Timed Models: From Theory to Implementation

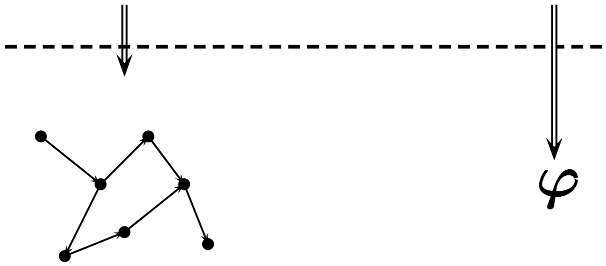
Patricia Bouyer

LSV - CNRS & ENS de Cachan

Model-checking

Does the system satisfy the property?

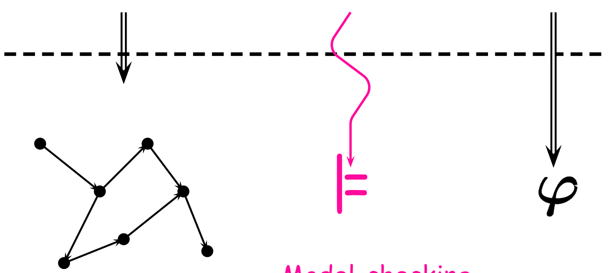
Modelling



Model-checking

Does the system satisfy the property?

Modelling



Model-checking Algorithm

Time!

Context: verification of embedded critical systems

Time

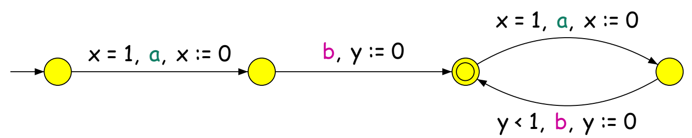
- ✔ naturally appears in real systems
- ✔ appears in properties (for ex. bounded response time)

➜ 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
- ✔



- 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: $L_{discrete} = \emptyset$

Roadmap

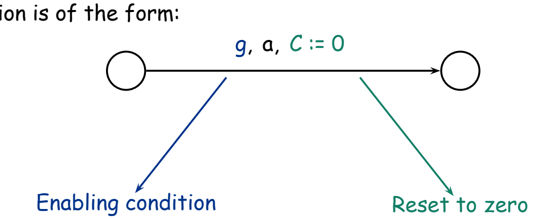
- ✔ **Timed automata, decidability issues**
- ✔ **Some extensions of the model**
- ✔ **Implementation of timed automata**

Timed automata, decidability issues

- ✔ presentation of the model
- ✔ decidability of the model
- ✔ the region automaton construction

Timed automata

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



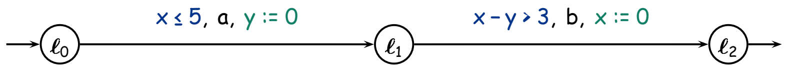
- ✔ An enabling condition (or guard) is:

$$g \ ::= \ x \sim c \ \mid \ x - y \sim c \ \mid \ g \wedge g$$

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

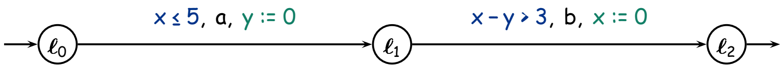
Timed automata (example)

x, y : clocks



Timed automata (example)

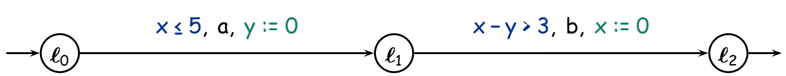
x, y : clocks



	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

Timed automata (example)

x, y : clocks

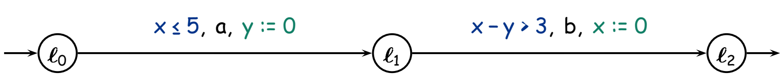


	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

(clock) valuation

Timed automata (example)

x, y : clocks



	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

(clock) valuation

➜ timed word $(a, 4.1)(b, 5.5)$

TA 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. $v \models g$, $v' = v[r \leftarrow 0]$
  - **delay transition:** $(q, v) \xrightarrow{\delta(d)} (q, v + d)$ if $d \in T$

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

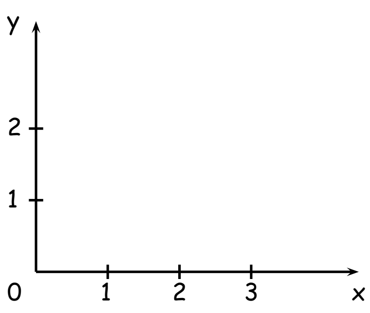
Positive key point: variables (clocks) have the same speed

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

Method: construct a finite abstraction

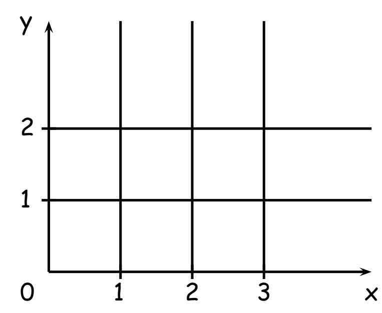
QEST'04 - Tutorial - September 2004 Timed Models: From Theory to Implementation - p. 10

The region abstraction



Equivalence of finite index

The region abstraction

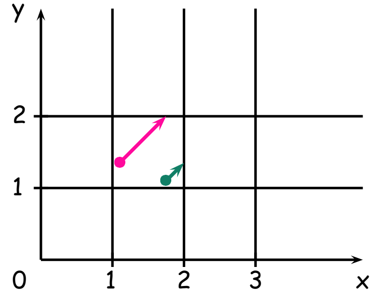


Equivalence of finite index

✔ "compatibility" between regions and constraints

QEST'04 - Tutorial - September 2004 Timed Models: From Theory to Implementation - p. 11

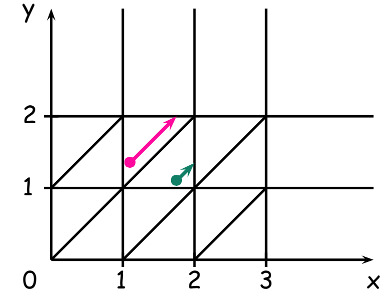
The region abstraction



Equivalence of finite index

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

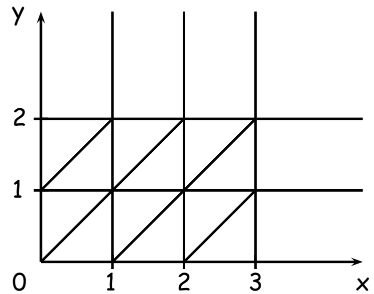
The region abstraction



Equivalence of finite index

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

The region abstraction

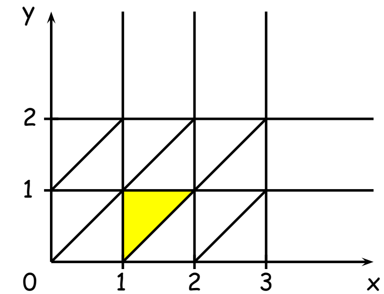


Equivalence of finite index

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

➜ a bisimulation property

The region abstraction



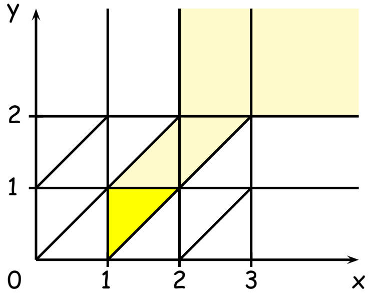
Equivalence of finite index

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

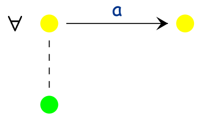
region defined by
$I_x =]1;2[$, $I_y =]0;1[$
$\{x\} < \{y\}$

successor regions

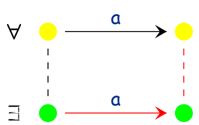
- ✔ "compatibility" between regions and constraints
- ✔ "compatibility" between regions and time elapsing

➜ a bisimulation property

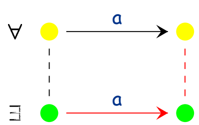
Time-Abstract Bisimulation

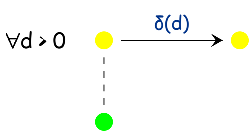


Time-Abstract Bisimulation

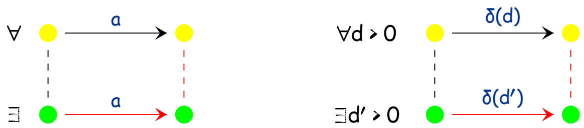


Time-Abstract Bisimulation

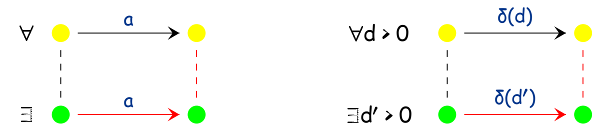




Time-Abstract Bisimulation

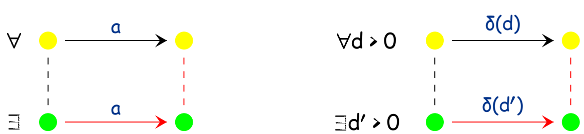


Time-Abstract Bisimulation



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

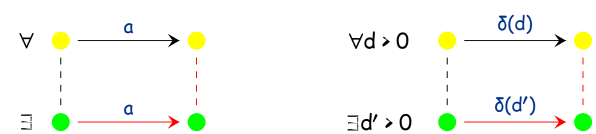
Time-Abstract Bisimulation



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

with $v_i \in R_i$ for all i.

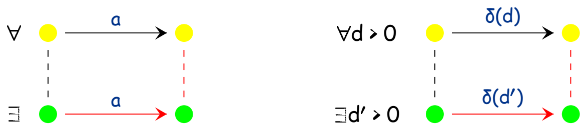
Time-Abstract Bisimulation



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

with $v_i \in R_i$ for all i.

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.

Remark: We can not check real-time properties with a time-abstract bisimulation. We need to add clocks for the formula we want to check.

The region automaton

timed automaton ⊗ 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.

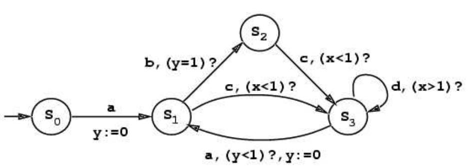
- ✔ $R'' \subseteq g$
- ✔ $[C \leftarrow 0]R'' \subseteq R'$

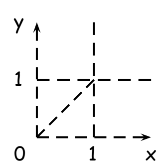
➜ time-abstract bisimulation

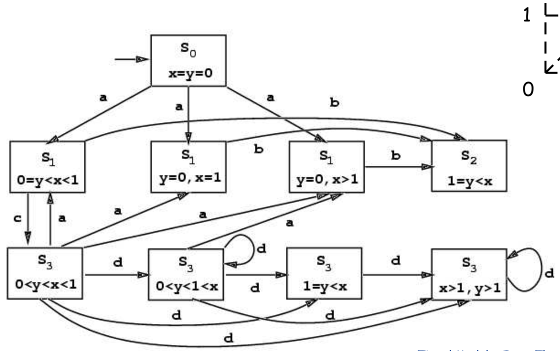
$\mathcal{L}$(reg. aut.) = UNTIME($\mathcal{L}$(timed aut.))

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

An example [AD 90's]







PSPACE-Easyness

¡ The size of the region graph is in $\mathcal{O}(|X|!.2^{|X|})$!

- ✔ **One configuration:** a discrete location + a region

PSPACE-Easyness

- ¡ The size of the region graph is in $\mathcal{O}(|X|!.2^{|X|})$!
- **One configuration:** a discrete location + a region
  - a discrete location: log-space

PSPACE-Easyness

- ¡ The size of the region graph is in $\mathcal{O}(|X|!.2^{|X|})$!
- **One configuration:** a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks

PSPACE-Easyness

- ¡ The size of the region graph is in $\mathcal{O}(|X|!.2^{|X|})$!
- **One configuration:** a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks

➜ needs polynomial space

PSPACE-Easyness

- ¡ The size of the region graph is in $\mathcal{O}(|X|!.2^{|X|})$!
- **One configuration:** a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks

➜ needs polynomial space

- By guessing a path: needs only to store two configurations

PSPACE-Easyness

¡ The size of the region graph is in $\mathcal{O}(|X|!.2^{|X|})$!

- ✔ **One configuration:** a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks

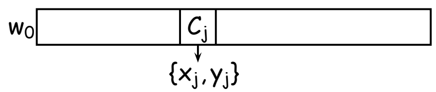
➜ needs polynomial space

- ✔ By guessing a path: needs only to store two configurations

➜ in NPSPACE, thus in PSPACE

PSPACE-Hardness

$\mathcal{M}$ LBTM
$w_0 \in \{a,b\}^*$ } $\leadsto$ $A_{\mathcal{M},w_0}$ s.t. $\mathcal{M}$ accepts w_0 iff the final state of $A_{\mathcal{M},w_0}$ is reachable



C_j contains a "a" iff $x_j = y_j$

C_j contains a "b" iff $x_j < y_j$

(these conditions are invariant by time elapsing)

➜ proof taken in **[Aceto & Laroussinie 2002]**

PSPACE-Hardness (cont.)

If $q \xrightarrow{a,a',\delta} q'$ is a transition of $\mathcal{M}$, then for each position i of the tape, we have a transition

$$(q,i) \xrightarrow{g,r:=0} (q',i')$$

where:

- ✔ g is $x_i = y_i$ (resp. $x_i < y_i$) if $a = a$ (resp. $a = b$)
- ✔ $r = \{x_i, y_i\}$ (resp. $r = \{x_i\}$) if $a = a$ (resp. $a = b$)
- ✔ $i' = i + 1$ (resp. $i' = i - 1$) if δ is right and $i < n$ (resp. left)

Enforcing time elapsing: on each transition, add the condition $t = 1$ and clock t is reset.

Initialization: init $\xrightarrow{t=1, r_0:=0}$ $(q_0, 1)$ where $r_0 = \{x_i \mid w_0[i] = b\} \cup \{t\}$

Termination: $(q_f, i) \longrightarrow$ end

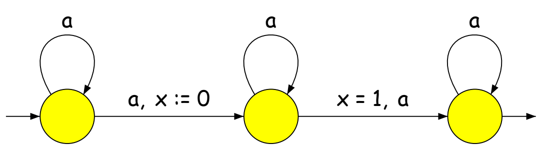
A Model Not Far From Undecidability

- ✔ 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]
- ✔ ...

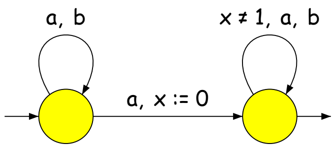
An example of non-determinizable TA:



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-complementable TA:



Partial conclusion

➜ a timed model interesting for verification purposes

Numerous works have been (and are) devoted to:

- ✔ the "theoretical" comprehension of timed automata
- ✔ extensions of the model (to ease the modelling)
  - expressiveness
  - analyzability
- ✔ algorithmic problems and implementation

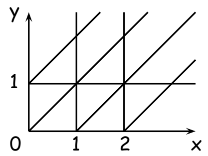
Some extensions of the model

- ✔ adding constraints of the form $x - y \sim c$
- ✔ adding silent actions
- ✔ adding constraints of the form $x + y \sim c$
- ✔ adding new operations on clocks

Adding diagonal constraints

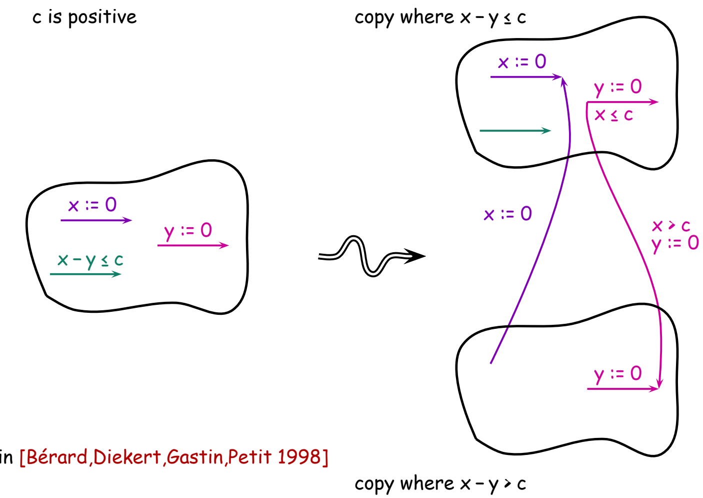
$x - y \sim c$ and $x \sim c$

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



- ✔ **Expressiveness:** no additional expressive power

Adding diagonal constraints (cont.)



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

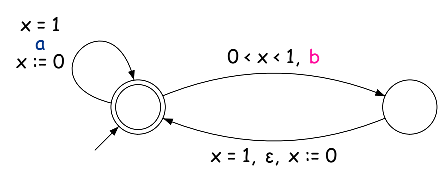
Adding diagonal constraints (cont.)

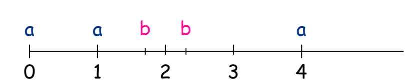
Open question: is this construction "optimal"?
In the sense that timed automata with diagonal constraints are exponentially more concise than diagonal-free timed automata.

Adding silent actions

$g, \varepsilon, C := 0$ [Bérard,Diekert,Gastin,Petit 1998]

- ✔ **Decidability:** yes (actions has no influence on the previous construction)
- ✔ **Expressiveness:** strictly more expressive!

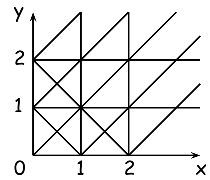




Adding constraints of the form $x + y \sim c$

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

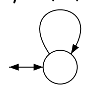
✔ **Decidability:** - for two clocks, decidable using the abstraction



- for four clocks (or more), undecidable!

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

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



The two-counter machine

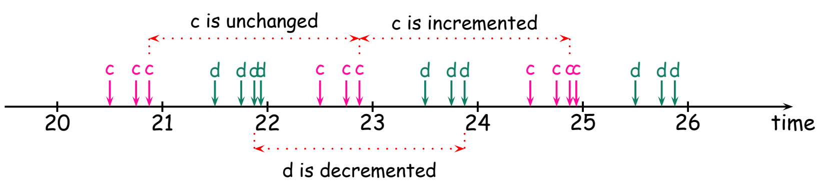
Definition. A two-counter machine is a finite set of instructions over two counters (x and y):

✔ Incrementation:
(p) : $x := x + 1$; goto (q)

✔ Decrementation:
(p) : if $x > 0$ then $x := x - 1$; goto (q) else goto (r)

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

Undecidability proof



➜ simulation of
- decrement of d
- increment of c

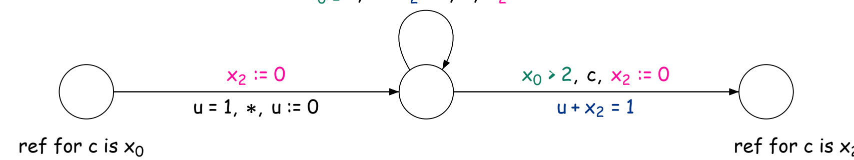
We will use 4 clocks:
- u, "tic" clock (each time unit)
- x_0, x_1, x_2: reference clocks for the two counters

"x_i reference for c" ≡ "the last time x_i has been reset is the last time action c has been performed"

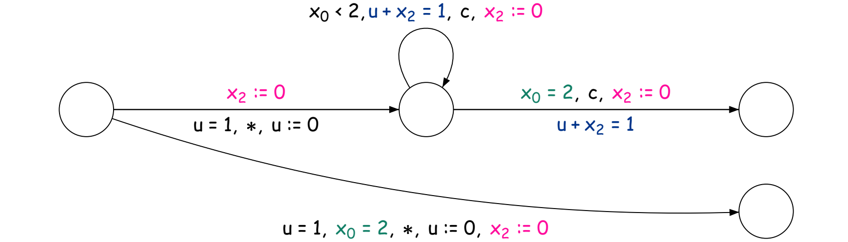
[Bérard,Dufourd 2000]

Undecidability proof (cont.)

✔ **Increment of counter c:**

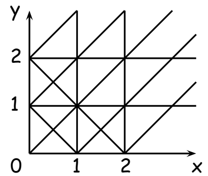


✔ **Decrement of counter c:**



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

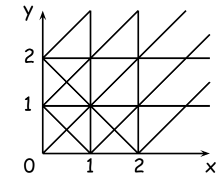
- ✔ Two clocks: decidable! using the abstraction



- ✔ 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!

Adding new operations on clocks

Several types of updates: $x := y + c$, $x :< c$, $x :> c$, etc...

Adding new operations on clocks

Several types of updates: $x := y + c$, $x :< c$, $x :> c$, etc...

- ✔ The general model is undecidable.

(simulation of a two-counter machine)

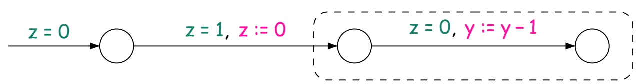
Adding new operations on clocks

Several types of updates: $x := y + c$, $x :< c$, $x :> c$, etc...

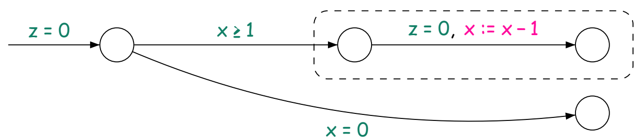
- ✔ The general model is undecidable.

  (simulation of a two-counter machine)

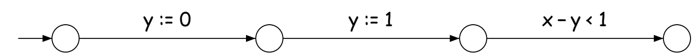
- ✔ Only decrementation also leads to undecidability
  - **Incrementation of counter** x



  - **Decrementation of counter** x



Decidability



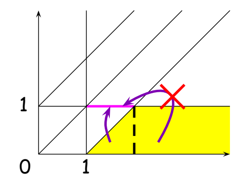


image by $y := 1$

➜ the bisimulation property is not met

The classical region automaton construction is not correct.

Decidability (cont.)

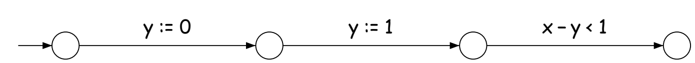
$\mathcal{A}$ ⇝ Diophantine linear inequations system
⇝ is there a solution?
⇝ if yes, belongs to a decidable class

Examples:

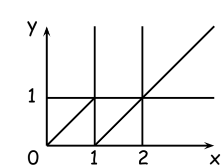
- ✔ constraint $x \sim c$ — $c \le max_x$
- ✔ constraint $x - y \sim c$ — $c \le max_{x,y}$
- ✔ update $x :\sim y + c$ — $max_x \le max_y + c$
  and for each clock z, $max_{x,z} \ge max_{y,z} + c$, $max_{z,x} \ge max_{z,y} - c$
- ✔ update $x :< c$ — $c \le max_x$
  and for each clock z, $max_z \ge c + max_{z,x}$

The constants (max_x) and ($max_{x,y}$) define a set of regions.

Decidability (cont.)



$$\begin{cases} max_y \ge 0 \\ max_x \ge 0 + max_{x,y} \\ max_y \ge 1 \\ max_x \ge 1 + max_{x,y} \\ max_{x,y} \ge 1 \end{cases} \Longrightarrow \begin{cases} max_x = 2 \\ max_y = 1 \\ max_{x,y} = 1 \\ max_{y,x} = -1 \end{cases}$$

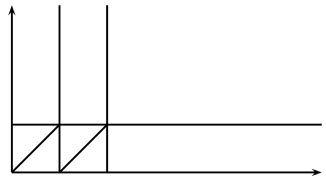


The bisimulation property is met.

What's wrong when undecidable?

Decrementation $x := x - 1$

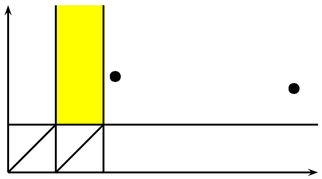
$max_x \leq max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

$max_x \leq max_x - 1$

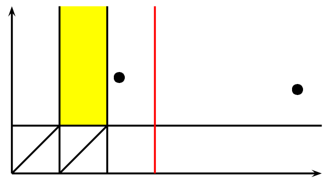


QEST'04 - Tutorial - September 2004 Timed Models: From Theory to Implementation - p. 33

What's wrong when undecidable?

Decrementation $x := x - 1$

$max_x \leq max_x - 1$

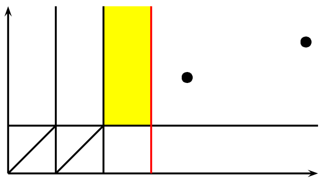


QEST'04 - Tutorial - September 2004 Timed Models: From Theory to Implementation - p. 33

What's wrong when undecidable?

Decrementation $x := x - 1$

$max_x \leq max_x - 1$

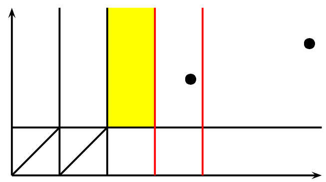


QEST'04 - Tutorial - September 2004 Timed Models: From Theory to Implementation - p. 33

What's wrong when undecidable?

Decrementation $x := x - 1$

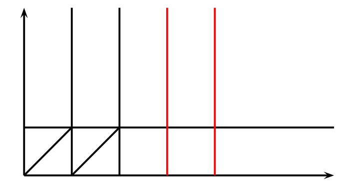
$max_x \leq max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

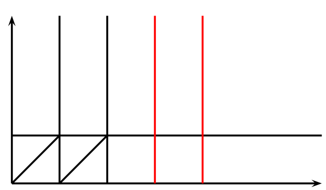
$max_x \leq max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

$max_x \leq max_x - 1$



etc...

Decidability (cont.)

	Diagonal-free constraints	General constraints
x := c, x := y	PSPACE-complete	PSPACE-complete
x := x + 1	PSPACE-complete	Undecidable
x := y + c		
x := x − 1	Undecidable	
x :< c	PSPACE-complete	PSPACE-complete
x :> c		Undecidable
x :~ y + c		
y + c <: x :< y + d		
y + c <: x :< z + d	Undecidable	

[Bouyer,Dufourd,Fleury,Petit 2000]

Implementation of Timed Automata

- ✔ analysis algorithms
- ✔ the DBM data structure
- ✔ a bug in the forward analysis

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]

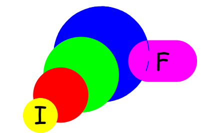
...but **on-the-fly technics** are preferred.

Reachability analysis

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

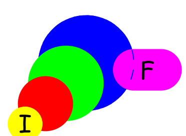
Reachability analysis

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

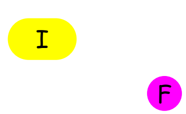


Reachability analysis

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

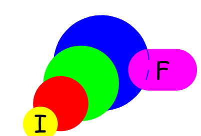


- ✔ **backward analysis algorithm:**
  compute the predecessors of final configurations

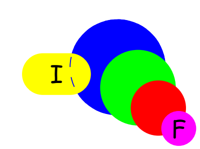


Reachability analysis

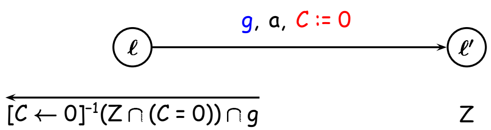
- ✔ **forward analysis algorithm:**
  compute the successors of initial configurations



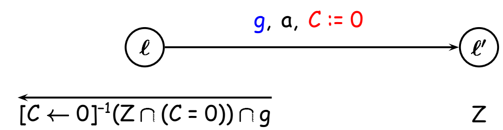
- ✔ **backward analysis algorithm:**
  compute the predecessors of final configurations

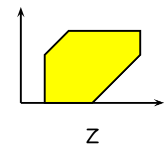


Note on the backward analysis

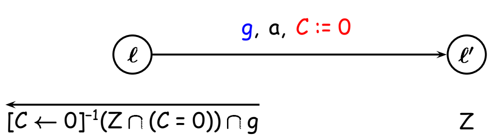


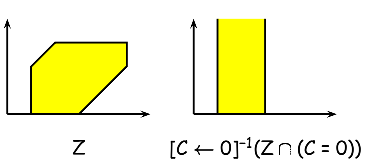
Note on the backward analysis



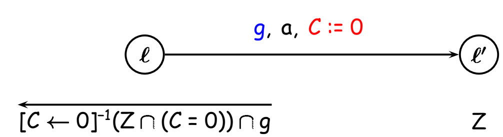


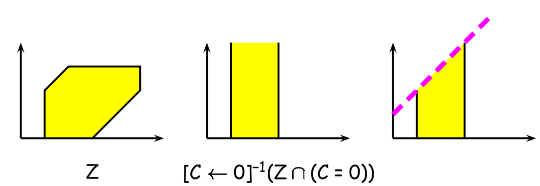
Note on the backward analysis



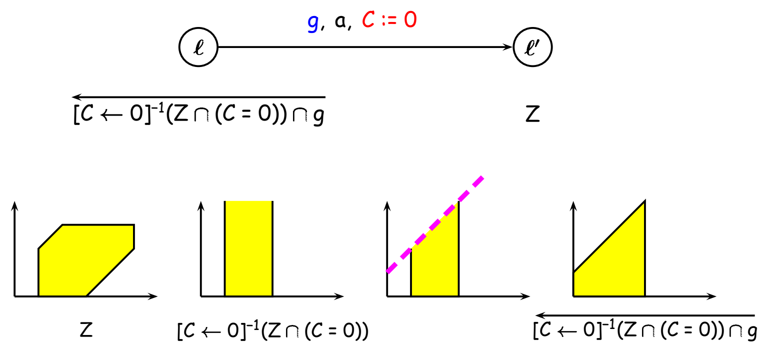


Note on the backward analysis

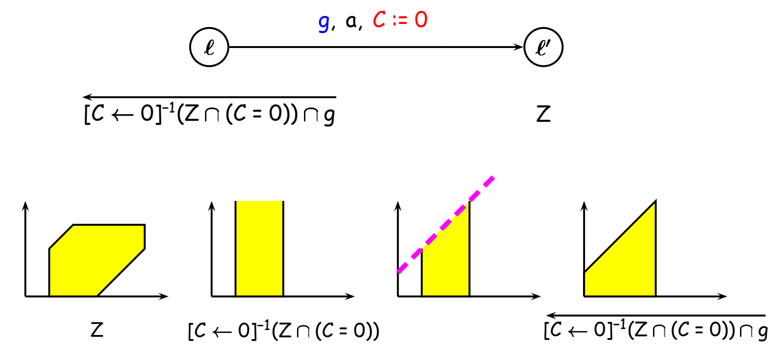




Note on the backward analysis



Note on the backward analysis



The exact backward computation terminates and is correct!

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. $v + t \in R$)
- ✔ $v' \equiv_{reg.} v$

There exists t' s.t. $v' + t' \equiv_{reg.} v + t$, 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.

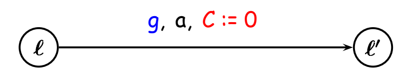
Because of the bisimulation property, we get that:

"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...

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

Forward analysis of TA

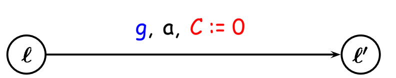


zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

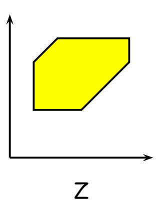
A zone is a set of valuations defined by a clock constraint

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

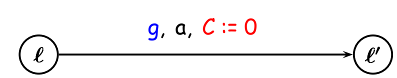
Forward analysis of TA



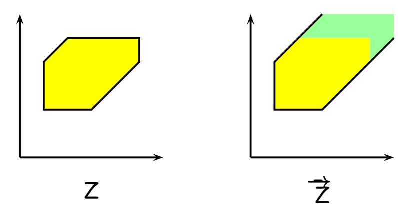
zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$



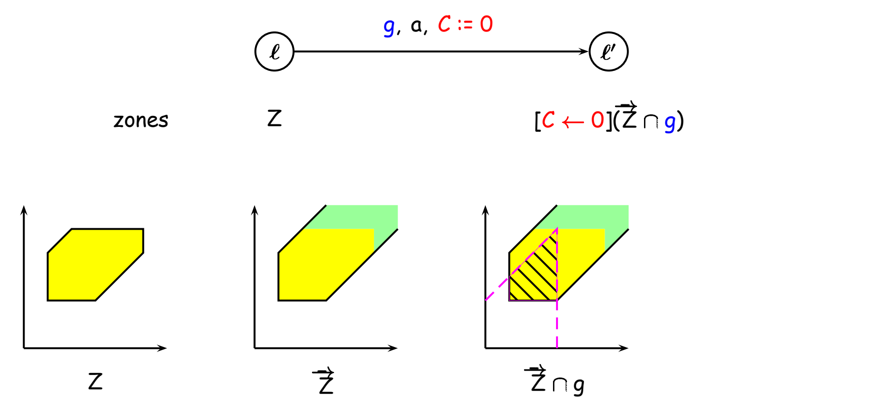
Forward analysis of TA



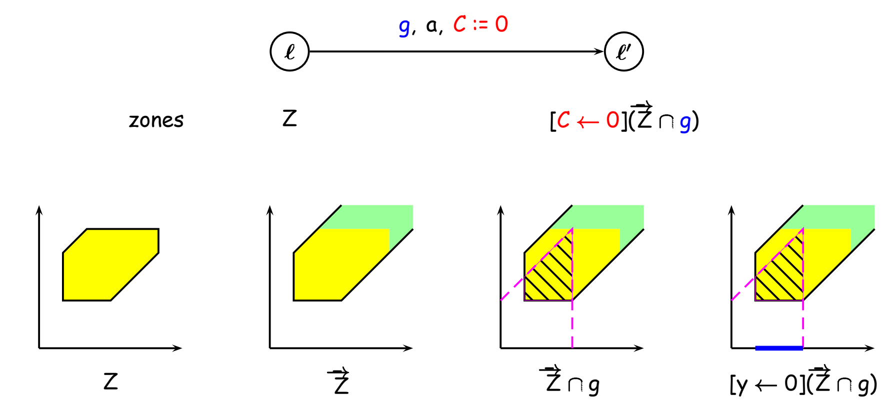
zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$



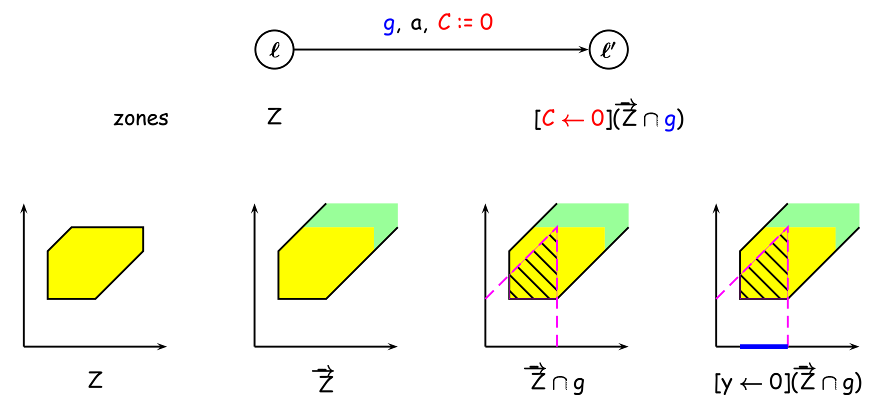
Forward analysis of TA



Forward analysis of TA

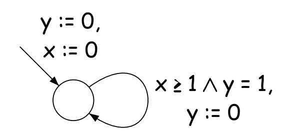


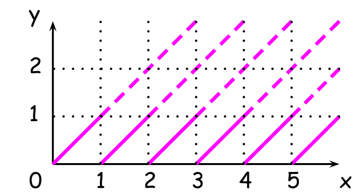
Forward analysis of TA



➜ a termination problem

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' still handled, then we don't need to handle 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' still handled, then we don't need to handle Z

➜ correct w.r.t. reachability

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

➜ correct w.r.t. reachability

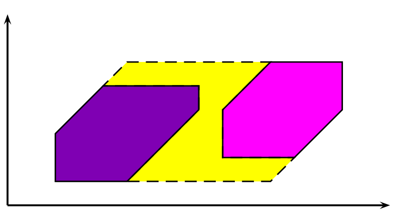
$$q \xrightarrow{g,a,C:=0} q' \quad \Longrightarrow \quad Act(q) = clocks(g) \cup (Act(q') \setminus C)$$

...

“Solutions” to this problem (cont.)

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

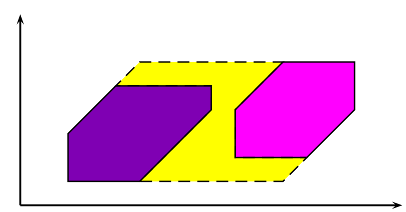
➜ “semi-correct” w.r.t. reachability



“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 [Dill 1989]

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

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \left[\begin{array}{ccc} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{array}\right] \end{array}$$

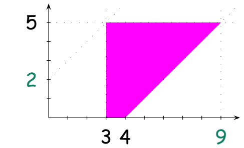
The DBM data structure

DBM (Difference Bounded Matrice) data structure [Dill 1989]

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

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \left[\begin{array}{ccc} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{array}\right] \end{array}$$

- ✔ Existence of a normal form



$$\left[\begin{array}{ccc} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{array}\right]$$

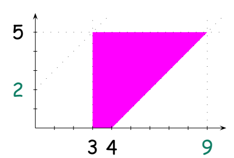
The DBM data structure

DBM (Difference Bounded Matrice) data structure [Dill 1989]

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

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \left[\begin{array}{ccc} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{array}\right] \end{array}$$

- ✔ Existence of a normal form



$$\left[\begin{array}{ccc} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{array}\right]$$

- ✔ All previous operations on zones can be computed using DBMs

The extrapolation operator

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

$$\left[\begin{array}{ccc} * & \boxed{>k} & * \\ * & * & * \\ \boxed{<-k} & * & * \end{array}\right] \rightsquigarrow \left[\begin{array}{ccc} * & \boxed{+\infty} & * \\ * & * & * \\ \boxed{-k} & * & * \end{array}\right]$$

- ✔ "intuitively", erase non-relevant constraints

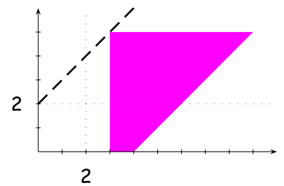
➜ ensures termination

The extrapolation operator

Fix an integer k ("∗" represents an integer between −k and +k)

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

- ✔ "intuitively", erase non-relevant constraints



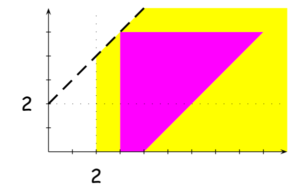
➜ ensures termination

The extrapolation operator

Fix an integer k ("∗" represents an integer between −k and +k)

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

- ✔ "intuitively", erase non-relevant constraints



➜ ensures termination

Challenge

Propose a **good** constant for the extrapolation:

- ✔ keep the correctness of the forward computation

Solution by the past: maximal constant appearing in the automaton

- ✔ Several correctness proofs can be found
- ✔ Implemented in tools like UPPAAL, KRONOS, RT-SPIN...
- ✔ Successfully used on real-life examples

Challenge

Propose a **good** constant for the extrapolation:

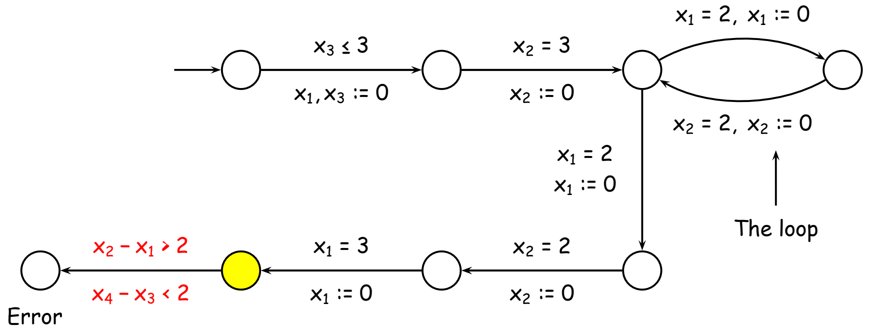
- ✔ keep the correctness of the forward computation

Solution by the past: maximal constant appearing in the automaton

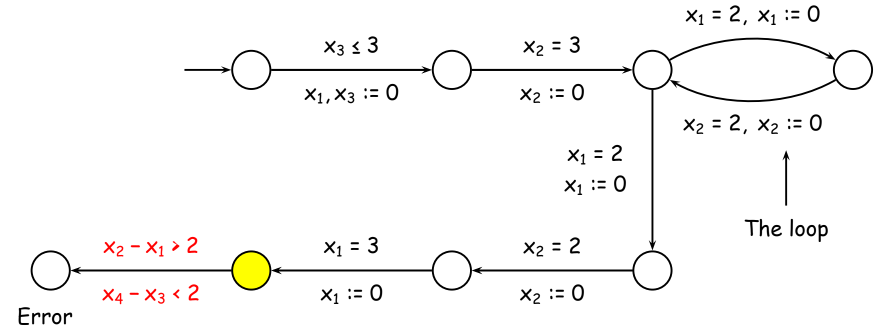
- ✔ Several correctness proofs can be found
- ✔ Implemented in tools like UPPAAL, KRONOS, RT-SPIN...
- ✔ Successfully used on real-life examples

However...

A problematic automaton

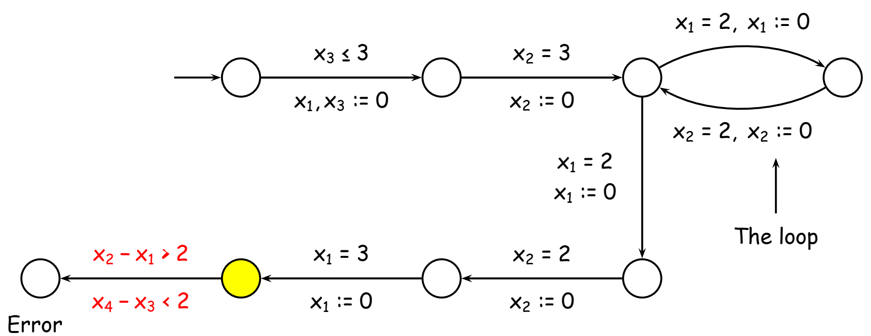


A problematic automaton

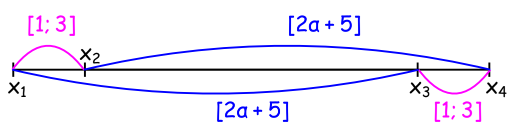


$v(x_1) = 0$
$v(x_2) = d$
$v(x_3) = 2a + 5$
$v(x_4) = 2a + 5 + d$

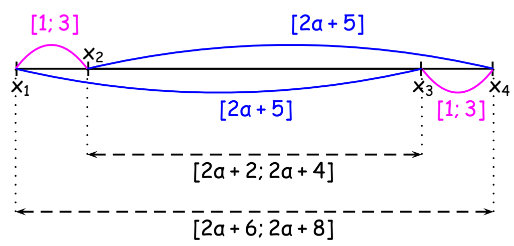
A problematic automaton



$v(x_1) = 0$
$v(x_2) = d$
$v(x_3) = 2a + 5$
$v(x_4) = 2a + 5 + d$

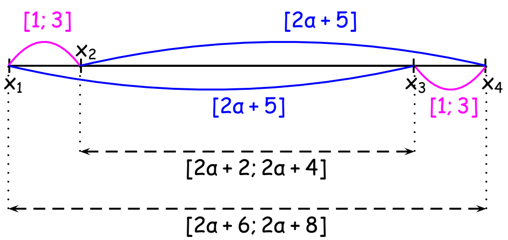


The problematic zone



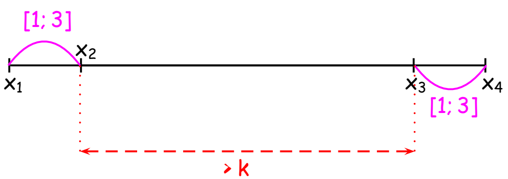
implies $x_1 - x_2 = x_3 - x_4$.

The problematic zone



implies $x_1 - x_2 = x_3 - x_4$.

If a is sufficiently large, after extrapolation:



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

General abstractions

Criteria for a good abstraction operator Abs:

General abstractions

Criteria for a good abstraction operator Abs:

- ✔ easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone

General abstractions

Criteria for a good abstraction operator Abs:

- ✔ easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone
- ✔ finiteness of the abstraction [Termination]
  {Abs(Z) | Z zone} is finite

General abstractions

Criteria for a good abstraction operator Abs:

- easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone
- finiteness of the abstraction [Termination]
  {Abs(Z) | Z zone} is finite
- completeness of the abstraction [Completeness]
  $Z \subseteq Abs(Z)$

General abstractions

Criteria for a good abstraction operator Abs:

- easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone
- finiteness of the abstraction [Termination]
  {Abs(Z) | Z zone} is finite
- completeness of the abstraction [Completeness]
  $Z \subseteq Abs(Z)$
- soundness of the abstraction [Soundness]
  the computation of $(Abs \circ Post)^*$ is correct w.r.t. reachability

General abstractions

Criteria for a good abstraction operator Abs:

- easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone
- finiteness of the abstraction [Termination]
  {Abs(Z) | Z zone} is finite
- completeness of the abstraction [Completeness]
  $Z \subseteq Abs(Z)$
- soundness of the abstraction [Soundness]
  the computation of $(Abs \circ Post)^*$ is correct w.r.t. reachability

For the previous automaton,

no abstraction operator can satisfy all these criteria!

Why that?

Assume there is a "nice" operator Abs.

The set {M DBM representing a zone Abs(Z)} is finite.

➜ k the max. constant defining one of the previous DBMs

We get that, for every zone Z,

$$Z \subseteq Extra_k(Z) \subseteq Abs(Z)$$

Problem!

Open questions:
- which conditions can be made weaker?
- find a clever termination criterium?
- use an other data structure than zones/DBMs?

What can we cling to?

Diagonal-free: only guards $x \sim c$
(no guard $x - y \sim c$)

Theorem: the classical algorithm is correct for diagonal-free timed automata.

What can we cling to?

Diagonal-free: only guards $x \sim c$
(no guard $x - y \sim c$)

Theorem: the classical algorithm is correct for diagonal-free timed automata.

General: both guards $x \sim c$ and $x - y \sim c$

Proposition: the classical algorithm is correct for timed automata that use **less than 3 clocks**.

(the constant used is bigger than the maximal constant...)

Conclusion & Further Work

- ✔ Decidability is quite well understood.
- ✔ Needs to understand better the **geometry** of the reachable state space.
- ✔ data structures for both **dense** and **discrete** parts

To be continued...

- ✔ Some other current challenges:
  - controller synthesis
  - implementability issues (program synthesis)

Bibliography

[ACD+92] Alur, Courcoubetis, Dill, Halbwachs, Wong-Toi. Minimization of Timed Transition Systems. CONCUR'92 (LNCS 630).

[AD90] Alur, Dill. Automata for Modeling Real-Time Systems. ICALP'90 (LNCS 443).

[AD94] Alur, Dill. A Theory of Timed Automata. TCS 126(2), 1994.

[AL02] Aceto, Laroussinie. Is your Model-Checker on Time? On the Complexity of Model-Checking for Timed Modal Logics. JLAP 52-53, 2002. 2002.

[BD00] Bérard, Dufourd. Timed Automata and Additive Clock Constraints. IPL 75(1-2), 2000.

[BDFP00a] Bouyer, Dufourd, Fleury, Petit. Are Timed Automata Updatable? CAV'00 (LNCS 1855).

[BDFP00b] Bouyer, Dufourd, Fleury, Petit. Expressiveness of Updatable Timed Automata. MFCS'00 (LNCS 1893).

[BDGP98] Bérard, Diekert, Gastin, Petit. Characterization of the Expressive Power of Silent Transitions in Timed Automata. Fundamenta Informaticae 36(2-3), 1998.

[BF99] Bérard, Fribourg. Automatic Verification of a Parametric Real-Time Program: the ABR Conformance Protocol. CAV'99 (LNCS 1633).

[Bouyer03] Bouyer. Untameable Timed Automata! STACS'03 (LNCS 2607).

Bibliography (cont.)

[Bouyer04] Bouyer. Forward analysis of updatable timed automata. Formal Methods in System Design 24(3),2004.

[Dill89] Dill. Timing Assumptions and Verification of Finite-State Concurrent Systems. Aut. Verif. Methods for Fin. State Sys. (LNCS 1989).

[DT98] Daws, Tripakis. Model-Checking of Real-Time Reachability Properties using Abstractions. TACAS'98 (LNCS 1384).

[DY96] Daws, Yovine. Reducing the Number of Clock Variables of Timed Automata. RTSS'96.

[LPY97] Larsen, Pettersson, Yi. UPPAAL in a Nutshell. Software Tools for Technology Transfer 1(1-2), 1997.

[Minsky67] Minsky. Computation: Finite and Infinite Machines. 1967.

[TY01] Tripakis, Yovine. Analysis of Timed Systems using Time-Abstracting Bisimulations. FMSD 18(1), 2001.

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

Kronos: `http://www-verimag.imag.fr/TEMPORISE/kronos/`

Uppaal: `http://www.uppaal.com/`