Vérification de systèmes temporisés et hybrides

Patricia Bouyer

LSV – CNRS & ENS de Cachan
Model-checking

Does the system satisfy the property?
Model-checking

Does the system satisfy the property?

Modelling

Model-checking Algorithm
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
A case for dense-time

**Time domain:** discrete (e.g. $N$) or dense (e.g. $Q^+$)
- A compositionality problem with discrete time
- Dense-time is a more general model than discrete time
- Not all timed systems can be discretized...

![Diagram](image)
A case for dense-time

**Time domain:** discrete (e.g. $N$) or dense (e.g. $Q^+$)
- A compositionality problem with discrete time
- Dense-time is a more general model than discrete time
- Not all timed systems can be discretized...

$$L_{\text{dense}} = \{(a b)^\omega, \tau) \mid \forall i, \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1}\}$$
A case for dense-time

**Time domain:** discrete (e.g. $N$) or dense (e.g. $Q^+$)
- A compositionality problem with discrete time
- Dense-time is a more general model than discrete time
- Not all timed systems can be discretized...

\[ x = 1, \ a, \ x := 0 \quad \text{b, y := 0} \]
\[ x = 1, \ a, \ x := 0 \quad \text{y < 1, b, y := 0} \]

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

- **Discrete-time:** \[ L_{discrete} = \emptyset \]
The model of timed automata

1. The model of timed automata

2. Decidability issues

3. Some extensions of the model

4. Implementation of timed automata

5. Concluding remarks
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)

$x, y : \text{clocks}$

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

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

$x, y : \text{clocks}$

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

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

$x, y : \text{clocks}$

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

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

(clock) valuation
Timed automata (example)

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

\(x \leq 5, \ a, \ y := 0\)

\(y > 1, \ b, \ x := 0\)

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

(clock) valuation

\(\rightarrow\) 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 $d \in T$
The train crossing example

Train_i with \( i = 1, 2, \ldots \)
The train crossing example

The gate:

- Open \(\xrightarrow{\text{GoDown?}, \ H_g := 0} \) Lowering, \(H_g < 10\)
- Raising, \(H_g < 10\), a \(\xrightarrow{H_g < 10, \ a} \) Close
- Lowering, \(H_g < 10\) \(\xrightarrow{H_g < 10, \ a} \) Close
- GoUp?, \(H_g := 0\)
The controller:

- $c_1$, $x_c \leq 20$
  - Exit?, $H_c := 0$
  - $H_c = 20$, GoUp!

- $c_0$
  - Exit?, $H_c := 0$
  - App?, $H_c := 0$
  - $H_c \leq 10$, GoDown!

- $c_2$, $x_c \leq 10$
  - Exit?
  - App?
The train crossing example

We use the synchronization function $f$:

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

\[ AG(\text{train.On} \Rightarrow \text{gate.Close}) \]
Some properties one could check:

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

\[ AG(\text{train.On} \Rightarrow \text{gate.Close}) \]

- Is the gate always closed for less than 5 minutes?
Some properties one could check:

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

  \[ AG(\text{train.On} \Rightarrow \text{gate.Close}) \]

- Is the gate always closed for less than 5 minutes?

  \[ \neg EF(\text{gate.Close} \land (\text{gate.Close} U_{>5\text{min}} \neg\text{gate.Close})) \]
Decidability issues

Outline

1. The model of timed automata

2. Decidability issues

3. Some extensions of the model

4. Implementation of timed automata

5. Concluding remarks
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 \( \varphi \) (e.g. in a timed extension of classical temporal logics): model-checking
- \( S \parallel A_T + \) reachability: testing automata
- ...
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 cannot be applied
Verification

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

- Problem: the set of configurations is infinite
  ➔ classical methods can not 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 can not be applied

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

**Theorem:** The emptiness problem for timed automata is decidable. It is PSPACE-complete.  
[Alur & Dill 1990’s]
Verification

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

- **Problem:** the set of configurations is infinite
  - classical methods can not be applied

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

**Theorem:** The emptiness problem for timed automata is decidable. It is PSPACE-complete. [Alur & Dill 1990’s]

**Note:** This is also the case for the discrete semantics.
Verification

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

- **Problem:** the set of configurations is infinite  
  ➔ classical methods can not be applied

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

**Theorem:** The emptiness problem for timed automata is decidable. It is PSPACE-complete.  
[Alur & Dill 1990’s]

**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

$\Rightarrow$ a bisimulation property
The region abstraction

Equivalence of finite index

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

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

→ a bisimulation property
The region automaton

**timed automaton ⊗ region abstraction**

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

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

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

→ **time-abstract bisimulation**

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

where \( \text{UNTIME}((a_1, t_1)(a_2, t_2)\ldots) = a_1 a_2 \ldots \)
An example [AD 90’s]
Time-abstract bisimulation
Time-abstract bisimulation
**Time-abstract bisimulation**

\[ \forall a \quad \exists \delta(d) \]

\[ \forall d > 0 \quad \exists \delta(d) \]
Time-abstract bisimulation
Time-abstract bisimulation

\[
\begin{align*}
\forall & 
\begin{array}{c}
\text{a} \\
\end{array} \\
\exists & 
\begin{array}{c}
\text{a} \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
\forall d > 0 & 
\begin{array}{c}
\delta(d) \\
\end{array} \\
\exists d' > 0 & 
\begin{array}{c}
\delta(d') \\
\end{array}
\end{align*}
\]

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

\[ \forall \ \exists \quad \delta(d) \quad \delta(d') \]

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

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

with \( v_i \in R_i \) for all \( i \).
Time-abstract bisimulation

\[
\forall \quad \exists \quad (\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \xrightarrow{a_2, t_2} (\ell_2, v_2) \xrightarrow{a_3, t_3} \ldots \\
(\ell_0, R_0) \xrightarrow{a_1} (\ell_1, R_1) \xrightarrow{a_2} (\ell_2, R_2) \xrightarrow{a_3} \ldots
\]

with \( v_i \in R_i \) for all \( i \).
Time-abstract bisimulation

\[ \forall a \quad \exists \delta(d) \]

\[ \forall d > 0 \quad \exists \delta(d') \]

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

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

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

**Remark:** Real-time properties can not be checked with a time-abstract bisimulation. An extended construction needs to be used.
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 can not be reduced to finite automata...
A model not far from undecidability

- 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

- Universality is undecidable
- Inclusion is undecidable
- Determinizability is undecidable
- Complementability is undecidable
- ... [Alur & Dill 90's]
  [Alur & Dill 90's]
  [Tripakis 2003]
  [Tripakis 2003]

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

![Diagram of timed automaton](image)

a, x := 0  a, x = 1, a
A model not far from undecidability

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

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

\[ \text{UNTIE} \left( \bar{L} \cap \{(a^*b^*,\tau) \mid \text{all } a\text{'s happen before } 1 \text{ and no two } a\text{'s simultaneously} \} \right) \] is not regular (exercise!)

[Alur & Dill 90's]  
[Alur & Dill 90's]  
[Tripakis 2003]  
[Tripakis 2003]  
[Alur, Madhusudan 2004]
Partial conclusion

→ a timed model interesting for verification purposes

Numerous works have been (and are) devoted to:

- the “theoretical” comprehension of timed automata (cf [Asarin 2004])
- extensions of the model (to ease modelling)
  - expressiveness
  - analyzability
- algorithmic problems and implementation
Outline

1. The model of timed automata
2. Decidability issues
3. Some extensions of the model
4. Implementation of timed automata
5. Concluding remarks
Role of diagonal constraints

\[ x - y \sim c \quad \text{and} \quad x \sim c \]

- **Decidability**: yes, using the region abstraction

- **Expressiveness**: no additional expressive power
Role of diagonal constraints (cont.)

\[ c \text{ is positive} \]

\[ x := 0 \quad y := 0 \quad x - y \leq c \]

\[ \text{copy where } x - y \leq c \]

\[ x := 0 \quad y := 0 \quad x \leq c \]

\[ x := 0 \quad x > c \quad y := 0 \]

\[ \text{copy where } x - y > c \]

\[ \rightarrow \text{ proof in [Bérard, Diekert, Gastin, Petit 1998]} \]
Role of diagonal constraints (cont.)

$c$ is positive

- $x := 0$
- $y := 0$
- $x - y \leq c$

- Proof in [Bérard, Diekert, Gastin, Petit 1998]
- Exponential blowup unavoidable in general

- [Bouyer, Chevalier 2005]

- $x := 0$
- $y := 0$
- $x \leq c$

- Copy where $x - y \leq c$

- $x := 0$
- $x > c$
- $y := 0$

- Copy where $x - y > c$
Adding silent actions

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

[Bérard, Diekert, Gastin, Petit 1998]

- **Decidability**: yes
  (actions have no influence on region automaton construction)

- **Expressiveness**: strictly more expressive!

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

\[ x = 1, \ \varepsilon, \ x := 0 \]
Adding constraints of the form $x + y \sim c$

- **Decidability:** for two clocks, **decidable** using the abstraction

- for four clocks (or more), **undecidable**!

- **Expressiveness:** more expressive! (even using two clocks)

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

\[\{(a^n, t_1 \ldots t_n) \mid n \geq 1 \text{ and } t_i = 1 - \frac{1}{2^i}\}\]
Adding constraints of the form $x + y \sim c$

- **Two clocks**: **decidable** using the abstraction

![Graph showing lines indicating constraints for two clocks]

- **Four clocks (or more)**: **undecidable**!
Adding constraints of the form $x + y \sim c$

- **Two clocks**: decidable using the abstraction

- **Three clocks**: open question!

- **Four clocks (or more)**: undecidable!
Linear hybrid automata

- A finite control structure + a set $X$ of dynamical variables

- A transition is of the form:

$$
\text{Act}_\ell \xrightarrow{g, \ a, \ \alpha} \text{Act}_{\ell'}
$$

- $g$ is a linear constraint on variables

- $\alpha$ is a jump condition, i.e. an affine update of the form $X' = A.X + B$

- in each state, an activity function assigning a slope to each variable (for each $x \in X$, $\text{Act}(x) \in [\ell, u]$)
The gas burner may leak.

- each time a leakage is detected, it is repaired or stopped in less than 1s
- two leakages are separated by at least 30s

Is it possible that the gas burner leaks during a time greater than $\frac{1}{20}$ of the global time after the 60 first minutes?

$$AG(y \geq 60 \Rightarrow 20t \leq y)$$
What about decidability?

⇒ almost everything is undecidable
[Henzinger, Kopke, Puri, Varaiya 98]

**Theorem.** The class of LHA with clocks and only one variable having possibly two slopes $k_1 \neq k_2$ is undecidable.

**Theorem.** The class of *stopwatch* automata is undecidable.

One of the “largest” classes of LHA which are decidable is the class of initialized rectangular automata.
Outline

1. The model of timed automata

2. Decidability issues

3. Some extensions of the model

4. Implementation of timed automata

5. Concluding remarks
The region automaton is not used for implementation:

- suffers from a combinatorics explosion
  (the number of regions is exponential in the number of clocks)
- no really adapted data structure
Notice

The region automaton is not used for implementation:

- suffers from a combinatorics explosion
  (the number of regions is exponential in the number of clocks)
- no really adapted data structure

Algorithms for “minimizing” the region automaton have been proposed...

[Alur & Co 1992] [Tripakis, Yovine 2001]
Notice

The region automaton is not used for implementation:
- suffers from a combinatorics explosion
  (the number of regions is exponential in the number of clocks)
- no really adapted data structure

Algorithms for “minimizing” the region automaton have been proposed...
[Alur & Co 1992] [Tripakis, Yovine 2001]

...but on-the-fly technics are preferred.
Reachability analysis

- **forward analysis algorithm:**
  compute the successors of initial configurations
Reachability analysis

- **forward analysis algorithm:**
  compute the successors of initial configurations
Reachability analysis

- **forward analysis algorithm:**
  compute the successors of initial configurations

- **backward analysis algorithm:**
  compute the predecessors of final configurations
Reachability analysis

- **forward analysis algorithm:**
  compute the successors of initial configurations

- **backward analysis algorithm:**
  compute the predecessors of final configurations
Symbolic representations

**Linear hybrid automata:** polyhedra, defined by inequations of the form

\[ a_1x_1 + a_2x_2 + \ldots + a_nx_n \nleq b \]

\[ \rightarrow \text{tool HyTech} \]

http://www-cad.eecs.berkeley.edu:80/~tah/HyTech/

**Timed automata:** particular kind of polyhedra called zones

\[ x_i - x_j \nleq c \quad \text{or} \quad x_i \nleq c \]
Note on the backward analysis of TA

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

\[ \left[ C \leftarrow 0 \right]^{-1}(Z \cap (C = 0)) \cap g \]

\[ Z \]
Note on the backward analysis of TA

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

\[ Z \]

\[ \left[ C \leftarrow 0 \right]^{-1}(Z \cap (C = 0)) \cap g \]
Note on the backward analysis of TA

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

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

\[ Z \]

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \]
Note on the backward analysis of TA

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

\[ Z \]

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \]
Note on the backward analysis of TA

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

\[ g, a, C := 0 \]
Note on the backward analysis of TA

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

\[ [C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \]

The exact backward computation terminates and is correct!
Forward analysis of timed automata

zones $Z$ 

$[C \leftarrow 0](\overline{Z} \cap g)$
Forward analysis of timed automata

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

zones \( Z \)

\[ [C \leftarrow 0](\bar{Z} \cap g) \]
Forward analysis of timed automata

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

zones \( Z \)

\[ [C \leftarrow 0](\overrightarrow{Z} \cap g) \]
Forward analysis of timed automata

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

\[ \ell \rightarrow \ell' \]

zones \[ Z \]

\[ [C \leftarrow 0](\overrightarrow{Z} \cap g) \]

\[ Z \]

\[ \overrightarrow{Z} \]

\[ \overrightarrow{Z} \cap g \]
Forward analysis of timed automata

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

zones

\[ Z \]

\[ [C \leftarrow 0](\vec{Z} \cap g) \]

\[ Z \]

\[ \vec{Z} \]

\[ \vec{Z} \cap g \]

\[ [y \leftarrow 0](\vec{Z} \cap g) \]
Forward analysis of timed automata

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

 zones \[ Z \]

\[ [C \leftarrow 0](\overrightarrow{Z} \cap g) \]

\[ \overrightarrow{Z} \cap g \]

\[ [y \leftarrow 0](\overrightarrow{Z} \cap g) \]

\[ \rightarrow \text{a termination problem} \]
Non termination of the forward analysis

\[ y := 0, \]
\[ x := 0 \]
\[ x \geq 1 \land y = 1, \]
\[ y := 0 \]
Non termination of the forward analysis

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

\[ x \geq 1 \land y = 1, \]
\[ y := 0 \]
Non termination of the forward analysis

\[ y := 0, \]
\[ x := 0 \]
\[ x \geq 1 \land y = 1, \]
\[ y := 0 \]
Non termination of the forward analysis

\[ y := 0, \]
\[ x := 0 \]
\[ x \geq 1 \land y = 1, \]
\[ y := 0 \]
Non termination of the forward analysis

$y := 0,$
$x := 0$

$x \geq 1 \land y = 1,$
$y := 0$

$\Rightarrow$ an infinite number of steps...
“Solutions” to this problem

(f.ex. in [Larsen, Pettersson, Yi 1997] or in [Daws, Tripakis 1998])

- **inclusion checking**: if $Z \subseteq Z'$ and $Z'$ already considered, then we don’t need to consider $Z$

  $\Rightarrow$ correct w.r.t. reachability
“Solutions” to this problem

(f.ex. in [Larsen, Pettersson, Yi 1997] or in [Daws, Tripakis 1998])

- **inclusion checking**: if $Z \subseteq Z'$ and $Z'$ already considered, then we don’t need to consider $Z$

  \[ \Rightarrow \text{correct w.r.t. reachability} \]

- **activity**: eliminate redundant clocks [Daws, Yovine 1996]

  \[ \Rightarrow \text{correct w.r.t. reachability} \]

\[
q \xrightarrow{g.a,C:=0} q' \implies \text{Act}(q) = \text{clocks}(g) \cup (\text{Act}(q') \setminus C)
\]

\[
\ldots
\]
"Solutions" to this problem (cont.)

- **convex-hull approximation**: if \( Z \) and \( Z' \) are computed then we overapproximate using "\( Z \uplus Z' \)".

  ➔ "semi-correct" w.r.t. reachability
“Solutions” to this problem (cont.)

- **convex-hull approximation**: if $Z$ and $Z'$ are computed then we overapproximate using “$Z \sqcup Z'$”.
  - “semi-correct” w.r.t. reachability

- **extrapolation**, a widening operator on zones
The DBM data structure

DBM (Difference Bounded Matrice) data structure

\[(x_1 \geq 3) \land (x_2 \leq 5) \land (x_1 - x_2 \leq 4)\]

\[
\begin{pmatrix}
  x_0 & x_1 & x_2 \\
  +\infty & -3 & +\infty \\
  +\infty & +\infty & 4 \\
  5 & +\infty & +\infty \\
\end{pmatrix}
\]
The DBM data structure

DBM (Difference Bounded Matrice) data structure

[Berthomieu, Menasche 1983] [Dill 1989]

\[(x_1 \geq 3) \land (x_2 \leq 5) \land (x_1 - x_2 \leq 4)\]

\[
\begin{pmatrix}
    x_0 & x_1 & x_2 \\
    x_0 & +\infty & -3 & +\infty \\
    x_1 & +\infty & +\infty & 4 \\
    x_2 & 5 & +\infty & +\infty \\
\end{pmatrix}
\]

- Existence of a normal form
The DBM data structure

DBM (Difference Bounded Matrice) data structure

[Berthomieu, Menasche 1983] [Dill 1989]

\[(x_1 \geq 3) \land (x_2 \leq 5) \land (x_1 - x_2 \leq 4)\]

\[
\begin{pmatrix}
  x_0 & x_1 & x_2 \\
  x_0 & +\infty & -3 & +\infty \\
  x_1 & +\infty & +\infty & 4 \\
  x_2 & 5 & +\infty & +\infty \\
\end{pmatrix}
\]

- Existence of a normal form

- All previous operations on zones can be computed using DBMs
The extrapolation operator

Fix an integer $k$  

\[
\begin{pmatrix}
* & > k & * \\
* & * & * \\
< -k & * & *
\end{pmatrix}
\sim
\begin{pmatrix}
* & +\infty & * \\
* & * & * \\
-k & * & *
\end{pmatrix}
\]

- "intuitively", erase non-relevant constraints

→ ensures termination
The extrapolation operator

Fix an integer $k$ ($\ast$ represents an integer between $-k$ and $+k$)

\[
\begin{pmatrix}
\ast & \textcolor{green}{> k} & \ast \\
\ast & \ast & \ast \\
\textcolor{red}{< -k} & \ast & \ast
\end{pmatrix}
\sim
\begin{pmatrix}
\ast & \textcolor{red}{+ \infty} & \ast \\
\ast & \ast & \ast \\
\textcolor{green}{-k} & \ast & \ast
\end{pmatrix}
\]

• “intuitively”, erase non-relevant constraints

\[\rightarrow \text{ ensures termination}\]
The extrapolation operator

Fix an integer $k$  

\[
\begin{pmatrix}
* & > k & * \\
* & * & * \\
< -k & * & *
\end{pmatrix}
\sim
\begin{pmatrix}
* & +\infty & * \\
* & * & * \\
- k & * & *
\end{pmatrix}
\]

• "intuitively", erase non-relevant constraints

→ ensures termination
Classical algorithm, focus on correctness

Take $k$ the maximal constant appearing in the constraints of the automaton.
Classical algorithm, focus on correctness

Take $k$ the maximal constant appearing in the constraints of the automaton.

**Theorem:** This algorithm is correct for diagonal-free timed automata.
Classical algorithm, focus on correctness

Take $k$ the maximal constant appearing in the constraints of the automaton.

**Theorem:** This algorithm is correct for diagonal-free timed automata.

**However**, this theorem does not extend to timed automata using diagonal clock constraints...

- Implemented in numerous tools:
  - **Uppaal**, http://www.uppaal.com/
  - **Kronos**, http://www-verimag.imag.fr/TEMPORISE/kronos/
  - ...

- Successfully used on many real-life examples since ten years.
Outline

1. The model of timed automata
2. Decidability issues
3. Some extensions of the model
4. Implementation of timed automata
5. Concluding remarks
Discussion on complexity


<table>
<thead>
<tr>
<th></th>
<th>Kripke structures S</th>
<th>Timed automaton A</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reachability</strong></td>
<td>NLOGSPACE-complete</td>
<td></td>
</tr>
<tr>
<td><strong>CTL/TCTL</strong></td>
<td>P-complete</td>
<td></td>
</tr>
<tr>
<td><strong>AF-(\mu)-calc./L_{\mu,\nu}</strong></td>
<td>P-complete</td>
<td></td>
</tr>
<tr>
<td><strong>full (\mu)-calc./L_{\mu,\nu}^+</strong></td>
<td>NP (\cap) co-NP</td>
<td></td>
</tr>
</tbody>
</table>
# Discussion on complexity


<table>
<thead>
<tr>
<th></th>
<th>Kripke structures S</th>
<th>Timed automaton A or ( S_1 \parallel \ldots \parallel S_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reachability</td>
<td>NLOGSPACE-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>CTL/TCTL</td>
<td>P-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>AF-( \mu )-calc./( L_{\mu,\nu} )</td>
<td>P-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>full ( \mu )-calc./( L^{+}_{\mu,\nu} )</td>
<td>NP ( \cap ) co-NP</td>
<td>EXPTIME-complete</td>
</tr>
</tbody>
</table>

Timing constraints induce a complexity blowup!
Discussion on complexity


|                          | Kripke structures $S$ | Timed automaton $A$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>or $(S_1 \parallel \ldots \parallel S_n)$</td>
</tr>
<tr>
<td>Reachability</td>
<td>NLOGSPACE-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>CTL/TCTL</td>
<td>P-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>AF-$\mu$-calc./$L_{\mu,\nu}$</td>
<td>P-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>full $\mu$-calc./$L_{\mu,\nu}^+$</td>
<td>NP $\cap$ co-NP</td>
<td>EXPTIME-complete</td>
</tr>
</tbody>
</table>

Timing constraints induce a complexity blowup!

From a complexity point of view, adding clocks $=$ adding components!
## Discussion on complexity


|                  | Kripke structures $S$ | Timed automaton $A$
|------------------|------------------------|---------------------------
| Reachability     | NLOGSPACE-complete     | PSPACE-complete            |
| CTL/TCTL         | P-complete             | PSPACE-complete            |
| AF-$\mu$-calc./$L_{\mu,\nu}$ | P-complete             | EXPTIME-complete           |
| full $\mu$-calc./$L^{+}_{\mu,\nu}$ | NP $\cap$ co-NP      | EXPTIME-complete           |

Timing constraints induce a complexity blowup!

From a complexity point of view, adding clocks = adding components!
State explosion problem

- due to parallel composition
- due to timing constraints
State explosion problem

- due to parallel composition
- due to timing constraints

From a complexity point of view:

no double complexity gap!
State explosion problem

- due to parallel composition
- due to timing constraints

**From a complexity point of view:**

no double complexity gap!
State explosion problem

- due to parallel composition
- due to timing constraints

From a complexity point of view:

no double complexity gap!

In practice:

- BDD-like techniques try to avoid discrete state explosion problem in untimed systems
  ➔ SMV verifies very large systems
State explosion problem

- due to parallel composition
- due to timing constraints

From a complexity point of view:

no double complexity gap!

In practice:

- BDD-like techniques try to avoid discrete state explosion problem in untimed systems  \( \rightarrow \) SMV verifies very large systems

- **Timed systems**: problems to deal with both explosions. Much smaller systems can be analyzed in practice.
State explosion problem

- due to parallel composition
- due to timing constraints

From a complexity point of view:

no double complexity gap!

In practice:

- BDD-like techniques try to avoid discrete state explosion problem in untimed systems ➔ SMV verifies very large systems

- **Timed systems:** problems to deal with both explosions. Much smaller systems can be analyzed in practice.
Decidability is quite well understood.

Needs to understand better the geometry of the reachable state space.
Conclusion & Further Work

- Decidability is quite well understood.
- Needs to understand better the **geometry** of the reachable state space.
- Some other current challenges:
  - controller synthesis
  - implementability issues (program synthesis)
  - optimal computations
  - ...
Conclusion & Further Work

- Decidability is quite well understood.
- Needs to understand better the geometry of the reachable state space.
- Some other current challenges:
  - controller synthesis
  - implementability issues (program synthesis)
  - optimal computations
  - ...

To be continued...
**The two-counter machine**

**Definition.** A two-counter machine is a finite set of instructions over two counters ($x$ and $y$):

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

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

**Theorem.** [Minsky 67] The halting problem for two counter machines is undecidable.
Undecidability proof

\[ \rightarrow \text{simulation of } \begin{align*} \bullet & \text{ decrementation of a counter} \\ \bullet & \text{ incrementation of a counter} \end{align*} \]

We will use 4 clocks:
\[ \bullet u, \text{ “tic” clock (each time unit)} \]
\[ \bullet x_0, x_1, x_2 : \text{ reference clocks for the two counters} \]

“\(x_i\) reference for \(c\)” \(\equiv\) “the last time \(x_i\) has been reset is the last time action \(c\) has been performed”

[Bérard, Dufourd 2000]
Undecidability proof (cont.)

- **Incrementation of counter $c$:**

  $x_0 \leq 2, \ u + x_2 = 1, \ c, \ x_2 := 0$

  $x_2 := 0$

  $u = 1, \ *, \ u := 0$

  $x_0 > 2, \ c, \ x_2 := 0$

  $u + x_2 = 1$

  ref for $c$ is $x_0$

- **Decrementation of counter $c$:**

  $x_0 < 2, \ u + x_2 = 1, \ c, \ x_2 := 0$

  $x_2 := 0$

  $u = 1, \ *, \ u := 0$

  $x_0 = 2, \ c, \ x_2 := 0$

  $u + x_2 = 1$

  ref for $c$ is $x_2$

  $u = 1, \ x_0 = 2, \ *, \ u := 0, \ x_2 := 0$
Note on the backward analysis (cont.)

If $\mathcal{A}$ is a timed automaton, we construct its corresponding set of regions.

Because of the bisimulation property, we get that:

“Every set of valuations which is computed along the backward computation is a finite union of regions”
If $\mathcal{A}$ is a timed automaton, we construct its corresponding set of regions.

Because of the bisimulation property, we get that:

“Every set of valuations which is computed along the backward computation is a finite union of regions”

Let $R$ be a region. Assume:

- $v \in \bar{R}$ (for ex. $v + t \in R$)
- $v' \equiv_{\text{reg.}} v$

There exists $t'$ s.t. $v' + t' \equiv_{\text{reg.}} v + t$, which implies that $v' + t' \in R$ and thus $v' \in \bar{R}$. 
Note on the backward analysis (cont.)

If $\mathcal{A}$ is a timed automaton, we construct its corresponding set of regions.

Because of the bisimulation property, we get that:

“Every set of valuations which is computed along the backward computation is a finite union of regions”

**But**, the backward computation is not so nice, when also dealing with integer variables...

$$i := j \cdot k + \ell \cdot m$$
A problematic automaton

\[ x_3 \leq 3 \quad \Rightarrow \quad x_1, x_3 := 0 \]

\[ x_2 = 3 \quad \Rightarrow \quad x_2 := 0 \]

\[ x_1 = 2, \quad x_1 := 0 \]

\[ x_2 = 2, \quad x_2 := 0 \]

\[ x_1 = 3 \quad \Rightarrow \quad x_1 := 0 \]

\[ x_2 = 2 \quad \Rightarrow \quad x_2 := 0 \]

\[ x_2 - x_1 > 2 \quad \Rightarrow \quad \text{Error} \]

\[ x_4 - x_3 < 2 \]

The loop
A problematic automaton

\[
\begin{align*}
  x_3 &\leq 3 \\
  x_1, x_3 &:= 0 \\
  x_2 &:= 0 \\
  x_1 &:= 0 \\
  x_2 &:= 0 \\
  x_1 &:= 2, x_1 := 0 \\
  x_2 &:= 2, x_2 := 0 \\
  x_1 &:= 2 \\
  &\text{The loop}
\end{align*}
\]

Error

\[
\begin{align*}
  x_2 - x_1 &> 2 \\
  x_4 - x_3 &< 2 \\
  x_1 &:= 3 \\
  x_1 &:= 0 \\
  x_2 &:= 2 \\
  x_2 &:= 0
\end{align*}
\]

\[
\begin{align*}
  \nu(x_1) &= 0 \\
  \nu(x_2) &= d \\
  \nu(x_3) &= 2\alpha + 5 \\
  \nu(x_4) &= 2\alpha + 5 + d
\end{align*}
\]
A problematic automaton

\[ x_3 \leq 3 \]
\[ x_1, x_3 := 0 \]
\[ x_1, x_3 := 0 \]
\[ x_2 = 3 \]
\[ x_2 := 0 \]
\[ x_2 := 0 \]
\[ x_1 = 2 \]
\[ x_1 := 0 \]
\[ x_2 = 2 \]
\[ x_2 := 0 \]
\[ x_1 = 2, x_1 := 0 \]
\[ x_2 = 2, x_2 := 0 \]

The loop

\[ x_2 - x_1 > 2 \]
\[ x_4 - x_3 < 2 \]
\[ x_1 = 3 \]
\[ x_1 := 0 \]
\[ x_2 = 2 \]
\[ x_2 := 0 \]
\[ x_1 := 0 \]
\[ x_2 := 0 \]

Error

\[ \nu(x_1) = 0 \]
\[ \nu(x_2) = d \]
\[ \nu(x_3) = 2\alpha + 5 \]
\[ \nu(x_4) = 2\alpha + 5 + d \]
The problematic zone

\[ [1; 3] \quad \rightarrow \quad [2\alpha + 5] \quad \rightarrow \quad [1; 3] \]

\[ [2\alpha + 2; 2\alpha + 4] \quad \rightarrow \quad [2\alpha + 6; 2\alpha + 8] \]

implies

\[ x_1 - x_2 = x_3 - x_4. \]
The problematic zone

If $\alpha$ is sufficiently large, after extrapolation:

implies $x_1 - x_2 = x_3 - x_4$.

does not imply $x_1 - x_2 = x_3 - x_4$. 

$> k$