Vérification de systèmes temporisés et hybrides

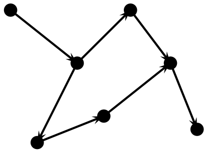
Patricia Bouyer

LSV – CNRS & ENS de Cachan

Model-checking

Does the system satisfy the property?

Modelling

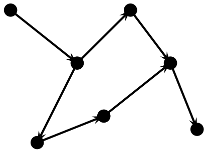


φ

Model-checking

Does the system satisfy the property?

Modelling



$\models$

φ

Model-checking Algorithm

Time!

Context: verification of embedded critical systems

Time

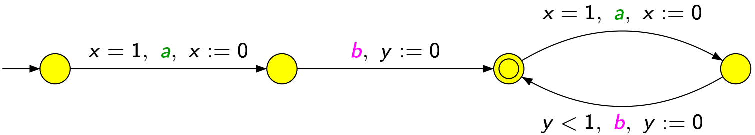
- naturally appears in real systems
- appears in properties (for ex. bounded response time)

➜ 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^+)

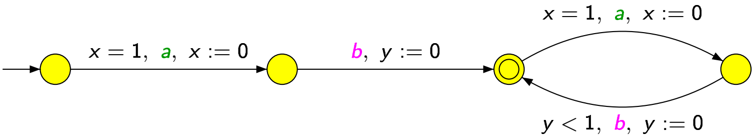
- A compositionality problem with discrete time
- Dense-time is a more general model than discrete time
- Not all timed systems can be discretized...



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...

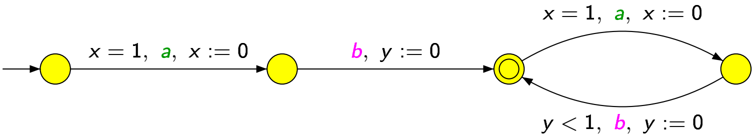


- Dense-time:
  $L_{dense} = \{((ab)^\omega, \tau) \mid \forall i,\ \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1}\}$

A case for dense-time

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

- A compositionality problem with discrete time
- Dense-time is a more general model than discrete time
- Not all timed systems can be discretized...

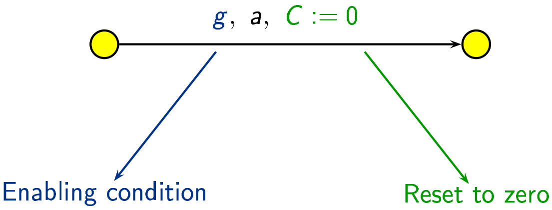


- Dense-time:
  $L_{dense} = \{((ab)^\omega, \tau) \mid \forall i,\ \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1}\}$
- Discrete-time: $L_{discrete} = \emptyset$

Outline

Timed automata [Alur & Dill 90's]

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



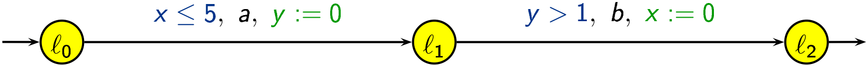
- An enabling condition (or guard) is:

$$g \quad ::= \quad x \sim c \quad | \quad g \wedge g$$

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

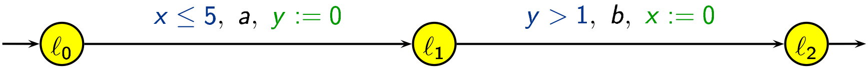
Timed automata (example)

x, y : clocks



Timed automata (example)

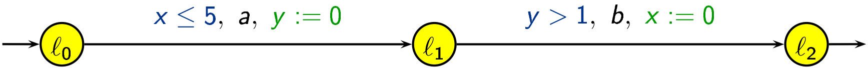
x, y : clocks



$$\begin{array}{cccccccc} & \ell_0 & \xrightarrow{\delta(4.1)} & \ell_0 & \xrightarrow{a} & \ell_1 & \xrightarrow{\delta(1.4)} & \ell_1 & \xrightarrow{b} & \ell_2 \\ x & 0 & & 4.1 & & 4.1 & & 5.5 & & 0 \\ y & 0 & & 4.1 & & 0 & & 1.4 & & 1.4 \end{array}$$

Timed automata (example)

x, y : clocks

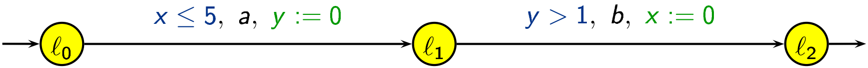


$$
\begin{array}{ccccccccc}
 & \ell_0 & \xrightarrow{\delta(4.1)} & \ell_0 & \xrightarrow{a} & \ell_1 & \xrightarrow{\delta(1.4)} & \ell_1 & \xrightarrow{b} & \ell_2 \\
x & 0 & & 4.1 & & 4.1 & & 5.5 & & 0 \\
y & 0 & & 4.1 & & 0 & & 1.4 & & 1.4
\end{array}
$$

(clock) valuation

Timed automata (example)

x, y : clocks



$$
\begin{array}{cccccccc}
 & \ell_0 & \xrightarrow{\delta(4.1)} & \ell_0 & \xrightarrow{a} & \ell_1 & \xrightarrow{\delta(1.4)} & \ell_1 & \xrightarrow{b} & \ell_2 \\
x & 0 & & 4.1 & & 4.1 & & 5.5 & & 0 \\
y & 0 & & 4.1 & & 0 & & 1.4 & & 1.4
\end{array}
$$

(clock) valuation

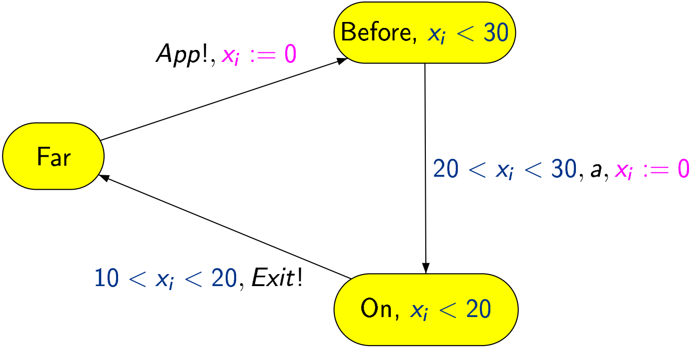
➜ timed word $(a, 4.1)(b, 5.5)$

Timed automata semantics

- $\mathcal{A} = (\Sigma, L, X, \longrightarrow)$ 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 $d \in T$

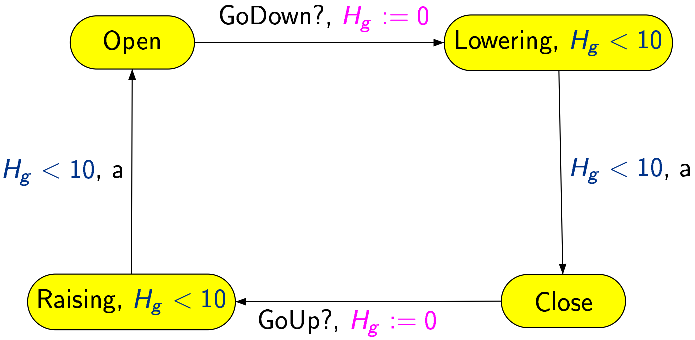
The train crossing example (1)

Train$_i$ with $i = 1, 2, \ldots$



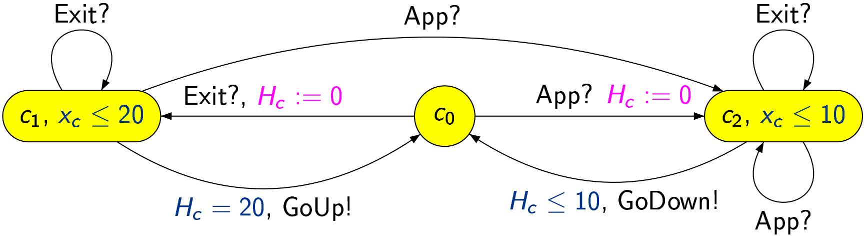
The train crossing example (2)

The gate:



The train crossing example (3)

The controller:



The train crossing example (4)

We use the synchronization function f:

Train_1	Train_2	Gate	Controller	
$App!$	.	.	$App?$	App
.	$App!$	.	$App?$	App
$Exit!$	.	.	$Exit?$	$Exit$
.	$Exit!$	.	$Exit?$	$Exit$
a	.	.	.	a
.	a	.	.	a
.	.	a	.	a
.	.	$GoUp?$	$GoUp!$	$GoUp$
.	.	$GoDown?$	$GoDown!$	$GoDown$

to define the parallel composition $(\mathsf{Train}_1 \parallel \mathsf{Train}_2 \parallel \mathsf{Gate} \parallel \mathsf{Controller})$

NB: the parallel composition does not add expressive power!

The train crossing example (5)

Some properties one could check:

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

The train crossing example (5)

Some properties one could check:

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

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

The train crossing example (5)

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?

The train crossing example (5)

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?

$$\neg EF(\text{gate.Close} \wedge (\text{gate.Close}\ U_{>5min}\ \neg\text{gate.Close}))$$

Outline

Classical verification problems

- reachability of a control state
- $\mathcal{S} \sim \mathcal{S}'$: bisimulation, etc...
- $L(\mathcal{S}) \subseteq L(\mathcal{S}')$: language inclusion
- $\mathcal{S} \models \varphi$ for some formula φ (*e.g.* in a timed extension of classical temporal logics): model-checking
- $\mathcal{S} \parallel A_T$ + reachability: testing automata
- ...

Verification

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

- reachability properties (final states)
- basic liveness properties (Büchi (or other) conditions)

Verification

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

- **Problem:** the set of configurations is infinite
  ➜ classical methods can not 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?

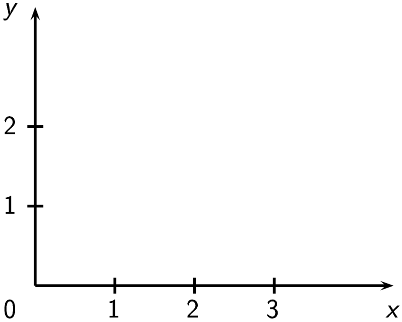
- **Problem:** the set of configurations is infinite
  ➜ classical methods can not be applied

- **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]**

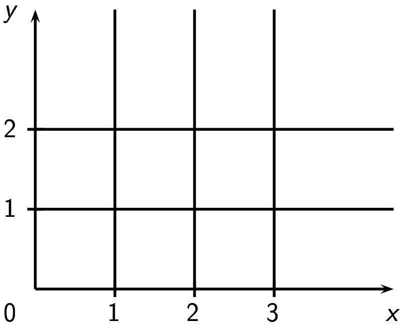
Method: construct a finite abstraction

The region abstraction



Equivalence of finite index

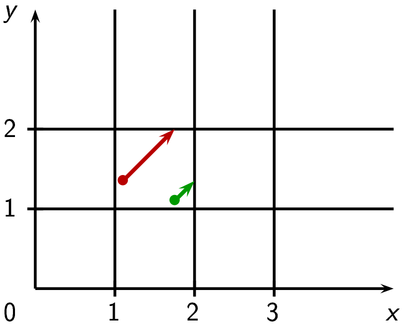
The region abstraction



Equivalence of finite index

- "compatibility" between regions and constraints

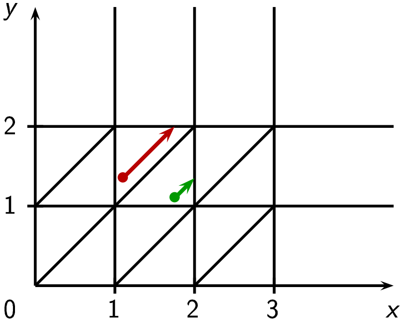
The region abstraction



Equivalence of finite index

- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

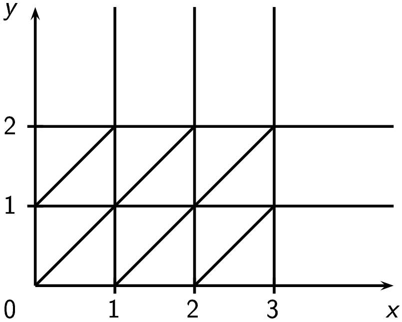
The region abstraction



Equivalence of finite index

- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

The region abstraction

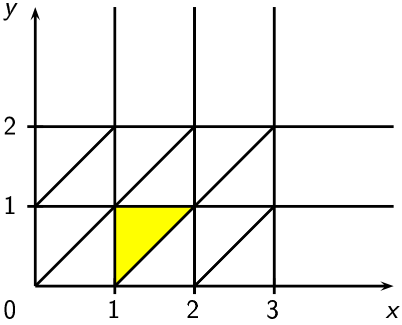


Equivalence of finite index

- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

➜ a bisimulation property

The region abstraction



Equivalence of finite index

region defined by

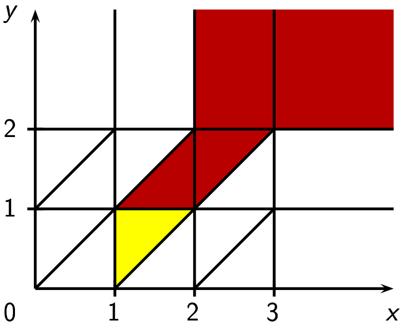
$I_x =]1;2[, \ I_y =]0;1[$

$\{x\} < \{y\}$

- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

➜ a bisimulation property

The region abstraction



Equivalence of finite index

region defined by

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

$\{x\} < \{y\}$

successor regions

- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

➜ a bisimulation property

The region automaton

timed automaton $\otimes$ region abstraction

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

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

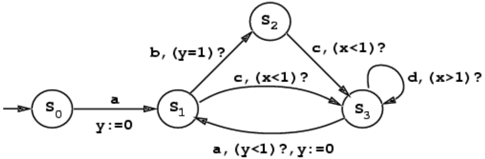
- $R'' \subseteq g$
- $[C \leftarrow 0]R'' \subseteq R'$

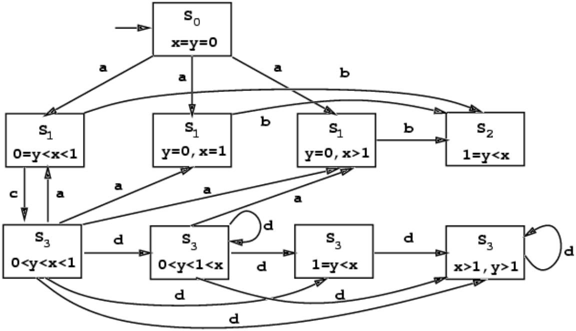
➜ **time-abstract bisimulation**

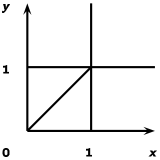
$\mathcal{L}$(reg. aut.) = UNTIME($\mathcal{L}$(timed aut.))

where $\mathrm{UNTIME}((a_1, t_1)(a_2, t_2)\dots) = a_1 a_2 \dots$

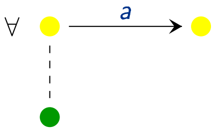
An example [AD 90's]



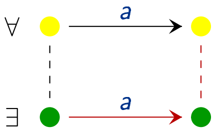




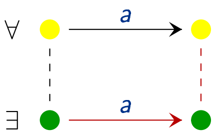
Time-abstract bisimulation

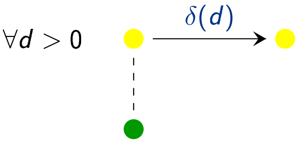


Time-abstract bisimulation

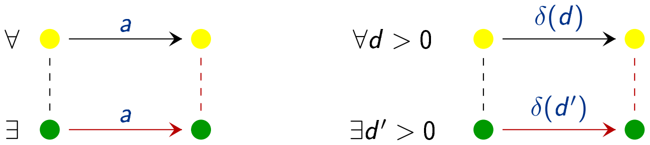


Time-abstract bisimulation

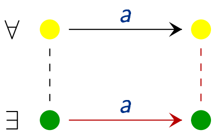


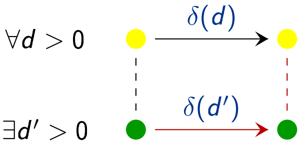


Time-abstract bisimulation



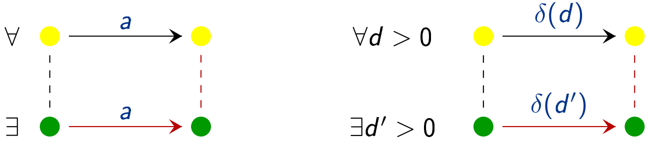
Time-abstract bisimulation





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

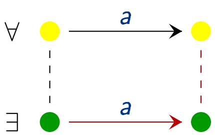
Time-abstract bisimulation

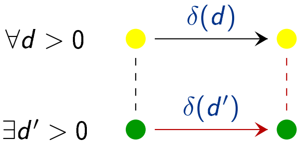


$$
\begin{tikzcd}
(\ell_0, v_0) \arrow[r, "a_1, t_1"] \arrow[d] & (\ell_1, v_1) \arrow[r, "a_2, t_2"] \arrow[d] & (\ell_2, v_2) \arrow[r, "a_3, t_3"] \arrow[d] & \ldots \\
(\ell_0, R_0) \arrow[r, "a_1"] & (\ell_1, R_1) \arrow[r, "a_2"] & (\ell_2, R_2) \arrow[r, "a_3"] & \ldots
\end{tikzcd}
$$

with $v_i \in R_i$ for all i.

Time-abstract bisimulation

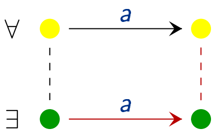


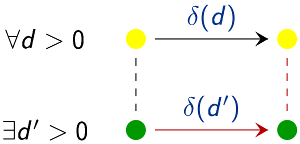


$$
\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \dots \\
\updownarrow & & \updownarrow & & \updownarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3} & \dots
\end{array}
$$

with $v_i \in R_i$ for all i.

Time-abstract bisimulation





$$
\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \ldots \\
\updownarrow & & \updownarrow & & \updownarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3} & \ldots
\end{array}
$$

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...

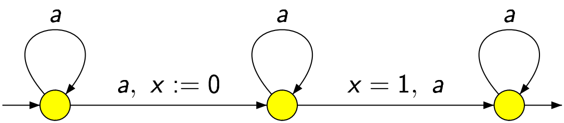
A model not far from undecidability

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

A model not far from undecidability

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

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

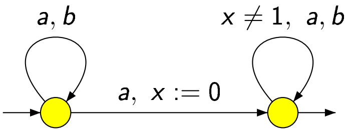


A model not far from undecidability

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

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

[Alur,Madhusudan 2004]



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

Partial conclusion

➜ a timed model interesting for verification purposes

Numerous works have been (and are) devoted to:

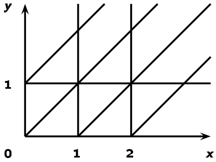
- the "theoretical" comprehension of timed automata (*cf* **[Asarin 2004]**)
- extensions of the model (to ease modelling)
  - expressiveness
  - analyzability
- algorithmic problems and implementation

Outline

Role of diagonal constraints

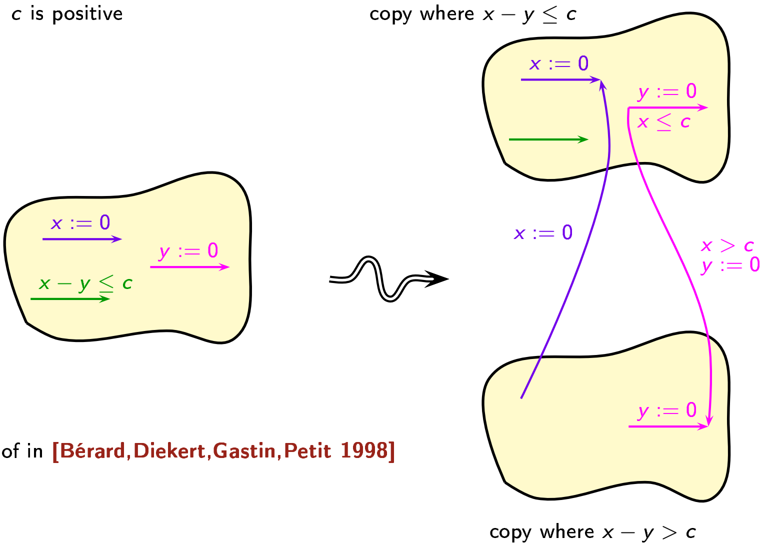
$$x - y \sim c \quad \text{and} \quad x \sim c$$

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



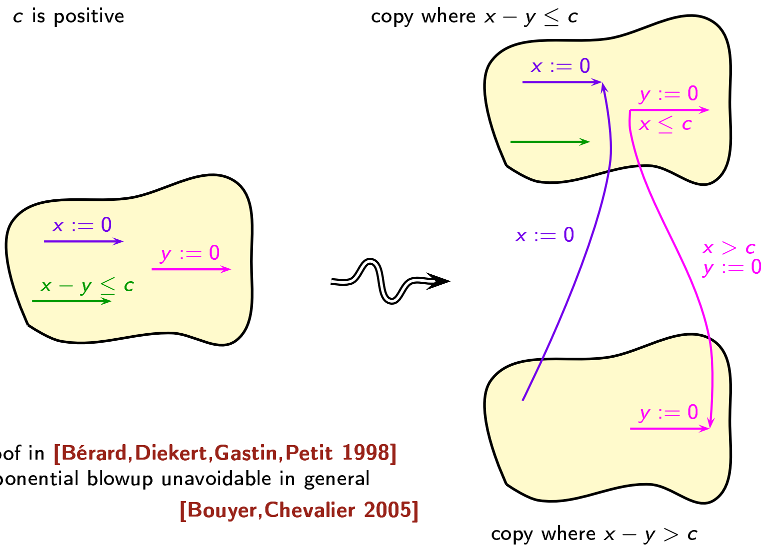
- **Expressiveness:** no additional expressive power

Role of diagonal constraints (cont.)



➜ proof in [Bérard,Diekert,Gastin,Petit 1998]

Role of diagonal constraints (cont.)



➜ proof in **[Bérard,Diekert,Gastin,Petit 1998]**
➜ exponential blowup unavoidable in general **[Bouyer,Chevalier 2005]**

Adding silent actions

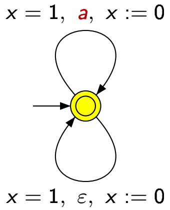
$$\xrightarrow{g,\,\varepsilon,\,C := 0}$$

[Bérard,Diekert,Gastin,Petit 1998]

- **Decidability:** yes

  (actions have no influence on region automaton construction)

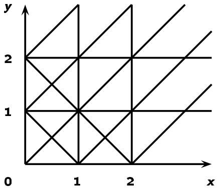
- **Expressiveness:** strictly more expressive!



Adding constraints of the form $x + y \sim c$

$x + y \sim c$ and $x \sim c$ **[Bérard, Dufourd 2000]**

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

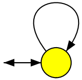


  - for four clocks (or more), undecidable!

▸ The proof

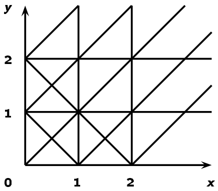
- **Expressiveness:** more expressive! (even using two clocks)

$$\{(a^n, t_1 \dots t_n) \mid n \geq 1 \text{ and } t_i = 1 - \frac{1}{2^i}\}$$



Adding constraints of the form $x + y \sim c$

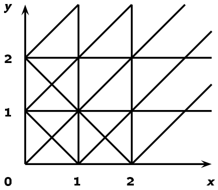
- Two clocks: decidable using the abstraction



- Four clocks (or more): undecidable!

Adding constraints of the form $x + y \sim c$

- Two clocks: decidable using the abstraction

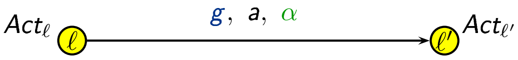


- Three clocks: open question!

- Four clocks (or more): undecidable!

Linear hybrid automata [Henzinger 1996]

- A finite control structure + a set X of **dynamical variables**
- A transition is of the form:

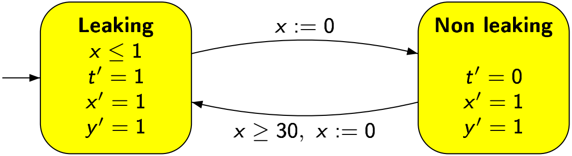


- g is a linear constraint on variables
- α is a jump condition, *i.e.* an affine update of the form $X' = A.X + B$
- in each state, an activity function assigning a slope to each variable (for each $x \in X$, $Act(x) \in [\ell, u]$)

Linear hybrid automata (example)

The **gas burner** may leak. [ACHH93]

- 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

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

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]

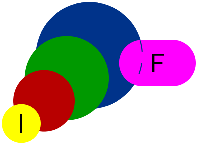
...but **on-the-fly technics** are prefered.

Reachability analysis

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

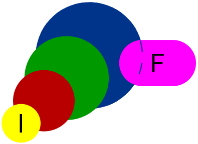
Reachability analysis

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

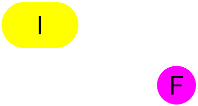


Reachability analysis

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

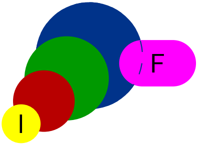


- **backward analysis algorithm:**
  compute the predecessors of final configurations

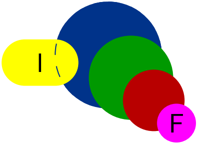


Reachability analysis

- **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

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \bowtie b$$

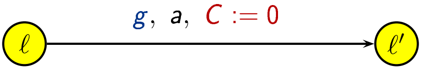
➜ tool HyTech

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

Timed automata: particular kind of polyhedra called zones

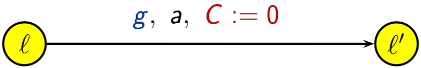
$$x_i - x_j \bowtie c \qquad \text{or} \qquad x_i \bowtie c$$

Note on the backward analysis of TA

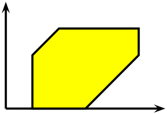


$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\qquad Z$

Note on the backward analysis of TA



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

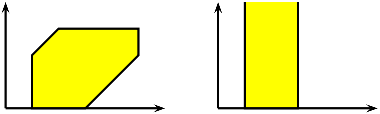


Z

Note on the backward analysis of TA

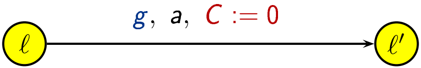
$$\ell \xrightarrow{\;g,\ a,\ C := 0\;} \ell'$$

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

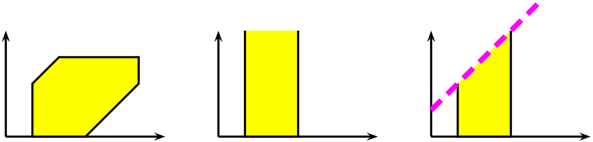


Z $\qquad\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$

Note on the backward analysis of TA

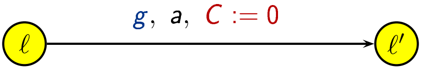


$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\qquad$ Z

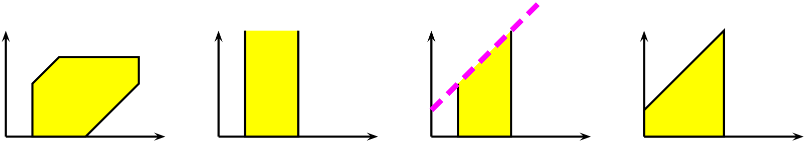


Z $\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$

Note on the backward analysis of TA

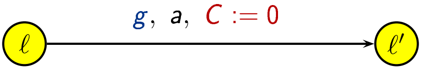


$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\qquad$ Z

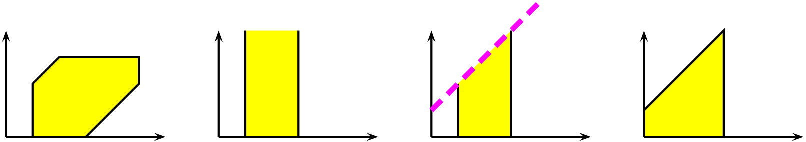


Z $\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$ $\qquad$ $\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$

Note on the backward analysis of TA



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

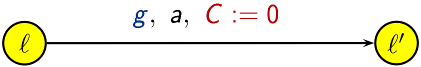


Z $\quad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$ $\quad$ $\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$

The exact backward computation terminates and is correct!

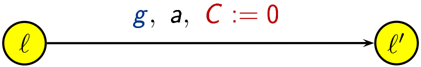
▸ The proof

Forward analysis of timed automata

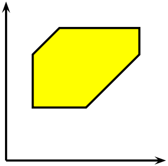


zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

Forward analysis of timed automata

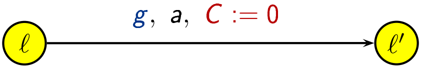


zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

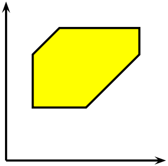


Z

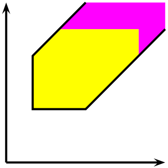
Forward analysis of timed automata



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



Z

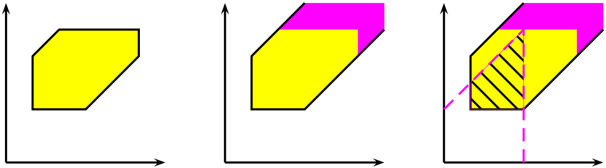


$\overrightarrow{Z}$

Forward analysis of timed automata

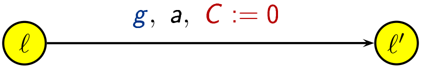
$$\ell \xrightarrow{g,\ a,\ C := 0} \ell'$$

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

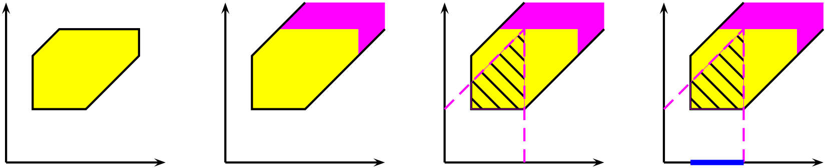


$Z \quad\quad \overrightarrow{Z} \quad\quad \overrightarrow{Z} \cap g$

Forward analysis of timed automata

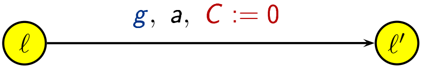


zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

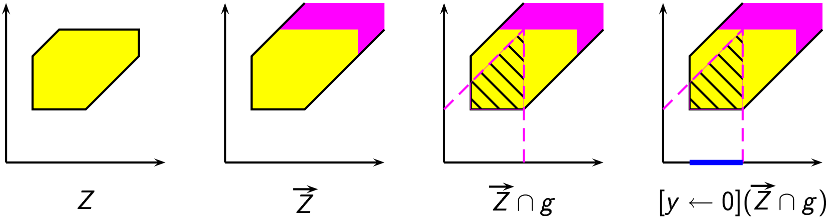


$Z \quad \overrightarrow{Z} \quad \overrightarrow{Z} \cap g \quad [y \leftarrow 0](\overrightarrow{Z} \cap g)$

Forward analysis of timed automata

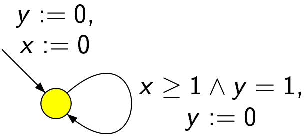


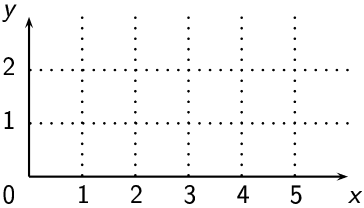
zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$



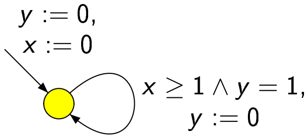
➜ a termination problem

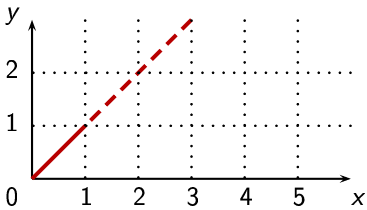
Non termination of the forward analysis



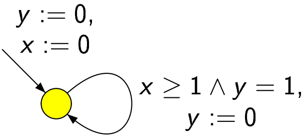


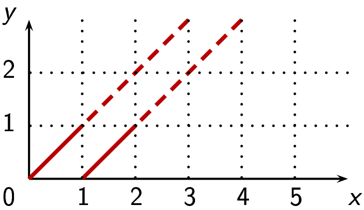
Non termination of the forward analysis



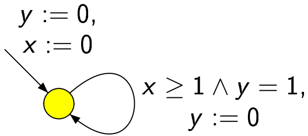


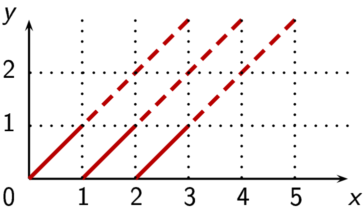
Non termination of the forward analysis



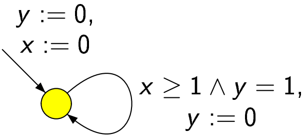


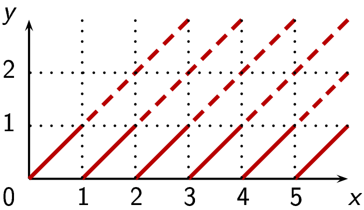
Non termination of the forward analysis





Non termination of the forward analysis





➜ 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

➜ 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

  ➜ correct w.r.t. reachability

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

  ➜ correct w.r.t. reachability

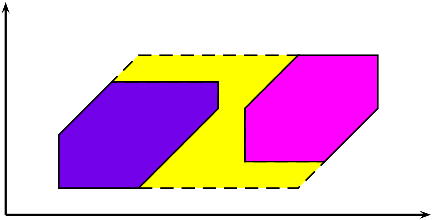
$$q \xrightarrow{g,a,C:=0} q' \quad \text{implies} \quad \mathrm{Act}(q) = \mathrm{clocks}(g) \cup (\mathrm{Act}(q') \setminus C)$$

...

"Solutions" to this problem (cont.)

- **convex-hull approximation**: if Z and Z' are computed then we overapproximate using "$Z \sqcup Z'$".

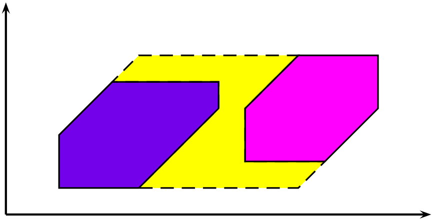
➜ "semi-correct" w.r.t. reachability



"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

[Berthomieu, Menasche 1983] [Dill 1989]

$$(x_1 \geq 3) \ \wedge \ (x_2 \leq 5) \ \wedge \ (x_1 - x_2 \leq 4)$$

$$\begin{array}{c c} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{pmatrix} \end{array}$$

The DBM data structure

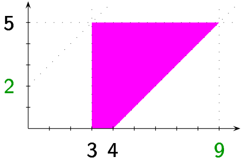
DBM (Difference Bounded Matrice) data structure

[Berthomieu, Menasche 1983] [Dill 1989]

$$(x_1 \geq 3) \;\wedge\; (x_2 \leq 5) \;\wedge\; (x_1 - x_2 \leq 4)$$

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{pmatrix} \end{array}$$

- Existence of a normal form



$$\begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix}$$

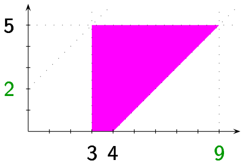
The DBM data structure

DBM (Difference Bounded Matrice) data structure

[Berthomieu, Menasche 1983] [Dill 1989]

$$(x_1 \geq 3) \ \wedge \ (x_2 \leq 5) \ \wedge \ (x_1 - x_2 \leq 4) \qquad \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{pmatrix} \end{array}$$

- Existence of a normal form



$$\begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix}$$

- All previous operations on zones can be computed using DBMs

The extrapolation operator

Fix an integer k ("$*$" represents an integer between $-k$ and $+k$)

$$\begin{pmatrix} * & \boxed{>k} & * \\ * & * & * \\ \boxed{<-k} & * & * \end{pmatrix} \quad \rightsquigarrow \quad \begin{pmatrix} * & \boxed{+\infty} & * \\ * & * & * \\ \boxed{-k} & * & * \end{pmatrix}$$

- "intuitively", erase non-relevant constraints

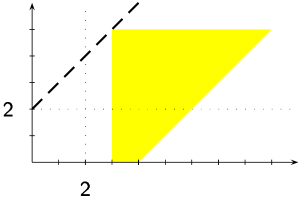
➜ ensures termination

The extrapolation operator

Fix an integer k ("$*$" represents an integer between $-k$ and $+k$)

$$\begin{pmatrix} * & \boxed{>k} & * \\ * & * & * \\ \boxed{<-k} & * & * \end{pmatrix} \rightsquigarrow \begin{pmatrix} * & \boxed{+\infty} & * \\ * & * & * \\ \boxed{-k} & * & * \end{pmatrix}$$

- "intuitively", erase non-relevant constraints



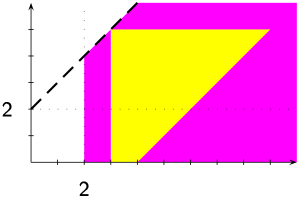
➜ ensures termination

The extrapolation operator

Fix an integer k ("$*$" represents an integer between $-k$ and $+k$)

$$\begin{pmatrix} * & \boxed{>k} & * \\ * & * & * \\ \boxed{<-k} & * & * \end{pmatrix} \rightsquigarrow \begin{pmatrix} * & \boxed{+\infty} & * \\ * & * & * \\ \boxed{-k} & * & * \end{pmatrix}$$

- "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...

A counter-example

- 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

Discussion on complexity

[Alur 1991, Alur Henzinger 1994, Alur Courcoubetis Dill 1993, Aceto Laroussinie 2002]

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

Discussion on complexity

[Alur 1991, Alur Henzinger 1994, Alur Courcoubetis Dill 1993, Aceto Laroussinie 2002]

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

Timing constraints induce a complexity blowup!

Discussion on complexity

[Alur 1991, Alur Henzinger 1994, Alur Courcoubetis Dill 1993, Aceto Laroussinie 2002]

	Kripke structures S	Timed automaton A or $(S_1 \parallel \ldots \parallel S_n)$
Reachability	NLOGSPACE-complete	PSPACE-complete
CTL/TCTL	P-complete	PSPACE-complete
AF-μ-calc./$L_{\mu,\nu}$	P-complete	EXPTIME-complete
full μ-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!

Discussion on complexity

[Alur 1991, Alur Henzinger 1994, Alur Courcoubetis Dill 1993, Aceto Laroussinie 2002]

	Kripke structures S	Timed automaton A or $(S_1 \parallel \ldots \parallel S_n)$ or $(A_1 \parallel \ldots \parallel A_n)$
Reachability	NLOGSPACE-complete	PSPACE-complete
CTL/TCTL	P-complete	PSPACE-complete
AF-μ-calc./$L_{\mu,\nu}$	P-complete	EXPTIME-complete
full μ-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 ➜ 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.

Conclusion & Further Work

- 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

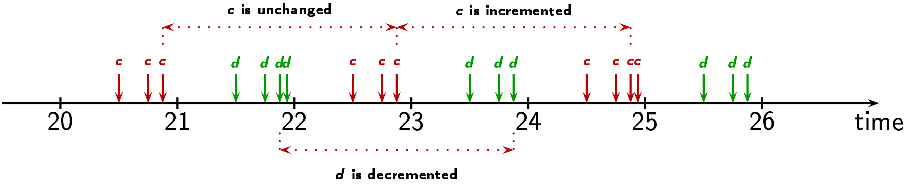
Definition. A two-counter machine is a finite set of instructions over two counters (x and y):

- Incrementation:
  $\texttt{(p)}: \quad x := x + 1;\ \texttt{goto (q)}$

- Decrementation:
  $\texttt{(p)}: \quad \texttt{if } x > 0 \texttt{ then } x := x - 1;\ \texttt{goto (q) else goto (r)}$

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

Undecidability proof



➜ simulation of
- decrementation of a counter
- incrementation of a counter

We will use 4 clocks:

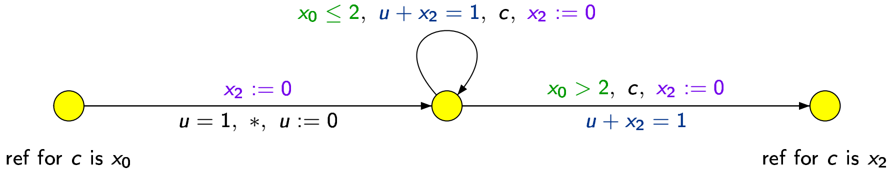
- u, "tic" clock (each time unit)
- x_0, x_1, x_2: reference clocks for the two counters

"x_i reference for c" $\equiv$ "the last time x_i has been reset is the last time action c has been performed"

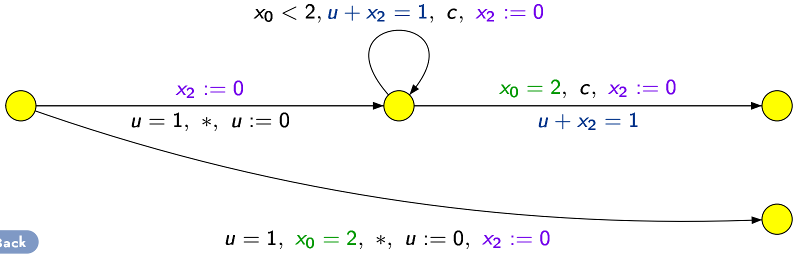
[Bérard,Dufourd 2000]

Undecidability proof (cont.)

- **Incrementation of counter c:**



- **Decrementation of counter c:**



Back

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"

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"

Let R be a region. Assume:

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

There exists t' s.t. $v' + t' \equiv_{\mathbf{reg.}} v + t$, which implies that $v' + t' \in R$ and thus $v' \in \overleftarrow{R}$.

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:

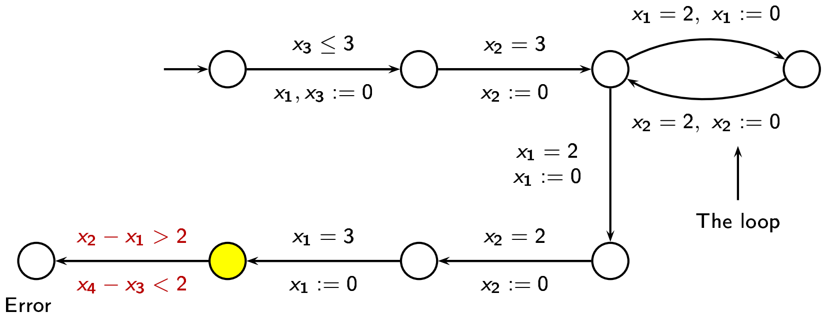
> "Every set of valuations which is computed along the backward computation is a finite union of regions"

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

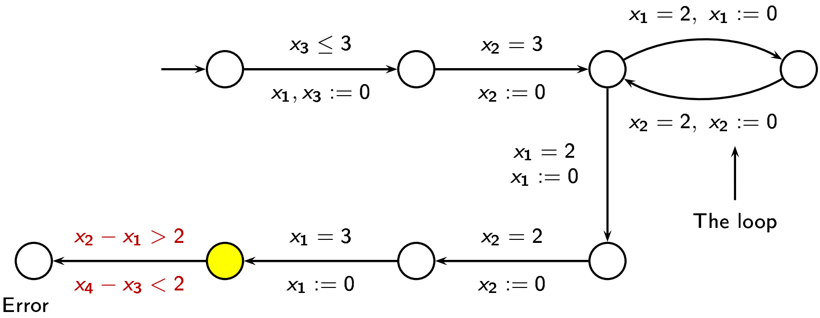
$$i := j.k + \ell.m$$

Back

A problematic automaton

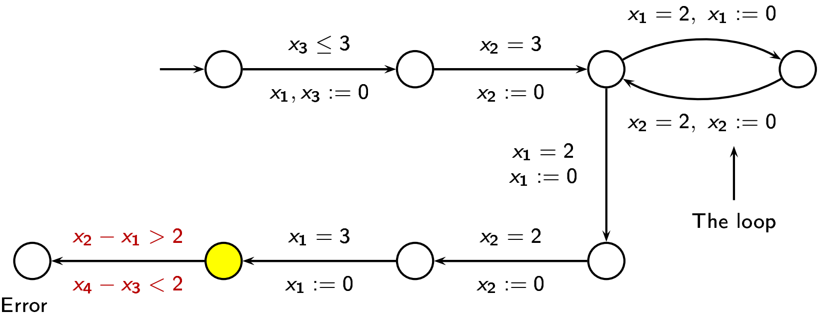


A problematic automaton

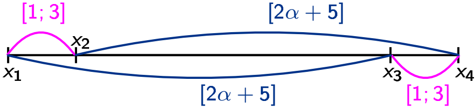


$$\begin{cases} v(x_1) = 0 \\ v(x_2) = d \\ v(x_3) = 2\alpha + 5 \\ v(x_4) = 2\alpha + 5 + d \end{cases}$$

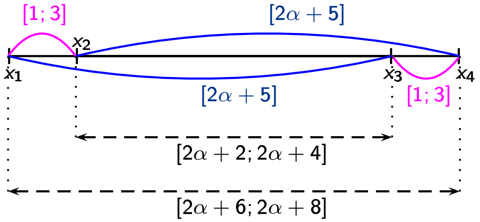
A problematic automaton



$$\begin{cases} v(x_1) = 0 \\ v(x_2) = d \\ v(x_3) = 2\alpha + 5 \\ v(x_4) = 2\alpha + 5 + d \end{cases}$$

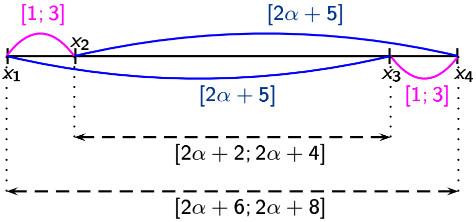


The problematic zone



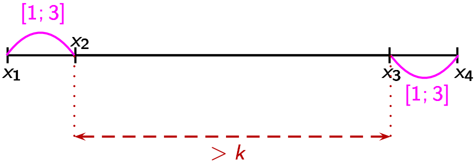
implies $\quad x_1 - x_2 = x_3 - x_4.$

The problematic zone



implies $x_1 - x_2 = x_3 - x_4$.

If α is sufficiently large, after extrapolation:



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

▸ Back