An Introduction to Timed Systems

Patricia Bouyer

LSV, ENS Cachan & CNRS, France
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. Understanding further...

5. Conclusion
**Context:** verification of embedded critical systems

**Time**
- naturally appears in real systems
- appears in properties (for ex. bounded response time)

→ Need of models and specification languages integrating timing aspects
Adding timing informations

- **Untimed case**: sequence of observable events
  - $a$: send message
  - $b$: receive message

\[
\begin{align*}
  a & \ b & a & \ b & a & \ b & a & \ b & \cdots = (a \ b)^\omega
\end{align*}
\]
Adding timing informations

- **Untimed case**: sequence of observable events
  
  - $a$: send message
  - $b$: receive message

  $a\ b\ a\ b\ a\ b\ a\ b\ a\ b\ \cdots = (a\ b)^\omega$

- **Timed case**: sequence of **dated** observable events

  $(a, d_1) (b, d_2) (a, d_3) (b, d_4) (a, d_5) (b, d_6) \cdots$

  - $d_1$: date at which the first $a$ occurs
  - $d_2$: date at which the first $b$ occurs, \ldots
Adding timing informations

- **Untimed case**: sequence of observable events
  - \( a \): send message
  - \( b \): receive message
  \[ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \a \ b \ a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b \a \ b
\]

- **Timed case**: sequence of dated observable events
  \[
  (a, d_1) (b, d_2) (a, d_3) (b, d_4) (a, d_5) (b, d_6) \cdots
  \]

  - \( d_1 \): date at which the first \( a \) occurs
  - \( d_2 \): date at which the first \( b \) occurs, \ldots

- **Discrete-time semantics**: dates are e.g. taken in \( \mathbb{N} \)

Ex: \((a, 1)(b, 3)(c, 4)(a, 6)\)
Adding timing informations

- **Untimed case**: sequence of observable events
  - \( a \): send message
  - \( b \): receive message
  \[
  a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ a \ b \ \cdots = (a \ b)^\omega
  \]

- **Timed case**: sequence of **dated** observable events
  \[
  (a, d_1) \ (b, d_2) \ (a, d_3) \ (b, d_4) \ (a, d_5) \ (b, d_6) \ \cdots
  \]
  - \( d_1 \): date at which the first \( a \) occurs
  - \( d_2 \): date at which the first \( b \) occurs, \ldots
  - **Discrete-time semantics**: dates are e.g. taken in \( \mathbb{N} \)
    - Ex: \( (a, 1)(b, 3)(c, 4)(a, 6) \)
  - **Dense-time semantics**: dates are e.g. taken in \( \mathbb{Q}_+ \), or in \( \mathbb{R}_+ \)
    - Ex: \( (a, 1.28). (b, 3.1). (c, 3.98)(a, 6.13) \)
A case for dense-time

**Time domain:** discrete (e.g. \( \mathbb{N} \)) or dense (e.g. \( \mathbb{Q}_+ \) or \( \mathbb{R}_+ \))

- A compositionality problem with discrete time
- Dense-time is a more general model than discrete time
- But, can we not always discretize?
A digital circuit
Discussion in the context of reachability problems for asynchronous digital circuits

[Alur 91]

[Brzozowski, Seger 1991]
A digital circuit

Discussion in the context of reachability problems for asynchronous digital circuits

Start with \( x=0 \) and \( y=[101] \) (stable configuration)
A digital circuit

Discussion in the context of reachability problems for asynchronous digital circuits

Start with $x=0$ and $y=[101]$ (stable configuration)

The input $x$ changes to 1. The corresponding stable state is $y=[011]$
A digital circuit

Discussion in the context of reachability problems for asynchronous digital circuits

Start with $x=0$ and $y=[101]$ (stable configuration)

The input $x$ changes to 1. The corresponding stable state is $y=[011]$

However, many possible behaviours, e.g.

$$[101] \xrightarrow{y_2}{1.2} [111] \xrightarrow{y_3}{2.5} [110] \xrightarrow{y_1}{2.8} [010] \xrightarrow{y_3}{4.5} [011]$$
A digital circuit

Discussion in the context of reachability problems for asynchronous digital circuits

Start with $x=0$ and $y=[101]$ (stable configuration)
The input $x$ changes to 1. The corresponding stable state is $y=[011]$

However, many possible behaviours, e.g.

\[
\begin{align*}
[101] & \xrightarrow{y_2 \quad 1.2} [111] & \xrightarrow{y_3 \quad 2.5} [110] & \xrightarrow{y_1 \quad 2.8} [010] & \xrightarrow{y_3 \quad 4.5} [011]
\end{align*}
\]

Reachable configurations: $\{[101], [111], [110], [010], [011], [001]\}$
Is discretizing sufficient? An example

This digital circuit is not 1-discretizable.
Is discretizing sufficient? An example

This digital circuit is not 1-discretizable.

Why that?

(initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)
Is discretizing sufficient? An example

[Alur 91]

This digital circuit is not 1-discretizable.

Why that? (initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)

\[
\begin{align*}
[11100000] \xrightarrow{y_1} [01100000] \xrightarrow{y_2} [00100000] \xrightarrow{y_3, y_5} [00001000] \xrightarrow{y_5, y_7} [00000010] \xrightarrow{y_7, y_8} [00000001]
\end{align*}
\]
Is discretizing sufficient? An example

This digital circuit **is not** 1-discretizable.

Why that?

(Initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)

\[
\begin{align*}
[11100000] & \xrightarrow{y_1} [01100000] \xrightarrow{y_2} [00100000] \xrightarrow{y_3, y_5} [00001000] \xrightarrow{y_5, y_7} [00000010] \xrightarrow{y_7, y_8} [00000001] \\
[11100000] & \xrightarrow{y_1, y_2, y_3} [00000000]
\end{align*}
\]
Is discretizing sufficient? An example [Alur 91]

- This digital circuit is not 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)

\[
\begin{align*}
[11100000] & \rightarrow [00000000] \\
[11100000] & \rightarrow [01110000] & \rightarrow [00000000] \\
[11100000] & \rightarrow [01111000] & \rightarrow [00000000]
\end{align*}
\]
Is discretizing sufficient? An example

This digital circuit **is not** 1-discretizable.

Why that? (initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)

\[
\begin{align*}
[11100000] & \xrightarrow{y_1} [01100000] & \xrightarrow{y_2} [00100000] & \xrightarrow{y_3, y_5} [00001000] & \xrightarrow{y_5, y_7} [00000010] & \xrightarrow{y_7, y_8} [00000001] \\
[11100000] & \xrightarrow{y_1, y_2, y_3} [00000000] \\
[11100000] & \xrightarrow{y_1} [01111000] & \xrightarrow{y_2, y_3, y_4, y_5} [00000000] \\
[11100000] & \xrightarrow{y_1, y_2} [00100000] & \xrightarrow{y_3, y_5, y_6} [00001100] & \xrightarrow{y_5, y_6} [00000000] \\
\end{align*}
\]
Is discretizing sufficient? An example

This digital circuit is not 1-discretizable.

Why that? (initially $x = 0$ and $y = [11100000]$, $x$ is set to 1)

\[
\begin{align*}
[11100000] & \xrightarrow{y_1} [01100000] & \xrightarrow{y_2} [00100000] & \xrightarrow{y_3,y_5} [00001000] & \xrightarrow{y_5,y_7} [00000010] & \xrightarrow{y_7,y_8} [00000001] \\
[11100000] & \xrightarrow{y_1,y_2,y_3} [00000000] \\
[11100000] & \xrightarrow{y_1} [01110000] & \xrightarrow{y_2,y_3,y_4,y_5} [00000000] \\
[11100000] & \xrightarrow{y_1,y_2} [00100000] & \xrightarrow{y_3,y_5,y_6} [00001100] & \xrightarrow{y_5,y_6} [00000000]
\end{align*}
\]
Is discretizing sufficient?

**Theorem** [Brzozowski Seger 1991]

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).
Is discretizing sufficient?

**Theorem** [Brzozowski Seger 1991]

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

**Claim**

Finding a correct granularity is as difficult as computing the set of reachable states in dense-time.
Is discretizing sufficient?

**Theorem** [Brzozowski Seger 1991]
For every \( k \geq 1 \), there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity \( \frac{1}{k} \)).

**Claim**
Finding a correct granularity is as difficult as computing the set of reachable states in dense-time.

**Further counter-example**
There exist systems for which no granularity exists. (see later)
Is discretizing sufficient?

**Theorem** [Brzozowski Seger 1991]

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

**Claim**

Finding a correct granularity is as difficult as computing the set of reachable states in dense-time.

**Further counter-example**

There exist systems for which no granularity exists. (see later)

Hence, we better consider a dense-time domain!
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. Understanding further...

5. Conclusion
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 \( \sim \in \{<, \leq, =, \geq, >\} \)
Timed automata (example)

$x, y$ : clocks

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

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

$x, y : \text{clocks}$

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

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

$x, y : \text{clocks}$

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

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

(clock) valuation
Timed automata (example)

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

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

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

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

(clock) valuation

\[ \begin{array}{c|c|c}
\ell_0 & 0 & 4.1 \\
\ell_1 & 4.1 & 0 \\
\ell_2 & 1.4 & 1.4 \\
\end{array} \]

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

• $A = (\Sigma, L, X, \rightarrow)$ 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 $\exists \ell \xrightarrow{g,a,r} \ell' \in A$ 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 $d \in T$
Discrete vs dense-time semantics

The timed automaton model
Discrete vs dense-time semantics

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

\[ y < 1, \quad b, \quad y := 0 \]

Dense-time:

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

- **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:** \[ L_{\text{discrete}} = \emptyset \]
Discrete vs dense-time semantics

- **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:** \( L_{\text{discrete}} = \emptyset \)

\[ x = 1, \ a, \ x := 0 \]
\[ b, \ y := 0 \]
\[ y < 1, \ b, \ y := 0 \]
Classical verification problems

- reachability of a control state
- \( S \sim S' \): bisimulation, etc...
- \( L(S) \subseteq L(S') \): language inclusion
- \( S \models \varphi \) for some formula \( \varphi \): model-checking
- \( S \parallel A_T + \) reachability: testing automata
- ...
The train crossing example

Train$_i$ with $i = 1, 2, ...$

- **Far**
  - $10 < x_i < 20$, Exit!

- **Before, $x_i < 30$**
  - $20 < x_i < 30$, $a$, $x_i := 0$
  - $App!$, $x_i := 0$

- **On, $x_i < 20$**
The train crossing example

The gate:

- **Open**
  - GoDown?, $H_g := 0$
  - $H_g < 10$, a

- **Lowering**, $H_g < 10$
  - $H_g < 10$, a

- **Raising**, $H_g < 10$
  - $H_g < 10$, a

- **Close**
  - GoUp?, $H_g := 0$
The train crossing example (3)

The controller:

\(c_1, x_c \leq 20\)

\(H_c = 20, \text{ GoUp!}\)

\(H_c = 0, \text{ App?} \rightarrow \text{Exit?}\)

\(c_0\)

\(H_c \leq 10, \text{ GoDown!}\)

\(c_2, x_c \leq 10\)

\(H_c = 0, \text{ App?} \rightarrow \text{Exit?}\)
The train crossing example

We use the synchronization function $f$:

<table>
<thead>
<tr>
<th>$\text{Train}_1$</th>
<th>$\text{Train}_2$</th>
<th>Gate</th>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{App!}$</td>
<td>.</td>
<td>.</td>
<td>$\text{App?}$</td>
</tr>
<tr>
<td>.</td>
<td>$\text{App!}$</td>
<td>.</td>
<td>$\text{App?}$</td>
</tr>
<tr>
<td>$\text{Exit!}$</td>
<td>.</td>
<td>.</td>
<td>$\text{Exit?}$</td>
</tr>
<tr>
<td>.</td>
<td>$\text{Exit!}$</td>
<td>.</td>
<td>$\text{Exit?}$</td>
</tr>
<tr>
<td>$a$</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>$a$</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>$a$</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>$\text{GoUp?}$</td>
<td>$\text{GoUp!}$</td>
<td>$\text{GoUp}$</td>
</tr>
<tr>
<td>.</td>
<td>$\text{GoDown?}$</td>
<td>$\text{GoDown!}$</td>
<td>$\text{GoDown}$</td>
</tr>
</tbody>
</table>

To define the parallel composition $(\text{Train}_1 \parallel \text{Train}_2 \parallel \text{Gate} \parallel \text{Controller})$

**NB:** The parallel composition does not add expressive power!
The train crossing example

Some properties one could check:

- Is the gate closed when a train crosses the road?
The train crossing example

Some properties one could check:

- Is the gate closed when a train crosses the road?

$$\mathbf{AG}(\text{train.On} \Rightarrow \text{gate.Close})$$
The train crossing example

Some properties one could check:

- Is the gate closed when a train crosses the road?
  \[ \text{AG} (\text{train.On } \Rightarrow \text{gate.Close}) \]

- Is the gate always closed for less than 5 minutes?
The train crossing example (5)

Some properties one could check:

- Is the gate closed when a train crosses the road?
  \[ \text{AG}(\text{train.On} \Rightarrow \text{gate.Close}) \]

- Is the gate always closed for less than 5 minutes?
  \[ \neg \text{EF}(\text{gate.Close} \land \text{E}(\text{gate.Close} \ \text{U}_{>5 \text{ min}} \ \neg \text{gate.Close})) \]
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. Understanding further...

5. Conclusion
Verification

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

- reachability properties  
- basic liveness properties

  (final states)

  (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 for finite-state systems cannot be applied
Verification

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

- **Problem:** the set of configurations is infinite
  - classical methods for finite-state systems cannot be applied

- **Positive key point:** variables (clocks) increase at the same speed
Verification

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

- **Problem:** the set of configurations is infinite
  - ➔ classical methods for finite-state systems cannot be applied

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

**Theorem** [Alur, Dill 1990]

The emptiness problem for timed automata is decidable. It is \( \text{PSPACE-complete} \).
Verification

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

- **Problem:** the set of configurations is infinite
  - ➜ classical methods for finite-state systems cannot be applied

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

**Theorem** [Alur, Dill 1990]

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

- “compatibility” between regions and constraints
- “compatibility” between regions and time elapsing
The region abstraction

"compatibility" between regions and constraints

"compatibility" between regions and time elapsing

→ a time-abstract bisimulation property
The region abstraction

Equivalence of finite index

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

- “compatibility” between regions and constraints
- “compatibility” between regions and time elapsing

$\rightarrow$ a time-abstract bisimulation property
The region abstraction

Equivalence of finite index

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

- “compatibility” between regions and constraints
- “compatibility” between regions and time elapsing
  ➔ a time-abstract bisimulation property
Time-abstract bisimulation
Time-abstract bisimulation
Time-abstract bisimulation

∀a → b

∃a → b

∀d > 0 δ(d) → b
Time-abstract bisimulation

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

\[ \delta(d) \quad \delta(d') \]
Time-abstract bisimulation

$\forall a \rightarrow \exists a$

$\exists d > 0 \rightarrow \forall d > 0$

$\exists d' > 0 \rightarrow \forall d' > 0$

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

∀ \bullet \xrightarrow{a} \bullet

∃ \bullet \xrightarrow{a} \bullet

∀ d > 0 \bullet \xrightarrow{\delta(d)} \bullet

∃ d' > 0 \bullet \xrightarrow{\delta(d')} \bullet

(\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.
Time-abstract bisimulation

\[ \forall \begin{array}{c} \bullet \end{array} \xrightarrow{a} \begin{array}{c} \bullet \end{array} \]

\[ \exists \begin{array}{c} \bullet \end{array} \xrightarrow{a} \begin{array}{c} \bullet \end{array} \]

\[ \forall d > 0 \begin{array}{c} \bullet \end{array} \xrightarrow{\delta(d)} \begin{array}{c} \bullet \end{array} \]

\[ \exists d' > 0 \begin{array}{c} \bullet \end{array} \xrightarrow{\delta(d')} \begin{array}{c} \bullet \end{array} \]

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

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

with \( v_i \in R_i \) for all \( i \).
Region automaton $\equiv$ finite bisimulation quotient
Region automaton $\equiv$ finite bisimulation quotient

timed automaton $\otimes$ region abstraction

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

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

- $R'' \subseteq g$
- $[C \leftarrow 0]R'' \subseteq R'$
Region automaton $\equiv$ finite bisimulation quotient

**timed automaton $\otimes$ region abstraction**

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

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

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

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

where $\text{UNTIME}((a_1, t_1)(a_2, t_2)\ldots) = a_1a_2\ldots$
An example [AD 90’s]
Consequence of region automata construction

Region automata:

correct finite (and exponential) abstraction for checking reachability/Büchi-like properties.
Consequence of region automata construction

Region automata:
correct finite (and exponential) abstraction for checking reachability/ Büchi-like properties.

However...
everything can not be reduced to finite automata...
A model not far from undecidability

Some bad news...

- Language universality is **undecidable**  
  [Alur, Dill 1990]
- Language inclusion is **undecidable**  
  [Alur, Dill 1990]
- Complementability is **undecidable**  
  [Tripakis 2003, Finkel 2006]

...
A model not far from undecidability

Some bad news...

- Language universality is **undecidable** [Alur, Dill 1990]
- Language inclusion is **undecidable** [Alur, Dill 1990]
- Complementability is **undecidable** [Tripakis 2003, Finkel 2006]
- ...

An example of non-determinizable/non-complementable timed aut.:

![Timed automaton diagram](image-url)
A model not far from undecidability

Some bad news...

- Language universality is undecidable [Alur,Dill 1990]
- Language inclusion is undecidable [Alur,Dill 1990]
- Complementability is undecidable [Tripakis 2003, Finkel 2006]
- ...

An example of non-determinizable/non-complementable timed aut.: [Alur,Madhusudan 2004]

UNTIME \( \bar{L} \cap \{(a^*b^*, \tau) \mid \text{all } a's \text{ happen before 1 and no two } a's \text{ simultaneously}\} \) is not regular (exercise!)
Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:
  - various extensions of timed automata
    - [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
    - [Bouyer, Dufourd, Fleury, Petit 2004] · · ·
Partial conclusion

This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:

- various extensions of timed automata
  [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
  [Bouyer, Dufourd, Fleury, Petit 2004] · · ·

- model-checking of branching-time properties (TCTL, timed μ-calculus)
  [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]
Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:
  - various extensions of timed automata
    - [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
    - [Bouyer, Dufourd, Fleury, Petit 2004] · · ·
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    - [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]
  - weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
    - [Bouyer, Larsen, Markey, Rasmussen 2006] [Bouyer, Larsen, Markey 2007]
Partial conclusion

This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:

- various extensions of timed automata
  - [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
  - [Bouyer, Dufourd, Fleury, Petit 2004] · · ·

- model-checking of branching-time properties (TCTL, timed $\mu$-calculus)
  - [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]

- weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
  - [Bouyer, Larsen, Markey, Rasmussen 2006] [Bouyer, Larsen, Markey 2007]

- o-minimal hybrid systems
  - [Lafferriere, Pappas, Sastry 2000] [Brihaye 2005]
Partial conclusion

This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:

- various extensions of timed automata
  [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
  [Bouyer, Dufourd, Fleury, Petit 2004] · · ·

- model-checking of branching-time properties (TCTL, timed $\mu$-calculus)
  [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]

- weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
  [Bouyer, Larsen, Markey, Rasmussen 2006] [Bouyer, Larsen, Markey 2007]

- o-minimal hybrid systems
  [Lafferriere, Pappas, Sastry 2000] [Brihaye 2005]

· · ·
Partial conclusion

This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:

- various extensions of timed automata
  - [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
  - [Bouyer, Dufourd, Fleury, Petit 2004] · · ·

- model-checking of branching-time properties (TCTL, timed $\mu$-calculus)
  - [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]

- weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
  - [Bouyer, Larsen, Markey, Rasmussen 2006] [Bouyer, Larsen, Markey 2007]

- o-minimal hybrid systems
  - [Lafferriere, Pappas, Sastry 2000] [Brihaye 2005]
  - · · ·

- Note however that it might be hard to prove there is a finite bisimulation quotient!
Partial conclusion

This idea of a finite bisimulation quotient has been applied to many “timed” or “hybrid” systems:

- various extensions of timed automata
  - [Bérard, Diekert, Gastin, Petit 1998] [Choffrut, Goldwurm 2000]
  - [Bouyer, Dufourd, Fleury, Petit 2004] · · ·

- model-checking of branching-time properties (TCTL, timed $\mu$-calculus)
  - [Alur, Courcoubetis, Dill 1993] [Laroussinie, Larsen, Weise 1995]

- weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
  - [Bouyer, Larsen, Markey, Rasmussen 2006] [Bouyer, Larsen, Markey 2007]

- o-minimal hybrid systems
  - [Lafferriere, Pappas, Sastry 2000] [Brihaye 2005]
  · · ·

- Note however that it might be hard to prove there is a finite bisimulation quotient!

- Note that in practice, the region automaton is not constructed, and symbolic technics based on zones are used.
1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. Understanding further...

5. Conclusion
The example of alternating timed automata

**Alternating timed automata** \( \equiv \text{ATA} \)

The example of alternating timed automata

**Alternating timed automata** ≡ ATA


**Example**

“No two a’s are separated by 1 unit of time”

\[
\begin{align*}
\ell_0, a, \text{true} & \quad \mapsto \quad \ell_0 \land (x := 0, \ell_1) \\
\ell_1, a, x \neq 1 & \quad \mapsto \quad \ell_1 \\
\ell_1, a, x = 1 & \quad \mapsto \quad \ell_2 \\
\ell_2, a, \text{true} & \quad \mapsto \quad \ell_2
\end{align*}
\]

\{ \ell_0 \text{ initial state} \}
\{ \ell_0, \ell_1 \text{ final states} \}
\{ \ell_2 \text{ losing state} \}
The example of alternating timed automata

**Alternating timed automata \( \equiv \) ATA**


**Example**

“No two \( a \)’s are separated by 1 unit of time”

\[
\begin{align*}
\ell_0, a, \text{true} &\quad \mapsto \quad \ell_0 \land (x := 0, \ell_1) \\
\ell_1, a, x \neq 1 &\quad \mapsto \quad \ell_1 \\
\ell_1, a, x = 1 &\quad \mapsto \quad \ell_2 \\
\ell_2, a, \text{true} &\quad \mapsto \quad \ell_2
\end{align*}
\]

\( \ell_0 \) initial state

\( \ell_0, \ell_1 \) final states

\( \ell_2 \) losing state
nice closure properties
nice closure properties

universality is as difficult as reachability
nice closure properties

universality is as difficult as reachability

more expressive than timed automata

[Lasota, Walukiewicz 2005]
nice closure properties ➔ universality is as difficult as reachability

more expressive than timed automata

Theorem

- Emptiness of ATA is undecidable.
- Emptiness of one-clock ATA is decidable, but non-primitive recursive.
- Emptiness for Büchi properties of one-clock ATA is undecidable.
- Emptiness of one-clock ATA with \(\varepsilon\)-transitions is undecidable.

[Lasota, Walukiewicz 2005]
nice closure properties

⇒ universality is as difficult as reachability

more expressive than timed automata

Theorem

- Emptiness of ATA is undecidable.
- Emptiness of one-clock ATA is decidable, but non-primitive recursive.
- Emptiness for Büchi properties of one-clock ATA is undecidable.
- Emptiness of one-clock ATA with ε-transitions is undecidable.

Lower bound: simulation of a lossy channel system...

[Lasota, Walukiewicz 2005]

[Schneebelen 2002]
Example

\[ x := 0 \]

\[ x \neq 1, a \]

\[ a \]

\[ x = 1, a \]
Example

Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)
Example

Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)

\(\{ (\ell_0, 0) \} \)
Example

Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)

\[
\begin{align*}
\{ (\ell_0, 0) \} \\
\downarrow \\
\{ (\ell_0, .3), (\ell_1, 0) \}
\end{align*}
\]
Example

Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)

\[
\begin{align*}
\{ (\ell_0, 0) \} \\
\downarrow \\
\{ (\ell_0, .3), (\ell_1, 0) \} \\
\downarrow \\
\{ (\ell_0, .8), (\ell_1, 0), (\ell_1, .5) \}
\end{align*}
\]
Example

Execution over timed word \((a, 0.3)(a, 0.8)(a, 1.4)(a, 1.8)(a, 2)\)

\[
\begin{align*}
\{ (\ell_0, 0) \} \\
\downarrow \\
\{ (\ell_0, 0.3), (\ell_1, 0) \} \\
\downarrow \\
\{ (\ell_0, 0.8), (\ell_1, 0), (\ell_1, 0.5) \} \\
\downarrow \\
\{ (\ell_0, 1.4), (\ell_1, 0), (\ell_1, 0.6), (\ell_1, 1.1) \}
\end{align*}
\]
Example

 Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)

\[
\begin{align*}
\{ (l_0, 0) \} \\
\downarrow \\
\{ (l_0, .3), (l_1, 0) \} \\
\downarrow \\
\{ (l_0, .8), (l_1, 0), (l_1, .5) \} \\
\downarrow \\
\{ (l_0, 1.4), (l_1, 0), (l_1, .6), (l_1, 1.1) \} \\
\downarrow \\
\{ (l_0, 1.8), (l_1, 0), (l_1, .4), (l_2, 1), (l_1, 1.5) \}
\end{align*}
\]
Example

Execution over timed word \((a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)\)

```
{ \(\ell_0, 0\) }
↓
{ \(\ell_0, .3\), \(\ell_1, 0\) }
↓
{ \(\ell_0, .8\), \(\ell_1, 0\), \(\ell_1, .5\) }
↓
{ \(\ell_0, 1.4\), \(\ell_1, 0\), \(\ell_1, .6\), \(\ell_1, 1.1\) }
↓
{ \(\ell_0, 1.8\), \(\ell_1, 0\), \(\ell_1, .4\), \(\ell_2, 1\), \(\ell_1, 1.5\) }
↓
{ \(\ell_0, 2\), \(\ell_1, 0\), \(\ell_1, .2\), \(\ell_1, .6\), \(\ell_2, 1.2\), \(\ell_1, 1.7\) }
```
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)

\[
\begin{align*}
(\ell, 0) & \quad (\ell, 0.3) & \quad (\ell, 1.2) & \quad (\ell, 2.3) & \quad (\ell', 0.4) & \quad (\ell', 1) & \quad (\ell', 0.8)
\end{align*}
\]
An abstraction

A configuration = a finite set of pairs $(\ell, x)$

$\{(\ell, 0), (\ell', 1)\}$

0.0
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)

\[
(\ell, 0) \quad (\ell, 0.3) \quad (\ell, 1.2) \quad (\ell, 2.3) \quad (\ell', 0.4) \quad (\ell', 1) \quad (\ell', 0.8)
\]

\[
\{(\ell, 0), (\ell', 1)\} \quad \{(\ell, 1)\}
\]

0.0 \quad 0.2
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)

\[
\begin{align*}
(l, 0) & \quad (l, 0.3) & \quad (l, 1.2) & \quad (l, 2.3) & \quad (l', 0.4) & \quad (l', 1) & \quad (l', 0.8) \\
\{(l, 0), (l', 1)\} & \quad \{l, 1\} & \quad \{(l, 0), (l, 2)\} \\
0.0 & \quad 0.2 & \quad 0.3
\end{align*}
\]
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)

\[
\begin{align*}
(\ell, 0) & \quad (\ell, 0.3) & \quad (\ell, 1.2) & \quad (\ell, 2.3) & \quad (\ell', 0.4) & \quad (\ell', 1) & \quad (\ell', 0.8) \\
\{((\ell, 0), (\ell', 1))\} & \quad \{((\ell, 1))\} & \quad \{((\ell, 0), (\ell, 2))\} & \quad \{((\ell', 0))\} \\
0.0 & \quad 0.2 & \quad 0.3 & \quad 0.4
\end{align*}
\]
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)
An abstraction

A configuration = a finite set of pairs $(\ell, x)$

\[
(\ell, 0) \quad (\ell, 0.3) \quad (\ell, 1.2) \quad (\ell, 2.3) \quad (\ell', 0.4) \quad (\ell', 1) \quad (\ell', 0.8)
\]

\[
\{(\ell, 0), (\ell', 1)\} \quad \{(\ell, 1)\} \quad \{(\ell, 0), (\ell, 2)\} \quad \{(\ell', 0)\} \quad \{(\ell', 0)\}
\]

\[
0.0 \quad 0.2 \quad 0.3 \quad 0.4 \quad 0.8
\]
An abstraction

A configuration = a finite set of pairs \((\ell, x)\)

Abstracted into:

\[
\{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \cdot \{(\ell', 0)\}
\]
Abstract transition system
Abstract transition system

{(ℓ, 0), (ℓ', 1)} · {(ℓ, 1)} · {(ℓ, 0), (ℓ, 2)} · {ℓ', 0} · {ℓ', 0}

Time successors:
Abstract transition system

{((\ell, 0), (\ell', 1))} \cdot \{((\ell, 1))\} \cdot \{((\ell, 0), (\ell, 2))\} \cdot \{((\ell', 0))\} \cdot \{((\ell', 0))\}

Time successors:

{((\ell', 1))} \cdot \{((\ell, 0), (\ell', 1))\} \cdot \{((\ell, 1))\} \cdot \{((\ell, 0), (\ell, 2))\} \cdot \{((\ell', 0))\}
Abstract transition system

\[\{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \cdot \{(\ell', 0)\}\]

Time successors:

\[\{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\}\]

\[\{(\ell', 1)\} \cdot \{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\}\]
Abstract transition system

{(ℓ, 0), (ℓ', 1)} · {(ℓ, 1)} · {(ℓ, 0), (ℓ, 2)} · {(ℓ', 0)} · {(ℓ', 0)}

Time successors:

{(ℓ', 1)} · {(ℓ, 0), (ℓ', 1)} · {(ℓ, 1)} · {(ℓ, 0), (ℓ, 2)} · {(ℓ', 0)}

{(ℓ', 1)} · {(ℓ', 1)} · {(ℓ, 0), (ℓ', 1)} · {(ℓ, 1)} · {(ℓ, 0), (ℓ, 2)}

{(ℓ, 1), (ℓ, 3)} · {(ℓ', 1)} · {(ℓ', 1)} · {(ℓ, 0), (ℓ', 1)} · {(ℓ, 1)}
Abstract transition system

\[
\{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \cdot \{(\ell', 0)\}
\]

Time successors:

\[
\{(\ell', 1)\} \cdot \{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\}
\]

\[
\{(\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\}
\]

\[
\{(\ell, 1), (\ell, 3)\} \cdot \{(\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\}
\]

\[
\{(\ell, 2)\} \cdot \{(\ell, 1), (\ell, 3)\} \cdot \{(\ell', 1)\} \cdot \{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\}
\]
Abstract transition system

\[
\{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \cdot \{(\ell', 0)\}
\]

Time successors:

\[
\{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\}
\]

\[
\{(\ell', 1)\} \cdot \{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\}
\]

\[
\{(\ell, 1), (\ell, 3)\} \cdot \{(\ell', 1)\} \cdot \{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\}
\]

\[
\{(\ell, 2)\} \cdot \{(\ell, 1), (\ell, 3)\} \cdot \{(\ell', 1)\} \cdot \{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\}
\]

\[
\{(\ell, 1), (\ell', 2)\} \cdot \{(\ell, 2)\} \cdot \{(\ell, 1), (\ell, 3)\} \cdot \{(\ell', 1)\} \cdot \{(\ell', 1)\}
\]
Abstract transition system

\{ (\ell, 0), (\ell', 1) \} \cdot \{ (\ell, 1) \} \cdot \{ (\ell, 0), (\ell, 2) \} \cdot \{ (\ell', 0) \} \cdot \{ (\ell', 0) \}

Time successors:

\{ (\ell', 1) \} \cdot \{ (\ell, 0), (\ell', 1) \} \cdot \{ (\ell, 1) \} \cdot \{ (\ell, 0), (\ell, 2) \} \cdot \{ (\ell', 0) \}

\{ (\ell', 1) \} \cdot \{ (\ell', 1) \} \cdot \{ (\ell, 0), (\ell', 1) \} \cdot \{ (\ell, 1) \} \cdot \{ (\ell, 0), (\ell, 2) \}

\{ (\ell, 1), (\ell, 3) \} \cdot \{ (\ell', 1) \} \cdot \{ (\ell', 1) \} \cdot \{ (\ell, 0), (\ell', 1) \} \cdot \{ (\ell, 1) \}

\{ (\ell, 2) \} \cdot \{ (\ell, 1), (\ell, 3) \} \cdot \{ (\ell', 1) \} \cdot \{ (\ell', 1) \} \cdot \{ (\ell, 0), (\ell', 1) \}

\{ (\ell, 1), (\ell', 2) \} \cdot \{ (\ell, 2) \} \cdot \{ (\ell, 1), (\ell, 3) \} \cdot \{ (\ell', 1) \} \cdot \{ (\ell', 1) \}

Transition $\ell \xrightarrow{x > 2, x := 0} \ell''$: 
Abstract transition system

Time successors:

Transition $\ell \xrightarrow{x > 2, x := 0} \ell''$: 
What can we do with that abstract transition system?

Correctness?
What can we do with that abstract transition system?

**Correctness?**

The previous abstraction is (almost) a *time-abstract bisimulation*. 
What can we do with that abstract transition system?

Correctness?
The previous abstraction is (almost) a time-abstract bisimulation.

Termination?
What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a *time-abstract bisimulation*.

Termination?

❌ possibly infinitely many abstract configurations
What can we do with that abstract transition system?

Correctness?
The previous abstraction is (almost) a **time-abstract bisimulation**.

Termination?

😊 possibly infinitely many abstract configurations

😊 there is a well-quasi ordering on the set of abstract configurations! (subword relation ⊑)
What can we do with that abstract transition system?

Correctness?
The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

😊 possibly infinitely many abstract configurations
😊 there is a well-quasi ordering on the set of abstract configurations!
  (subword relation \( \sqsubseteq \))
+ downward compatibility:

\[
(\gamma_1 \sqsubseteq \gamma'_1 \text{ and } \gamma'_1 \leadsto \gamma_2) \Rightarrow (\gamma_1 \leadsto^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma'_2)
\]
What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

😊 possibly infinitely many abstract configurations
😊 there is a well-quasi ordering on the set of abstract configurations!
   (subword relation \(\sqsubseteq\))
   + downward compatibility:
     \[(\gamma_1 \sqsubseteq \gamma'_1 \text{ and } \gamma'_1 \sim \gamma'_2) \Rightarrow (\gamma_1 \sim^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma'_2)\]
   + downward-closed objective (all states are accepting)
What can we do with that abstract transition system?

Correctness?
The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

😊 possibly infinitely many abstract configurations

😢 there is a well-quasi ordering on the set of abstract configurations!
   (subword relation $\sqsubseteq$)
   + downward compatibility:
     $(\gamma_1 \sqsubseteq \gamma'_1 \text{ and } \gamma'_1 \sim \gamma'_2) \Rightarrow (\gamma_1 \sim^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma'_2)$
   + downward-closed objective (all states are accepting)

Recipe learned on Monday:
What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

😊 possibly infinitely many abstract configurations
😊 there is a well-quasi ordering on the set of abstract configurations!
   (subword relation ⊒)
   + downward compatibility:
     \[(\gamma_1 \sqsubseteq \gamma'_1 \text{ and } \gamma'_1 \sim \gamma'_2) \Rightarrow (\gamma_1 \sim^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma'_2)\]
   + downward-closed objective (all states are accepting)

Recipe learned on Monday:

\[(\text{Higman's lemma } + \text{ Koenig's lemma}) \Rightarrow \text{termination}\]
What can we do with that abstract transition system?

Correctness?
The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

😊 possibly infinitely many abstract configurations
😊 there is a well-quasi ordering on the set of abstract configurations!
   (subword relation $\sqsubseteq$)
   + downward compatibility:
     $$(\gamma_1 \sqsubseteq \gamma'_1 \text{ and } \gamma'_1 \sim \gamma'_2) \Rightarrow (\gamma_1 \leadsto^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma'_2)$$
   + downward-closed objective (all states are accepting)

Recipe learned on Monday:

$$(\text{Higman's lemma } + \text{Koenig's lemma}) \Rightarrow \text{termination}$$

Alternative

The abstract transition system can be simulated by a kind of FIFO channel machine.
A digression on timed automata
A digression on timed automata

\[ x, y \in r_0, \{y\} < \{x\} \]

\[ (y, r_0) \cdot (x, r_0) \]
A digression on timed automata

$x \in r_1, y \in r_0, \{x\} < \{y\}$
A digression on timed automata

\[ x, y \in r_1, \{y\} < \{x\} \]

\[ (y, r_1) \cdot (x, r_1) \]
The classical region automaton can be simulated by a channel machine (with a single bounded channel).
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata

[Abdulla, Jonsson 1998]
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata
  [Abdulla, Jonsson 1998]
- timed Petri nets
  [Abdulla, Nylén 2001]
Partial conclusion

Similar technics apply to:
- networks of single-clock timed automata [Abdulla, Jonsson 1998]
- timed Petri nets [Abdulla, Nylén 2001]
- MTL model checking [Ouaknine, Worrell 2005, 2007]
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata  
  [Abdulla, Jonsson 1998]
- timed Petri nets  
  [Abdulla, Nylén 2001]
- MTL model checking  
  [Ouaknine, Worrell 2005, 2007]
- coFlatMTL model checking  
  [Bouyer, Markey, Ouaknine, Worrell 2007]
  (using channel machines with a bounded number of cycles)
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata [Abdulla, Jonsson 1998]
- timed Petri nets [Abdulla, Nylén 2001]
- MTL model checking [Ouaknine, Worrell 2005, 2007]
- coFlatMTL model checking [Bouyer, Markey, Ouaknine, Worrell 2007]
  (using channel machines with a bounded number of cycles)
- single-clock automata inclusion checking [Ouaknine, Worrel 2004]
Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata [Abdulla, Jonsson 1998]
- timed Petri nets [Abdulla, Nylén 2001]
- MTL model checking [Ouaknine, Worrell 2005, 2007]
- coFlatMTL model checking (using channel machines with a bounded number of cycles) [Bouyer, Markey, Ouaknine, Worrell 2007]
- single-clock automata inclusion checking [Ouaknine, Worrell 2004]
- ...
Outline

1. Introduction

2. The timed automaton model

3. Timed automata, decidability issues

4. Understanding further...

5. Conclusion
Conclusion

- Justification of the dense-time semantics
Conclusion

- Justification of the dense-time semantics
- Two main technics for proving decidability of real-time systems
  - Are missing: initialized rectangular automata
Conclusion

- Justification of the dense-time semantics
- Two main technics for proving decidability of real-time systems
  - Are missing: initialized rectangular automata

Some current streams of research in timed systems:
- quantitative model-checking
- real-time logics
- robustness, implementability issues
- timed games
- …