \rangle \),

\[ w = q_0q_1 \ldots q_n \ldots \text{ is a play in } G \text{ if } \]

\[ \forall i \geq 0 : q_i \in Q_1 \cup Q_2 \]
Plays, Prefixes of Plays

Let $G = \langle Q_1, Q_2, \iota, \delta \rangle$,

\[ w = q_0 q_1 \ldots q_n \ldots \] is a \textbf{play} in $G$ if

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Notations} \\
\hline
\textbf{Let} $w = q_0 q_1 \ldots q_n \ldots$:
\textcolor{red}{\text{w}(i)} denotes position $i$
\textcolor{red}{\text{w}(0, i)} denotes the prefix up to position $i$
\textcolor{red}{\text{last}(w(0, i)) = w(i)}
\hline
\end{tabular}
\end{center}
Plays, Prefixes of Plays

Let $G = \langle Q_1, Q_2, \iota, \delta \rangle$, 

$w = q_0q_1 \ldots q_n \ldots$ is a **play** in $G$ if

1) $w(0) = \iota$

2) $\forall i \geq 0 : \delta(w(i), w(i + 1))$

We denote the set of plays in $G$ by: $\text{Plays}(G)$ and

$\text{PrefPlays}(G) = \{ q_0q_1 \ldots q_n | \exists w \in \text{Plays}(G) \land \forall 1 \leq i \leq n : w(i) = q_i \}$

$\text{PrefPlays}_k(G) = \{ w \in \text{PrefPlays}(G) \land \text{last}(w) \in Q_k \}$
Who is winning?

Play: 0000 0100 0101 1101 ...
Who is winning?

Play: 0000 0100 0101 1101 ...

Is this a **good** or a **bad** play for **Player k**?
Who is winning?

A winning condition (for Player $k$) is a set of plays

$$W \subseteq (Q_1 \cup Q_2)$$
Game

= 

Two-player game structure

+ 

Winning condition for Player $k$
Strategies

Players are playing **according to strategies**.

A **Player $k$ strategy** in $G$ is a function:

$$\lambda : \text{PrefPlays}_k(G) \rightarrow Q_1 \cup Q_2$$

with the restriction that:

$$\forall w \in \text{PrefPlays}_k(G) : \delta(\text{last}(w), \lambda(w))$$
Outcome of a strategy

\( w \) is a possible **outcome** of the Player \( k \) strategy \( \lambda \) if

\[
\forall i \geq 0 : w(i) \in Q_k : w(i + 1) = \lambda(w(0, i))
\]

\( w \) is a play where Player \( k \) plays according to strategy \( \lambda \)
Outcome of a strategy

\( w \) is a possible **outcome** of the Player \( k \) strategy \( \lambda \) if

\[ \forall i \geq 0 : w(i) \in Q_k : w(i + 1) = \lambda(w(0, i)) \]

The set of plays that have this property is denoted

\[ \text{Outcome}_k(G, \lambda) \]
Winning strategy

- Given a pair \((G, W)\)
- We say that Player \(k\) wins the game \((G, W)\) if and only if:

\[
\exists \lambda : \text{Outcome}_k(G, \lambda) \subseteq W
\]
 Winning strategy

• Given a pair \((G, W)\)

• We say that Player \(k\) wins the game \((G, W)\) if and only if:

\[
\exists \lambda : \text{Outcome}_k(G, \lambda) \subseteq W
\]

That is, no matter how the other player resolves his choices, when player \(k\) plays according to \(\lambda\), the resulting play belongs to \(W\). Player \(k\) can force the play to be in \(W\).
Winning strategy

• Given a pair \((G, W)\)

• We say that Player \(k\) wins the game \((G, W)\) if and only if:

\[ \exists \lambda : \text{Outcome}_k(G, \lambda) \subseteq W \]

We say \(\lambda\) that is a **winning strategy** for player \(k\) in the game \((G, W)\)
Winning strategies

= Controllers that enforce winning plays
Winning conditions

- Not all winning conditions are reasonable
- One often assumes that the set of winning plays is a regular set
- We show here how to solve reachability and safety games
Reachability Games
Reachability Game

\[(G, W) \text{ is a reachability game if} \]

\[\exists Q \subseteq Q_1 \cup Q_2 : W = \{w \in \text{Plays}(G) \mid \exists i : w(i) \in Q\}\]

That is \(W\) is a set of plays that reaches the set of locations \(Q\).

\[\text{Reach}(G, Q)\]
A Reachability Game

Does Player I, who owns the rounded positions, have a strategy (against any choices of Player II) to reach the set \{1101, 1111\}?
Safety Games
Safety Game

$(G, W)$ is a **safety game** if

\[
\exists Q \subseteq Q_1 \cup Q_2 : W = \{ w \in \text{Plays}(G) \mid \forall i \geq 0 : w(i) \in Q \}
\]

That is $W$ is the set of plays that stay within given set of positions $Q$.

Safe$(G, Q)$
Does Player I, who owns the rounded positions, have a strategy (against any choices of Player II) to stay within the set of states \( Q \setminus \{1111\} \)?
Symbolic algorithms to solve games
Player $k$ Controllable Predecessors

X is a set of positions

$1\text{CPre}_G(X) = \{ q \in Q_1 \mid \exists q' : \delta(q, q') \land q' \in X \} \cup \{ q \in Q_2 \mid \forall q' : \delta(q, q') : q' \in X \}$

Set of Player I positions where he has a choice of successor that lies in X

Set of Player II positions where all her choices for successors lie in X
Player $k$ Controllable Predecessors

$1CPre_G(X) = \{ q \in Q_1 \mid \exists q' : \delta(q, q') \land q' \in X \} \cup \{ q \in Q_2 \mid \forall q' : \delta(q, q') : q' \in X \}$

Symmetrically

$2CPre_G(X) = \{ q \in Q_2 \mid \exists q' : \delta(q, q') \land q' \in X \} \cup \{ q \in Q_1 \mid \forall q' : \delta(q, q') : q' \in X \}$
Player $k$ Controllable Predecessors

$$1\text{CPre}_G(X) = \{ q \in Q_1 \mid \exists q' : \delta(q, q') \land q' \in X \} \cup \{ q \in Q_2 \mid \forall q' : \delta(q, q') : q' \in X \}$$

Monotonic functions over $\langle 2^{Q_1 \cup Q_2}, \subseteq \rangle$

$$2\text{CPre}_G(X) = \{ q \in Q_2 \mid \exists q' : \delta(q, q') \land q' \in X \} \cup \{ q \in Q_1 \mid \forall q' : \delta(q, q') : q' \in X \}$$
\[ X = \{1000, 0101, 1111\} \]
\[
X = \{1000, 0101, 1111\}
\]
\[
1\text{CPre}(X) = \{0000\} \cup \{0100, 1101\}
\]

Rounded positions, there exists a red successor
$X = \{1000, 0101, 1111\}$

$1\text{CPre}(X) = \{0000\} \cup \{0100, 1101\}$

- Rounded positions, there exists a red successor
- Squared positions, all successors are red
Fixpoints to Solve Games

Reachability game for set $Q$

$$\mu X \cdot Q \cup 1\text{CPre}(X)$$

Safety game for set $Q$

$$\nu X \cdot Q \cap 1\text{CPre}(X)$$
Fixpoint for a safety game

Does Player I, who owns the rounded positions, have a strategy to stay within the set of states $Q \setminus \{1111\}$?
Fixpoint for a safety game

We must compute

$$\nu X \cdot (Q \setminus \{1111\}) \cap 1CPre(X)$$

To do that, we use the Tarski fixpoint theorem.
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1C\text{Pre}(Q) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1C\text{Pre}(Q) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]

\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1CPre(Q) \]
\[ X_1 = (Q \setminus \{1111\}) \cap 1CPre(X_0) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]

\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
\[ X_2 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_1) \]
Fixpoint for a safety game

\[
X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q)
\]
\[
X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0)
\]
\[
X_2 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_1)
\]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
\[ X_2 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_1) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
\[ X_2 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_1) \]
Fixpoint for a safety game

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
\[ X_2 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_1) \]
Fixpoint for a safety game

This is the greatest fixpoint

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
\[ X_2 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_1) = X_1 \]
Fixpoint for a safety game

This is the greatest fixpoint

\[ X_0 = (Q \setminus \{1111\}) \cap 1\text{CPre}(Q) \]
\[ X_1 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_0) \]
\[ X_2 = (Q \setminus \{1111\}) \cap 1\text{CPre}(X_1) = X_1 \]

\[ X_2 \] is exactly the set of positions from which Player I can avoid entering \( \{1111\} \), no matter how Player II behaves.
Theorem

Let \( G = \langle Q_1, Q_2, \nu, \delta \rangle \) be a TGS, let \( \text{Reach}(G, Q) \) be a reachability game defined on \( G \), Player I has a winning strategy for this game iff

\[
\nu \in \mu X \cdot Q \cup 1\text{CPre}(X)
\]
Let $G = \langle Q_1, Q_2, \nu, \delta \rangle$ be a TGS, let Safe$(G, Q)$ be a safety game defined on $G$, Player I has a winning strategy for this game iff

$$\nu \in \nu X \cdot Q \cap 1CPre(X)$$
Some more results

Any finite state game with regular objective can be solved.
Some more results

Any finite state game with regular objective can be solved.

Strategies for safety and reachability games are positional (no need for memory).
Some more results

Any finite state game with regular objective can be solved.

Strategies for safety and reachability games are **positional** (no need for memory).

For more complicated games, like LTL games, finite **memory** is needed.
Some more results

Any finite state game with **regular objective** can be solved.

Strategies for safety and reachability games are **positional** (no need for memory).

For more complicated games, like LTL games, finite **memory** is needed.

**Determinacy theorem:** In positional games (where a position is owned by a player), games are determinate in the following sense:

For any regular set of plays $W$,

Player I has a strategy to win $(G, W)$ iff

Player II does not have a strategy to win $(G, \overline{W})$
From the red states, and only from those states, Player II has a strategy to reach the state 1111
Timed Controller Synthesis
Timed Automata [AD94]
Timed Automata \cite{AD94}

\[ TA = \text{Finite State Automata} + \text{Clocks} \]

State of a TA: \((l, v)\) where \(l\) is a location and \(v\) is a valuation of the clocks.
Timed Automata [AD94]

We need a game version

TA = Finite State Automata + Clocks

State of a TA: \((l, v)\) where \(l\) is a location and \(v\) is a valuation of the clocks.
Simple Timed Game Automata

\[
\langle L_1, L_2, l_0, X, E, Inv \rangle
\]

where:

- \( L_1 \) and \( L_2 \) are locations where Player I, respectively Player II, makes choices.
- \( l_0 \) is the initial location.
Simple Timed Game Automata

\[
\langle L_1, L_2, l_0, X, E, Inv \rangle
\]

where:

- \( X \) is a finite set of clocks
- \( E \subseteq L_1 \cup L_2 \times 2^X \times 2^{R^n} \times L_1 \cup L_2 \), a set of edges
- \( Inv : L_1 \cup L_2 \rightarrow 2^{R^n} \), the invariants labeling locations
Simple Timed Games

As before, the positions of the games are partitioned into positions that belong to Player I and positions that belong to Player II.

Games on STGA are played as follows:

In a Player’s $k$ position, Player $k$ proposes a \textbf{time} $t$ and an \textbf{action} $a$ to be played. This choice must be valid in the sense that it must not violate the \textbf{invariant} and the action $a$ must be \textbf{enabled} after $t$ time units. The game then proceeds to the next position.
Timed Play:

\[(l_0, \langle 0, 0, 0 \rangle) \xrightarrow{0.5}_i (l_1, \langle 0.5, 0, 0.5 \rangle)\]

Player II chooses to **wait 0.5** and then to play **\(i\)**
Timed Play:

\[ (l_0, \langle 0, 0, 0 \rangle) \xrightarrow{i}^{0.5} (l_1, \langle 0.5, 0, 0.5 \rangle) \xrightarrow{a}^{0.5} (l_2, \langle 0, 0.5, 1 \rangle) \]

Player I chooses to **wait 0.5** and then to play **a**
A timed two-player game structure is a tuple \( G = \langle Q_1, Q_2, \iota, \delta_t \rangle \) where:

- \( Q_1 \) and \( Q_2 \) are two disjoint sets of positions
- \( \iota \in Q_1 \cup Q_2 \) is the initial position
- \( \delta_t \subseteq (Q_1 \cup Q_2) \times \mathbb{R} \times (Q_1 \cup Q_2) \) is the timed transition relation

We assume that \( \forall q \in Q_1 \cup Q_2 : \exists t \in \mathbb{R} : \exists q' \in Q_1 \cup Q_2 : \delta_t(q, t, q') \)
From STGA to TTGS

\[ \langle L_1, L_2, l_0, X, E, \text{Inv} \rangle \quad \rightarrow \quad G = \langle Q_1, Q_2, \iota, \delta_t \rangle \]

\[ Q_1 = \{ (l, v) \mid l \in L_1 \land v \models \text{Inv}(l) \} \]

\[ Q_2 = \{ (l, v) \mid l \in L_2 \land v \models \text{Inv}(l) \} \]

\[ \iota = (l_0, 0^{\lvert X \rvert}) \]

\[ \delta((l, v), t, (l', v')) \quad \text{iff} \quad \exists \langle l, r, g, l' \rangle \in E : \]

\[ \forall t' : 0 \leq t' \leq t : v + t \models \text{Inv}(l) \land v + t \models g \land v' = v + t[r := 0] \]
Timed Play

Let \( G = \langle Q_1, Q_2, \iota, \delta_t \rangle \),

\[
  w = q_0 \rightarrow^{t_0} q_1 \rightarrow^{t_1} q_2 \cdots q_n \rightarrow^{t_n} \ldots
\]

is a \textbf{timed play} in \( G \) if

1) \( w(0) = \iota \)
2) \( \forall i \geq 0 : \delta_t( w(i)(q), w(i)(t), w(i+1)(q) ) \)

The set of timed plays of \( G \) is noted \( \text{Plays}(G) \)

\[
\text{PrefPlays}(G) = \{ q_0 \rightarrow^{t_0} \cdots \rightarrow^{t_{n-1}} q_n \mid \exists w \in \text{Plays}(G) \land \forall 0 \leq i \leq n : w(i)(q) = q_i \land w(i)(t) = t_i \}
\]

\[
\text{PrefPlays}_k(G) = \{ w \in \text{PrefPlays}(G) \land \text{last}(w) \in Q_k \}
\]
Timed Strategy

Players are playing according to **timed strategies**.

A Player $k$ strategy in $G$ is a function:

$$\lambda : \text{PrefPlays}_k(G) \rightarrow \mathbb{R} \times Q_1 \cup Q_2$$

with the restriction that:

$$\forall w \in \text{PrefPlays}_k(G) : \delta(\text{last}(w), \lambda(w)(t), \lambda(w)(q))$$
Outcome of a timed strategy

\[ w = q_0 \rightarrow^{t_0} q_1 \rightarrow^{t_1} q_2 \ldots q_n \rightarrow^{t_n} \ldots \]

is a possible outcome of the Player \( k \) timed strategy \( \lambda \) if

\[ \forall i \geq 0 : q_i \in Q_k \rightarrow t_i = \lambda(w(0, i))(t) \land q_{i+1} = \lambda(w(0, i))(q) \]

The set of timed plays that have this property is denoted

\[ \text{Outcome}_k(G, \lambda) \]
Symbolic algorithms to solve timed games
Player $k$ timed controllable predecessors

$$1CPre_G(X) = \{ q \in Q_1 \mid \exists t \in R, q' : \delta_t(q, t, q') \land q' \in X \} \cup \{ q \in Q_2 \mid \forall t \in R, q' : \delta_t(q, t, q') \rightarrow q' \in X \}$$

Set of Player I positions where he has a choice of successor that lies in $X$

Set of Player II positions where all her choices for successors lie in $X$
Player $k$ timed controllable predecessors

$$1\text{CPre}_{G}(X) = \{q \in Q_1 \mid \exists t \in \mathbb{R}, q' : \delta_t(q, t, q') \land q' \in X\} \cup \{q \in Q_2 \mid \forall t \in \mathbb{R}, q' : \delta_t(q, t, q') \rightarrow q' \in X\}$$

Symmetrically

$$2\text{CPre}_{G}(X) = \{q \in Q_2 \mid \exists t \in \mathbb{R}, q' : \delta_t(q, t, q') \land q' \in X\} \cup \{q \in Q_1 \mid \forall t \in \mathbb{R}, q' : \delta_t(q, t, q') \rightarrow q' \in X\}$$
Player $k$ timed controllable predecessors

$1\text{CPre}_G(X) = \{q \in Q_1 \mid \exists t \in \mathbb{R}, q' : \delta_t(q, t, q') \land q' \in X\} \cup \{q \in Q_2 \mid \forall t \in \mathbb{R}, q' : \delta_t(q, t, q') \rightarrow q' \in X\}$

Symmetrically

$2\text{CPre}_G(X) = \{q \in Q_2 \mid \exists t \in \mathbb{R}, q' : \delta_t(q, t, q') \land q' \in X\} \cup \{q \in Q_1 \mid \forall t \in \mathbb{R}, q' : \delta_t(q, t, q') \rightarrow q' \in X\}$

Difficulty: here $X$ ranges over the subsets of an infinite set
Region equivalence
Region equivalence

Finite number of equivalence classes
Region equivalence

All valuations of a region satisfies the same guards and invariants
Region equivalence

Time elapsing and time predecessors preserve regions
Region equivalence

Reset and inverse reset operations preserve regions
1CPre preserves regions

**Theorem.** If $X$ is a union of regions then $1\text{CPre}(X)$ is a union of regions.

**Corollary.** Safety, Reachability and more generally LTL games are decidable on timed game structures generated by timed automata.
Zenoness
Not all timed strategies are reasonable

A timed play \( w = q_0 \rightarrow^{t_0} q_1 \rightarrow^{t_1} q_2 \ldots q_n \rightarrow^{t_n} \ldots \)

is **Zeno** if: \( \exists t \in \mathbb{R} : \sum_{i=0}^{\infty} t_i \leq t \)

Time does not diverge
Not all timed strategies are reasonable

Does Player I have a timed strategy to avoid entering location $l_2$?
Not all timed strategies are reasonable

Consider the following timed strategy for Player I:

Let \( w \in \text{PrefPlay}_1(G) \):

if \( \text{last}(w) = (l_0, v) \) then let \( t = 1 - \frac{1 - v(x)}{2} \) and \( \lambda(w) = (t, (l_1, v(x) + t)) \)

if \( \text{last}(w) = (l_1, v) \) then let \( t = 1 - \frac{1 - v(x)}{2} \) and \( \lambda(w) = (t, (l_0, v(x) + t)) \)
Not all timed strategies are reasonable

When Player I plays this strategy, the only outcome of the games is:

$$(l_0, 0) \rightarrow \frac{1}{2} (l_1, \frac{1}{2}) \rightarrow \frac{1}{4} (l_0, \frac{3}{4}) \rightarrow \frac{1}{8} (l_1, \frac{7}{8}) \ldots$$
Not all timed strategies are reasonable

When Player I plays this strategy, the only outcome of the games is:

\[(l_0, 0) \rightarrow (l_1, 7/8) \rightarrow (l_0, 0) \rightarrow \ldots\]

Clearly, such a strategy cannot be implemented.
Not all timed strategies are reasonable

They are algorithmic solutions to avoid the synthesis of zeno strategies. The correctness of those solutions can be explained using the region graph.
Not all timed strategies are reasonable

They are algorithmic solutions to avoid the synthesis of **zeno strategies**. The correctness of those solutions can be explained using the region graph.

**But Zenoness is not the only problem**
Implementability issues for timed models
Model-based Development

- Make a model of the environment: `Environment`
- Make clear the control objective: `Bad`
- Make a model of your control strategy: `ControllerMod`
- Verify: Does `Environment || ControllerMod` avoid `Bad`?
- Good, but after?
From Correct Models to Correct Implementations

• **Should we verify code?**
  – this may be difficult (too much details)

• **Can we translate model into code?**
  ... there are tools for that ...

• **... and preserve properties?**
  ... good question...
Timed automata are (in general) not implementable (in a formal sense)...

Why?
– Zenoness: 0, 0.5, 0.75, 0.875, ...
– No minimal bound between two transitions: 0, 0.5, 1, 1.75, 2, 2.875, 3, ...
– And more … (robustness)
No Minimal Bound between Two Transitions
It can be controlled

\[ x \leq 2 \quad x = 1 \]
\[ x := 0 \quad x := 0 \]
\[ y := 0 \quad y := 0 \]
\[ z > 0 \quad c \]
\[ y = 1 \]  
\[ z := 0 \]

\[ l_0 \rightarrow a \rightarrow l_1 \]

\[ \delta_i : \text{time in } l_2 \text{ during loop } i \]
\[ \text{the controller must ensure: } \sum_{i=0}^{+\infty} \delta_i < x_0 - y_0 \]
• One can specify instantaneous responses but not implement them.

Not implementable
• *Instantaneous synchronisations between environment and controller are not implementable.*
• Models use continuous clocks and implementations use digital clocks with finite precision.
• My controller strategy may be correct because of
  – ... it is zeno...
  – ... it acts faster and faster?
  – ... it reacts *instantaneously* to events, timeouts,...? (synchrony hypothesis)
  – ... it uses *infinitely* precise clocks?
Give an alternative semantics to timed automata: Almost ASAP semantics.
- enabled transitions of the controller become urgent only after $\Delta$ time units;
- events from the environment are received by the controller within $\Delta$ time units;
- truth values of guards are enlarged by $f(\Delta)$.

where $\Delta$ is a parameter
Definition of the AASAP semantics

Definition 13 [AASAP semantics] Given an ELASTIC controller

\[ A = \langle \text{Loc}, l_0, \text{Var}, \text{Lab}, \text{Edg} \rangle \]

and \( \Delta \in \mathbb{Q}^{\geq 0} \), the AASAP semantics of \( A \), noted \( [A]_{\Delta}^{\text{AASAP}} \) is the STTS

\[ T = \langle S, \tau, \Sigma_{\text{in}}, \Sigma_{\text{out}}, \Sigma_{\tau}, \rightarrow \rangle \]

where:

1. For the discrete transitions, we distinguish five cases:
   - (A4.1) \( \sigma \in \text{Lab}_d \). We have \( \langle l, e, I, d, \sigma, (l', e', I, d) \rangle \in \Delta \) if there exists \( (l', e', R, \sigma) \in \text{Edg} \) such that \( l' = l_{\Delta_{AA}}, e' = e, R = 0 \);
   - (A4.2) \( \sigma \in \text{Lab}_d \). We have \( \langle l, e, I, d, \sigma, (l', e', I, d) \rangle \in \Delta \) if \( l' = l_{\sigma} = 0 \);
   - (A4.3) \( \sigma \in \text{Lab}_c \). We have \( \langle l, e, I, d, \sigma, (l', e', J, 0) \rangle \in \Delta \) if there exists \( (l', e', R, \sigma) \in \text{Edg} \) such that \( l' = l_{\Delta_{AA}}, R = 0 \) and \( l' = l_{R} = 0 \);
   - (A4.4) \( \sigma = e \). We have for any \( l, e, I, d \in S \) \( \langle l, e, I, d, \sigma, (l', e', I, d) \rangle \in \Delta \);
   - (A4.5) \( \sigma = e \). We have for any \( l, e, I, d \in S \) \( \langle l, e, I, d, \sigma, (l', e', I, d) \rangle \in \Delta \).

2. For the continuous transitions:
   - (A4.6) for any \( \Delta \in \mathbb{R}^{\geq 0} \), we have \( \langle l, e, I, d, \Delta \rangle \in \Delta \); the two following conditions are satisfied:
     - for any edge \( (l, e, R, \sigma) \in \text{Edg} \) with \( \sigma \in \text{Lab}_d \cup \text{Lab}_c \), we have that
       \[ 0 \leq e' = e + \Delta \lor \text{TS}(e' + R, \sigma) \leq \Delta \]
     - for any edge \( (l, e, R, \sigma) \in \text{Edg} \) with \( \sigma \in \text{Lab}_d \), we have that
       \[ 0 \leq e' \leq e + \Delta \lor \text{TS}(e' + R, \sigma) \leq \Delta \lor (e + R)(\sigma) \leq \Delta \]
Intuition...

One can specify instantaneous responses but not implement them.

Not implementable

\[
x := 0 \quad \Rightarrow \quad x \leq 0
\]

Solution: allow some delay

\[
x := 0 \quad \Rightarrow \quad x \leq \Delta
\]
Instantaneous synchronisations between environment and controller are not implementable.

Solution: Uncouple event from perception by the controller.
Intuition...

Models use **continuous clocks and implementations use digital clocks with finite precision**

- Classical controller
  - Not implementable

- Solution:
  - Slightly relax the constraints
• The question that we ask when we make verification is no more:

Does Environment $\parallel$ ControllerMod avoid Bad?

• But:

for which values of $\Delta$, does Environment $\parallel$ ControllerMod($\Delta$) avoid Bad?
• Fixed (you know your target platform) :

  Given $\Delta > 0$, does $\text{Environment} \parallel \text{ControllerMod}(\Delta)$ avoid $\text{Bad}$?

• Existence (is my system implementable ?) :

  does there exist $\Delta > 0$ such that $\text{Environment} \parallel \text{ControllerMod}(\Delta)$ avoid $\text{Bad}$?

• Max (how fast must my controller be ?) :

  $\text{Max } \Delta$ such that
Implementability of the AASAP semantics
• AASAP semantics defines a “tube” of strategies instead of a unique strategy in the ASAP semantics.

• This tube can be refined into an implementation while preserving safety properties verified on the AASAP-sem
• We define an “implementation semantics” based on:

Read System Clock
Update Sensor Values
Check all transitions and fire one if possible

• The timed behaviour of this scheme is determined by two values:
  – Time length of a loop : $\Delta_L$
  – Time between two clock ticks : $\Delta_P$
Definition 15 [Program Semantics] Let $A$ be an ELASTIC controller and $\Delta_L$, $\Delta_P \in \mathbb{Q}^0$. We define $\Delta_S = \Delta_L + 2\Delta_P$. The $(\Delta_L, \Delta_P)$ program semantics of $A$, noted $[A]_{\Delta_L, \Delta_P}^{\text{PrG}}$ is the structured timed transition system $T = (S, I, T_{\text{in}}, T_{\text{out}}, \Sigma_\tau, \rightarrow)$ where:

1. $S$ is the set of tuples $(l, r, T, I, u, d, f)$ such that $l \in \text{Loc}$, $r$ is a function from $\text{Var}$ into $\mathbb{R}^+$, $T \in \mathbb{R}^+$, $f$ is a function from $\text{Lab}_a$ into $\mathbb{R}^{+0} \cup \{\bot\}$, $u \in \mathbb{R}^+$, $d \in \mathbb{R}^+$, and $l \in \{T, \bot\}$;
2. $I = (l_0, r_0, I_0, 0, \bot)$ where $r$ is such that for any $x \in \text{Var}$, $r(x) = 0$ if $I$ is such that for any $x \in \text{Lab}_a, r(x) = l$;
3. $\Sigma_a = \text{Lab}_a$, $\Sigma_{\tau} = \text{Lab}_a$, $\Sigma = \text{Lab}_a \cup \{\tau\}$;
4. the transition relation $\rightarrow$ is defined as follows:
   - for the discrete transitions:
     a. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = 0$;
   b. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = \bot$;
   c. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = 0$;
   d. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = \bot$;
   e. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = 0$;
   f. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = \bot$;
   - for the continuous transitions:
     a. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = 0$;
     b. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = \bot$;
     c. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = 0$;
     d. let $\sigma \in \text{Lab}_a$, $(l, r, T, I, u, d, f) \in \sigma$, $(l, r, T, I, u, d, f) \in \rightarrow I(\sigma) = l$ and $I' = I(\sigma) = \bot$;
Proof of “implementability”?

Theorem:

For any timed controller, its AASAP semantics simulates (in the formal sense) its implementation semantics, provided that:

$$\Delta > 3\Delta_L + 4\Delta_P$$

In this case, the implementation is guaranteed to preserve verified properties of the model, that is:

$$\text{Environment} \parallel \text{ControllerMod}(\Delta) \text{ avoid Bad}$$

implies

$$\text{Environment} \parallel \text{ControllerImpl}(\Delta_L, \Delta_P) \text{ avoid Bad}$$
• Faster is better!

For any $\Delta_1, \Delta_2$ such that $\Delta_1 < \Delta_2$:

```
if Environment || ControllerMod($\Delta_2$) avoid Bad
    then
Environment || ControllerMod($\Delta_1$) avoid Bad
```
Properties of the AASAP Semantics

- If $\Delta > 0$, we get for free a proof that strategies:
  - are nonzeno
  - are such that transitions does not need to be taken faster and faster
- If only $\Delta = 0$ guarantees some reachability property, then the control strategy is not implementable
If $\alpha=1$ then the system is safe if and only if $\Delta=0$
If $\alpha=2$ then the system is safe if and only if $\Delta<0.25$
• The **AASAP semantics** can be coded into a parametric timed automata with only one clock compared to the parameter $\Delta \in \mathbb{Q}$.

• Unfortunately, the reachability problem for that class of timed automata is **undecidable**... Direct corollary of [CHR02].

• **Hytech** implements a semi-decision procedure for that problem.

• **Does there exist** $\Delta > 0$ such that
  
  Environment $\parallel\parallel$ ControllerMod($\Delta$) **avoid Bad**?
Fig. 5. Structure of our tool set.
Methodology to develop controllers

1. Models using synchrony hypothesis
   Environment || ControllerMod

2. Check
   Does Environment || ControllerMod(0) avoid Bad?

3. Compute the largest $\Delta_1$ such that
   Environment || ControllerMod($\Delta_1$) avoid Bad

4. if $\Delta_1 > 3 \Delta_L + 4 \Delta_P$

5. Generate code
   This code will enforce the safety property
• Two player games are natural theoretical model to study the synthesis problem

• There exist elegant algorithms to solve general games

• The step to go from a model to a correct implementation needs more investigations
Bibliography
General references on games and synthesis


References on timed and hybrid games


References on implementability issues and robustness


K. Altisen and S. Tripakis. Implementation of timed automata: an issue of semantics or modeling?. In FORMATS'05 (to appear). A previous version of this paper is available as VERIMAG Technical Report TR-2005-12.
