Timed Automata – From Theory to Implementation

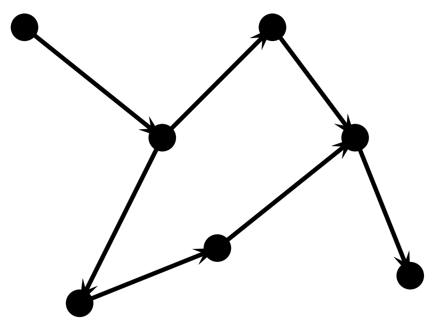
Patricia Bouyer

LSV – CNRS & ENS de Cachan

Model-checking

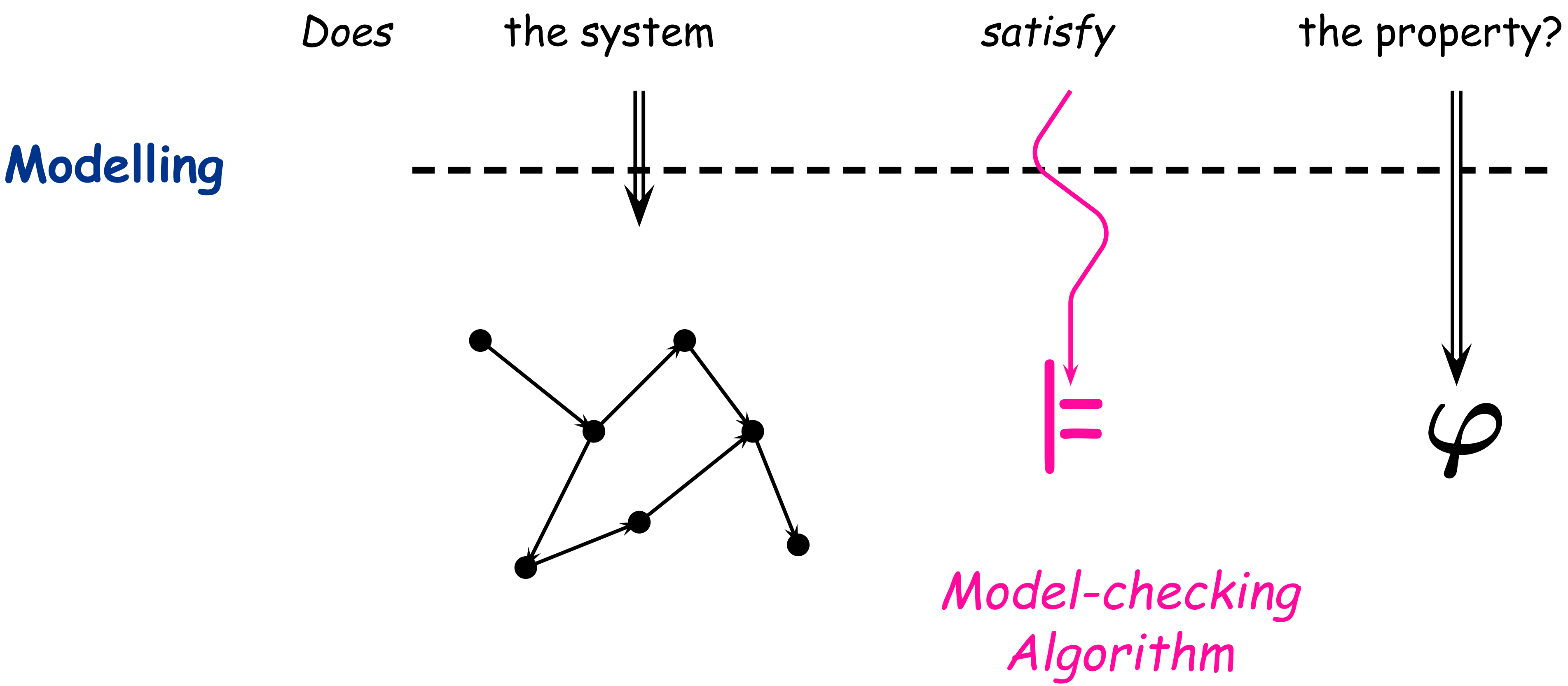
Does the system *satisfy* the property?

Modelling



φ

Model-checking



Roadmap

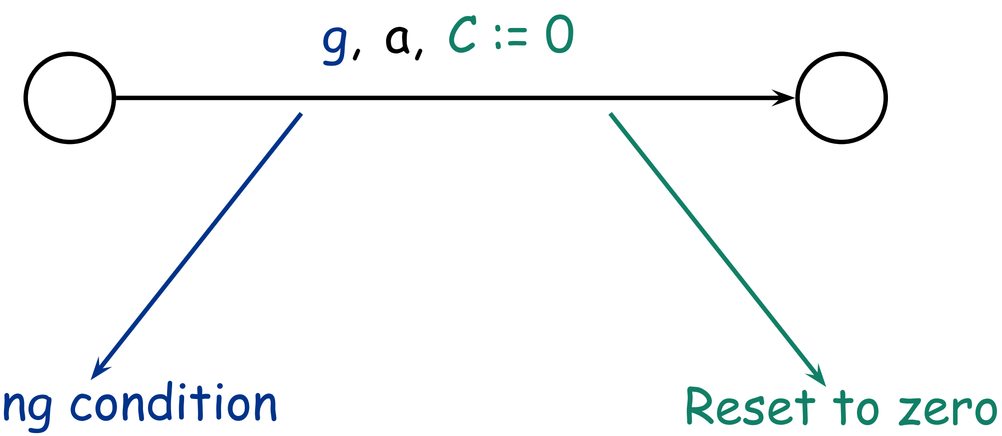
- ✓ Timed automata, decidability issues
- ✓ Some extensions of the model
- ✓ Implementation of timed automata

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:



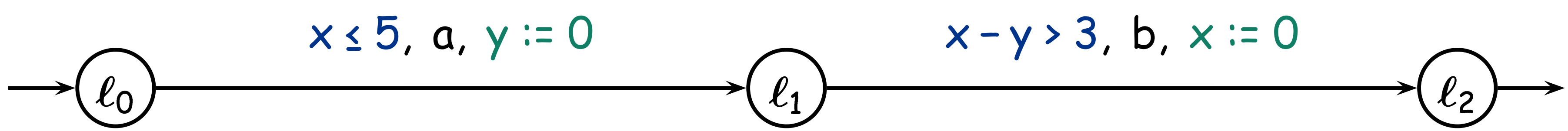
- ✔ An enabling condition (or guard) is:

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

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

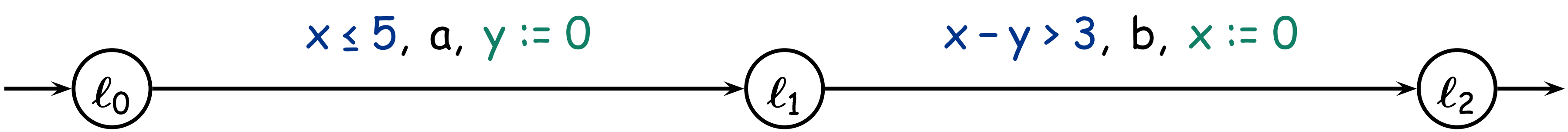
Timed automata (example)

x, y : clocks



Timed automata (example)

x, y : clocks



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

Timed automata (example)

x, y : clocks

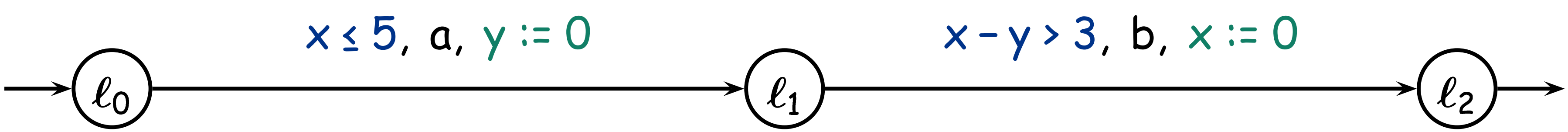
$$\rightarrow \ell_0 \xrightarrow{\;x \leq 5,\ a,\ y := 0\;} \ell_1 \xrightarrow{\;x - y > 3,\ b,\ x := 0\;} \ell_2 \rightarrow$$

	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 $(a, 4.1)(b, 5.5)$

Emptiness checking

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

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

Emptiness checking

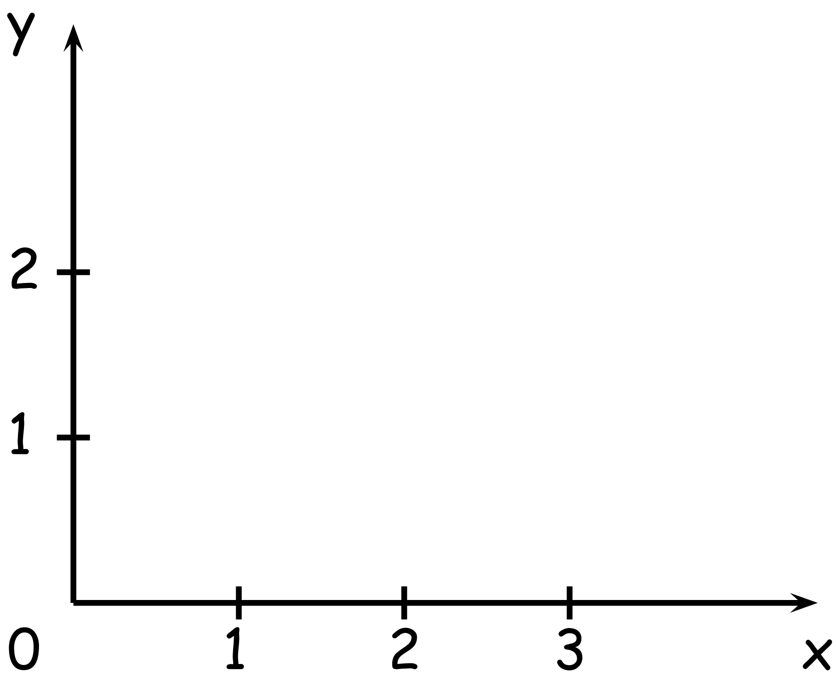
Emptiness problem: is the language accepted by a timed automaton empty?

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

Theorem: The emptiness problem for timed automata is decidable.
It is PSPACE-complete.

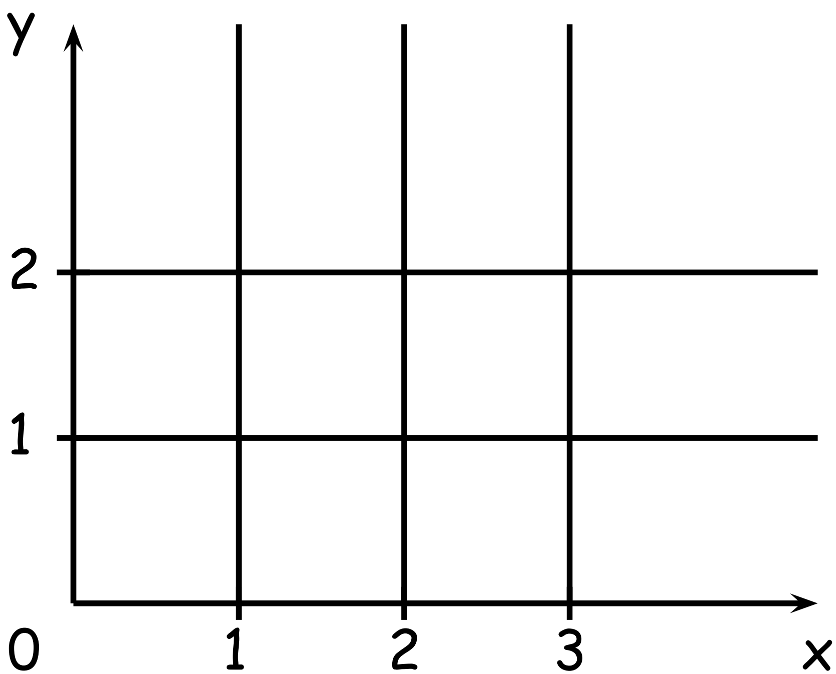
[Alur & Dill 1990's]

The region abstraction



Equivalence of finite index

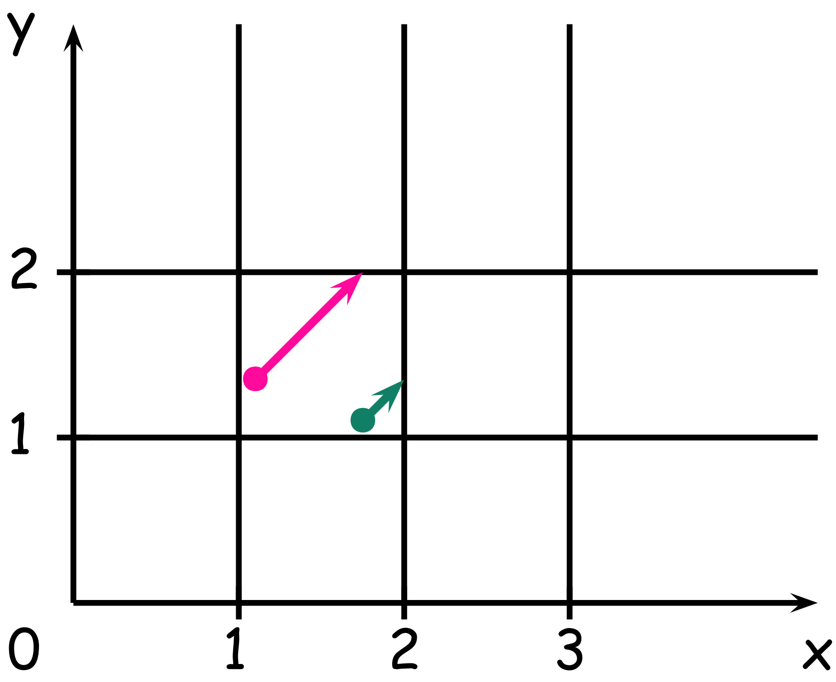
The region abstraction



Equivalence of finite index

✔ "compatibility" between regions and constraints

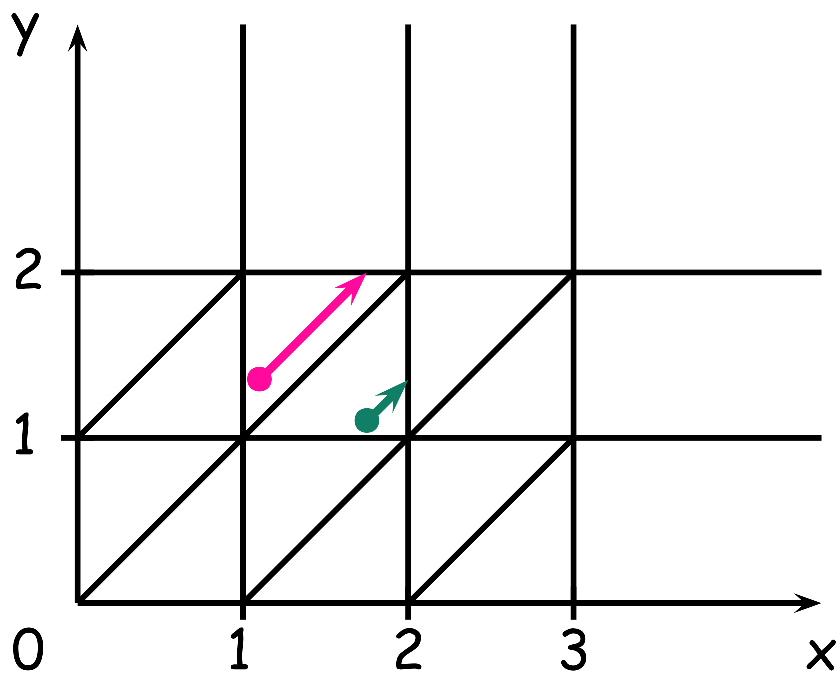
The region abstraction



Equivalence of finite index

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

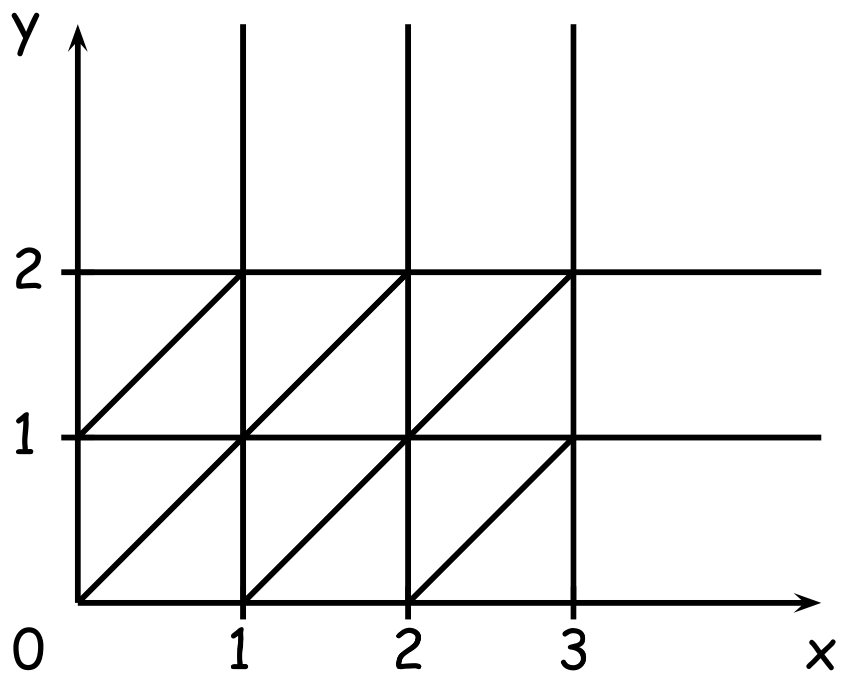
The region abstraction



Equivalence of finite index

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

The region abstraction

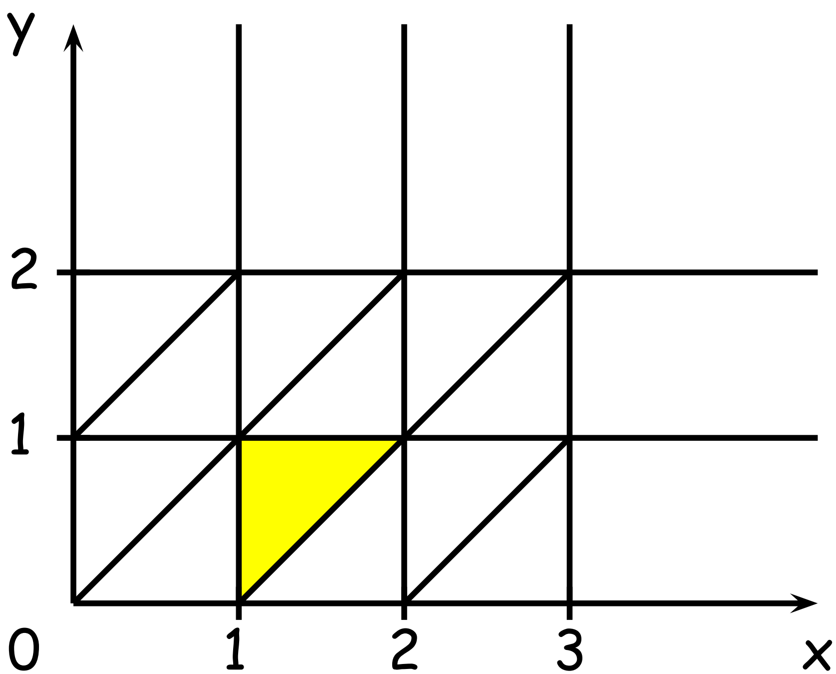


Equivalence of finite index

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

➜ a bisimulation property

The region abstraction



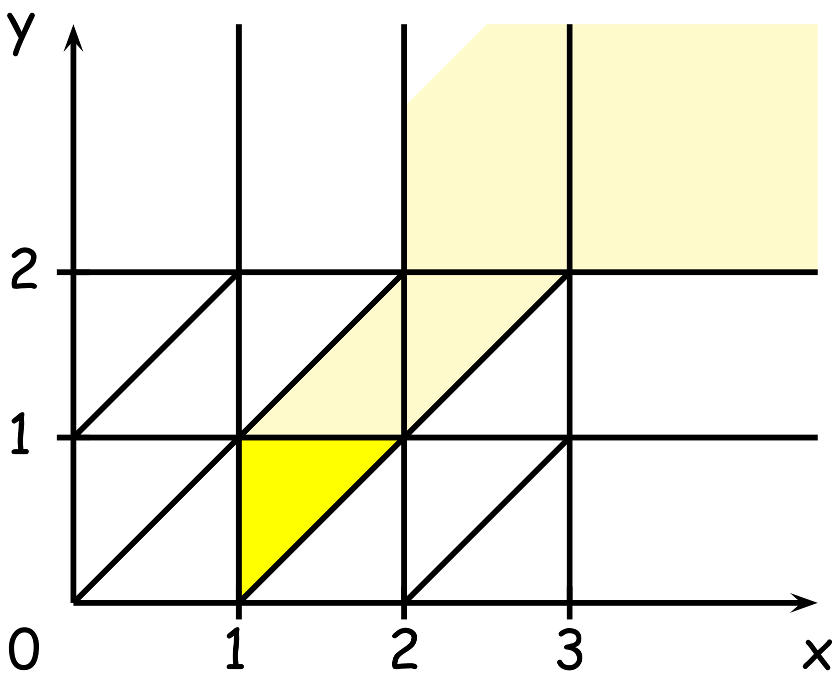
Equivalence of finite index

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:

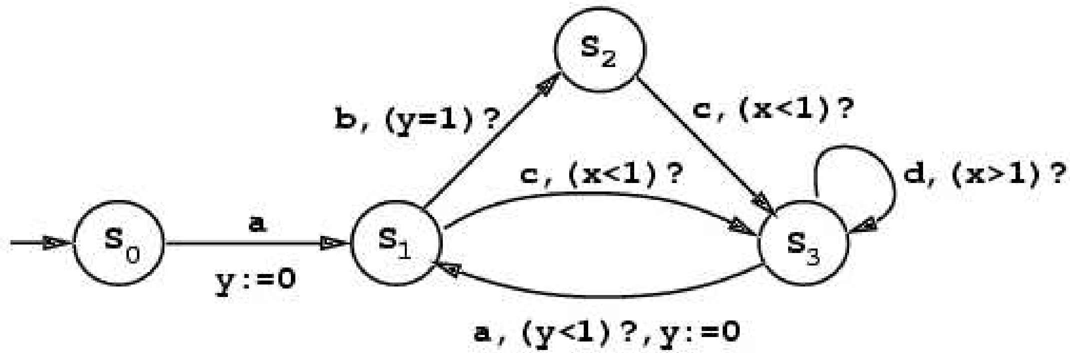
$(\ell, R) \xrightarrow{a} (\ell', R')$ if there exists $R'' \in \mathrm{Succ}_t^*(R)$ s.t.

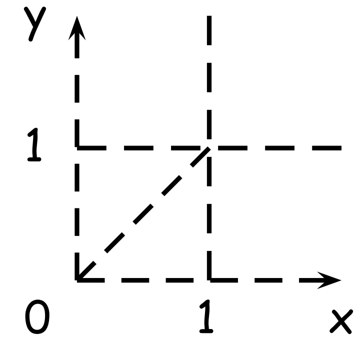
- ✔ $R'' \subseteq g$
- ✔ $[C \leftarrow 0]R'' \subseteq R'$

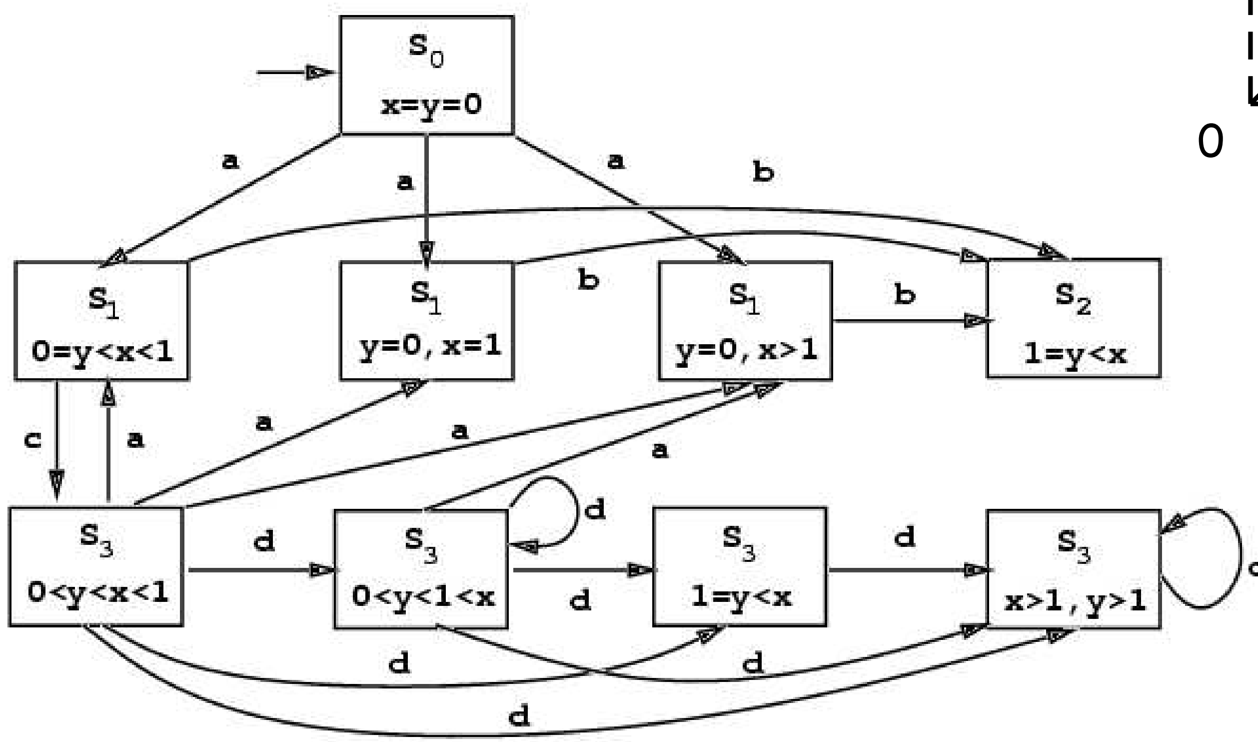
$\mathcal{L}(\text{reg. aut.}) = \text{UNTIME}(\mathcal{L}(\text{timed aut.}))$

where $\text{UNTIME}((a_1, t_1)(a_2, t_2)\ldots) = a_1 a_2 \ldots$

An example [AD 90's]







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

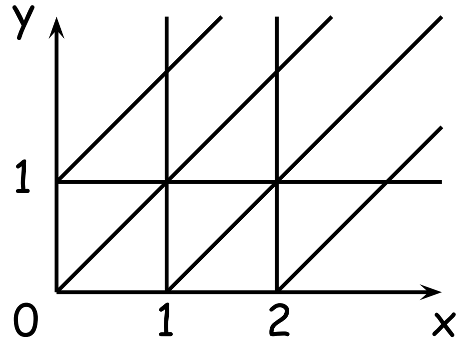
Some extensions of the model

- ✔ 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

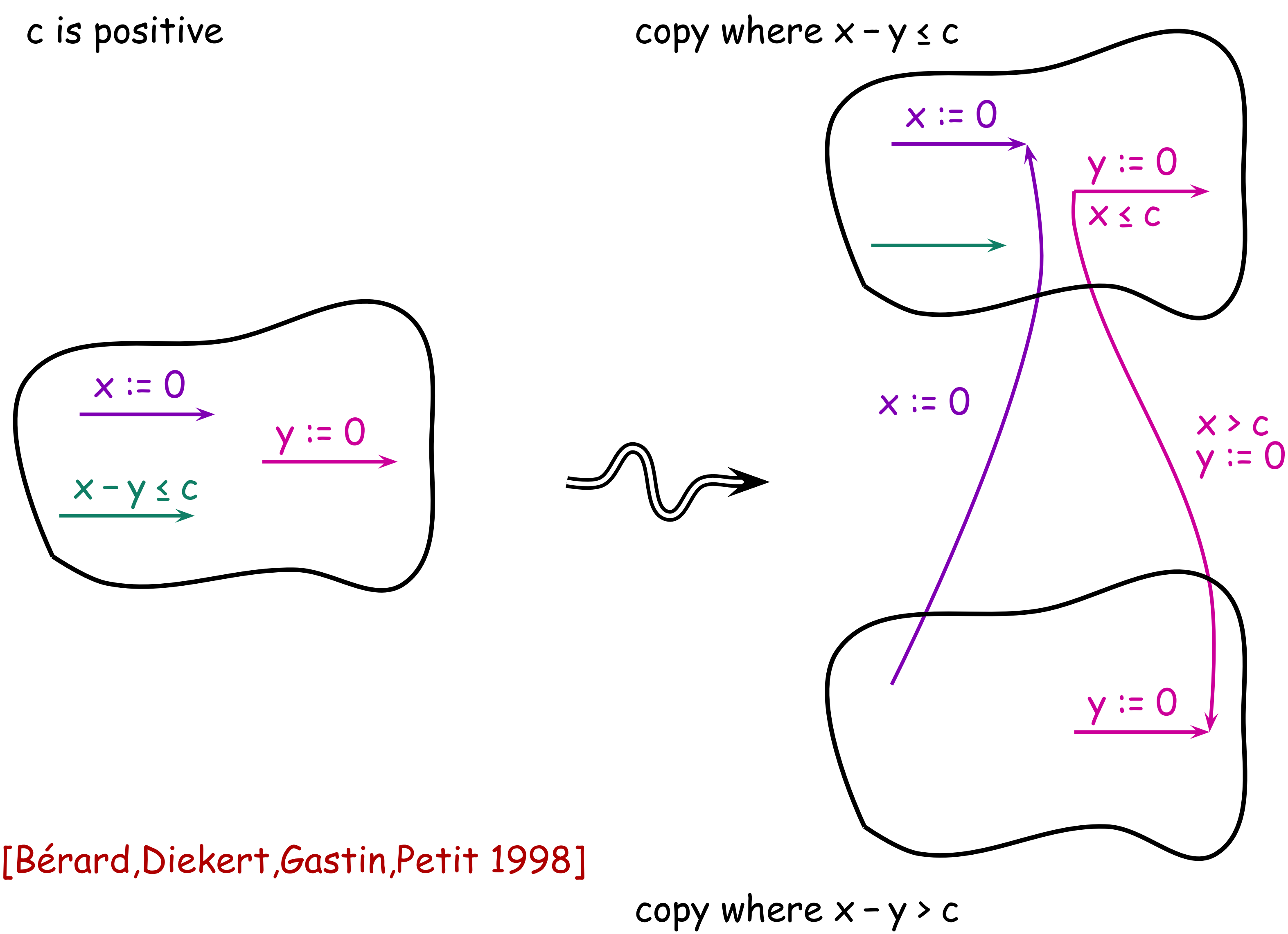
$x - y \sim c$ and $x \sim c$

- ✔ **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.

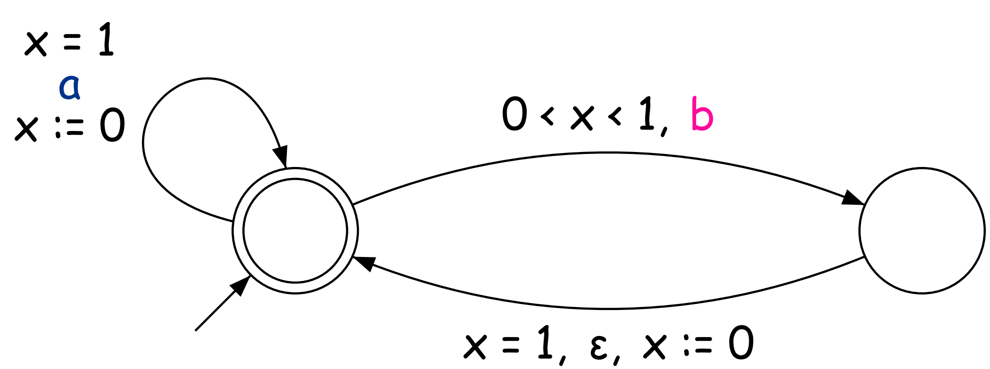
Adding silent actions

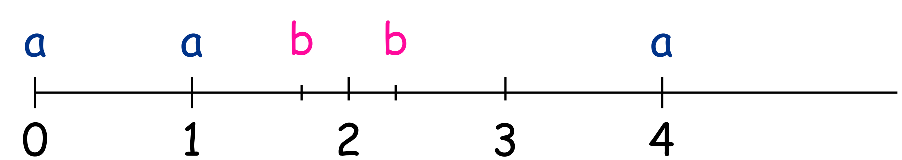
$$\xrightarrow{g,\ \varepsilon,\ C := 0}$$

[Bérard,Diekert,Gastin,Petit 1998]

- **Decidability:** yes (actions has no influence on the previous construction)

- **Expressiveness:** strictly more expressive!

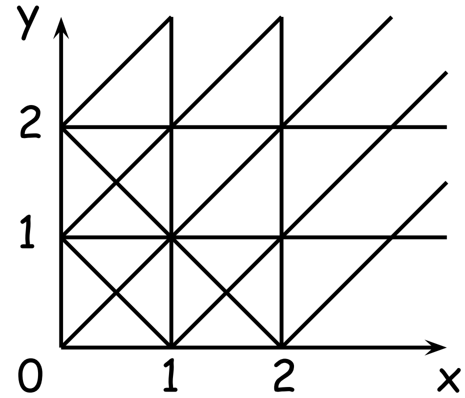




Adding constraints of the form $x + y \sim c$

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

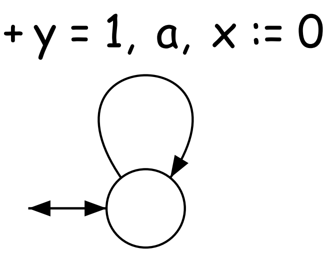
- ✔ **Decidability:** - for two clocks, decidable using the abstraction



  - for four clocks (or more), undecidable!

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

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



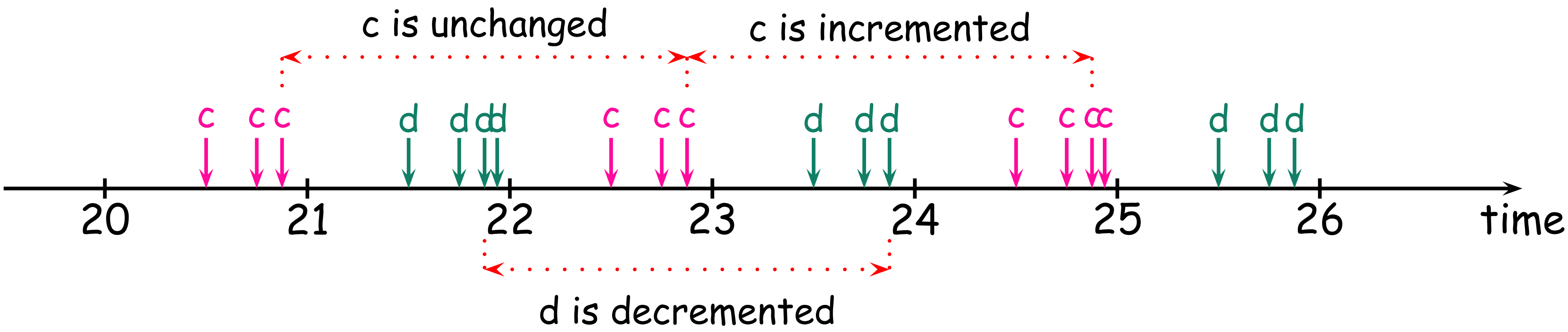
The two-counter machine

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

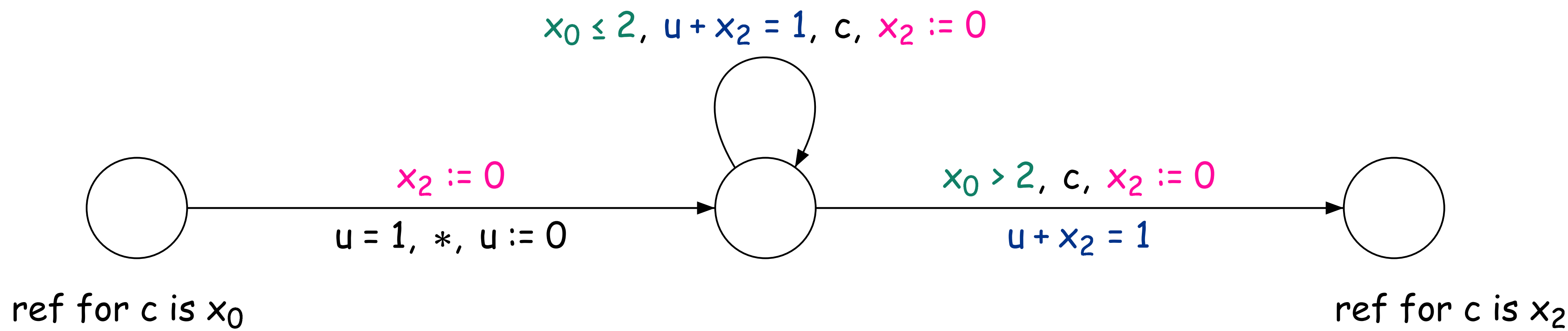
We will use 4 clocks:
- u, "tic" clock (each time unit)
- x_0, x_1, x_2: reference clocks for the two counters

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

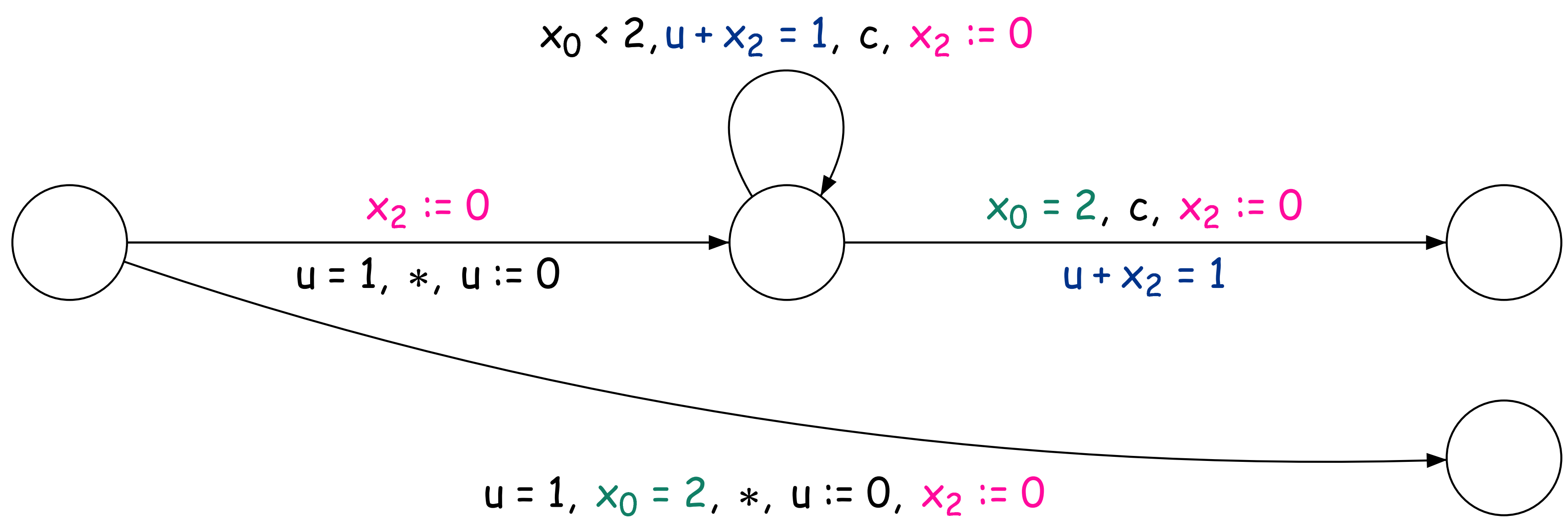
[Bérard,Dufourd 2000]

Undecidability proof (cont.)

- Increment of counter c:

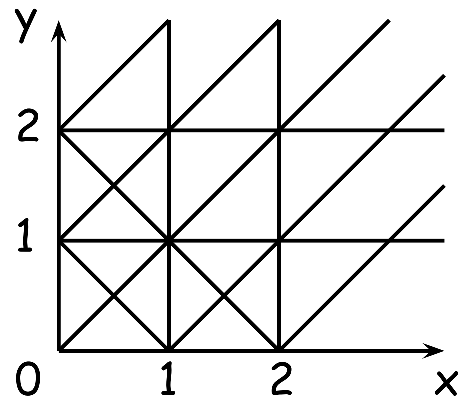


- Decrement of counter c:



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

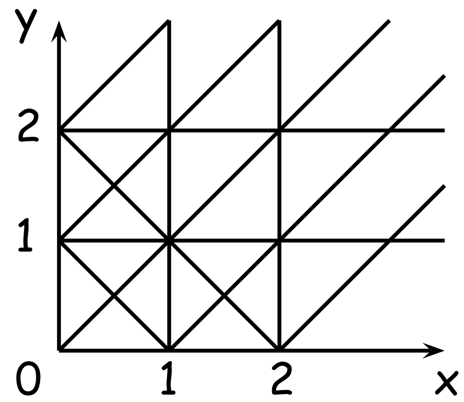
- ✔ Two clocks: decidable! using the abstraction



- ✔ 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)

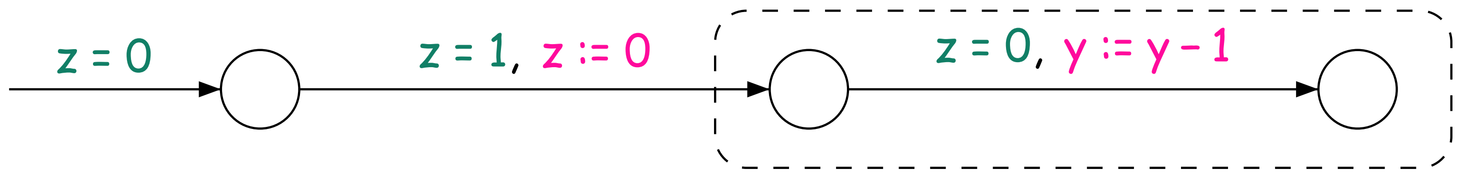
Adding new operations on clocks

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

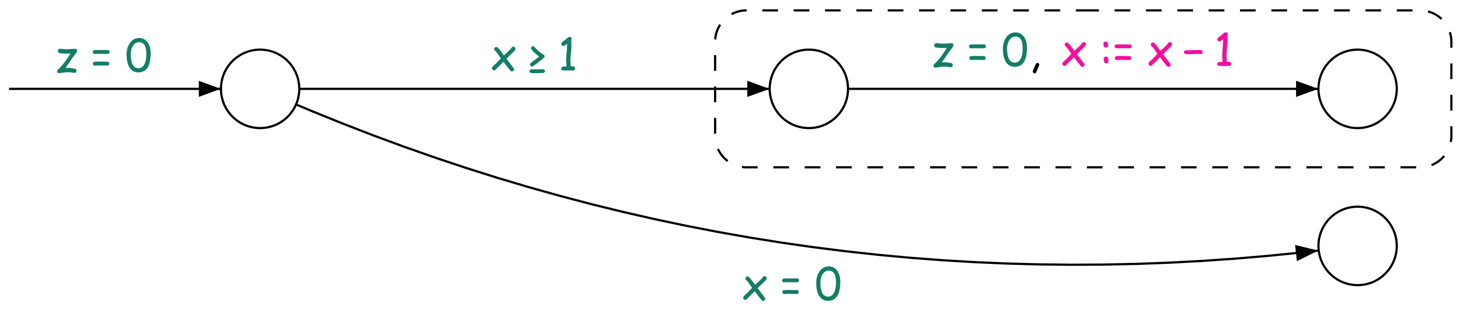
- ✔ The general model is undecidable.

  (simulation of a two-counter machine)

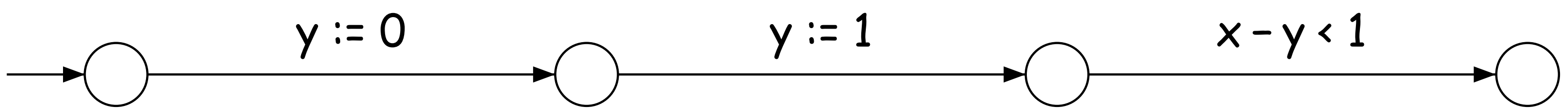
- ✔ Only decrementation also leads to undecidability
  - **Incrementation of counter x**



  - **Decrementation of counter x**



Decidability



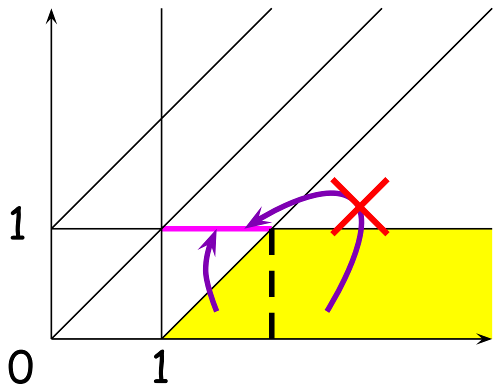


image by y := 1

➜ the bisimulation property is not met

The classical region automaton construction is not correct.

Decidability (cont.)

$\mathcal{A} \quad \rightsquigarrow \quad$ Diophantine linear inequations system

$\rightsquigarrow \quad$ is there a solution?

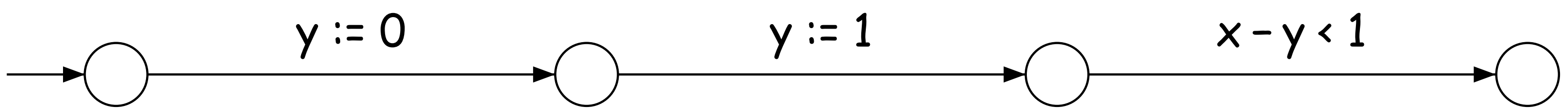
$\rightsquigarrow \quad$ if yes, belongs to a decidable class

Examples:

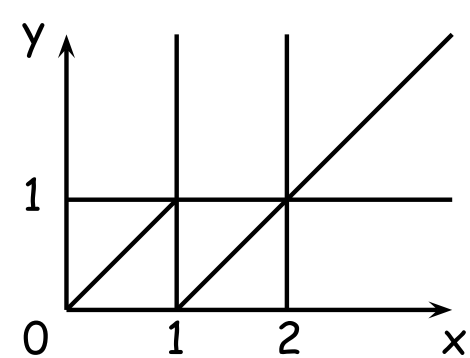
- ✔ constraint $x \sim c$ — $c \leq max_x$
- ✔ constraint $x - y \sim c$ — $c \leq max_{x,y}$
- ✔ update $x :\sim y + c$ — $max_x \leq max_y + c$
  and for each clock z, $max_{x,z} \geq max_{y,z} + c$, $max_{z,x} \geq max_{z,y} - c$
- ✔ update $x :< c$ — $c \leq max_x$
  and for each clock z, $max_z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

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

Decidability (cont.)



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

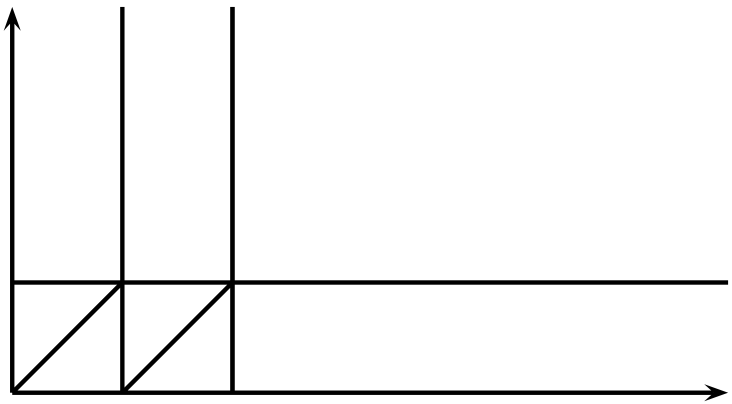


The bisimulation property is met.

What's wrong when undecidable?

Decrementation $x := x - 1$

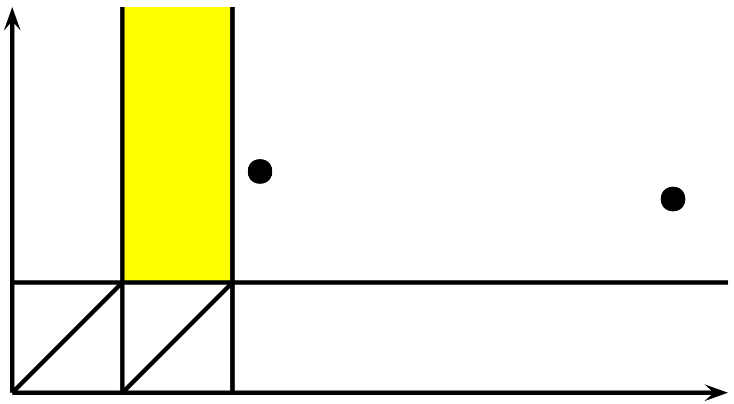
$max_x \leq max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

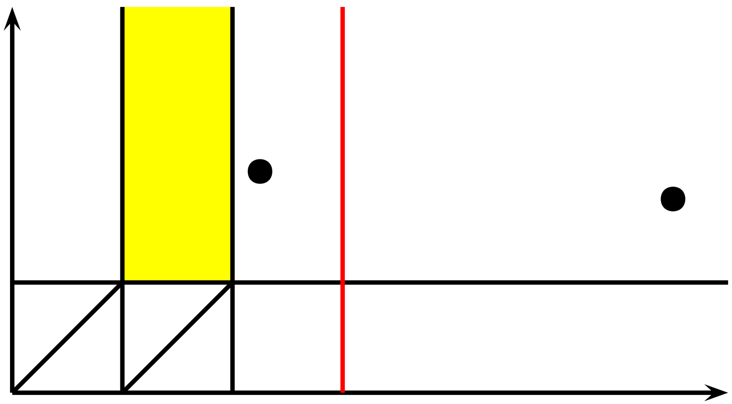
$max_x \leq max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

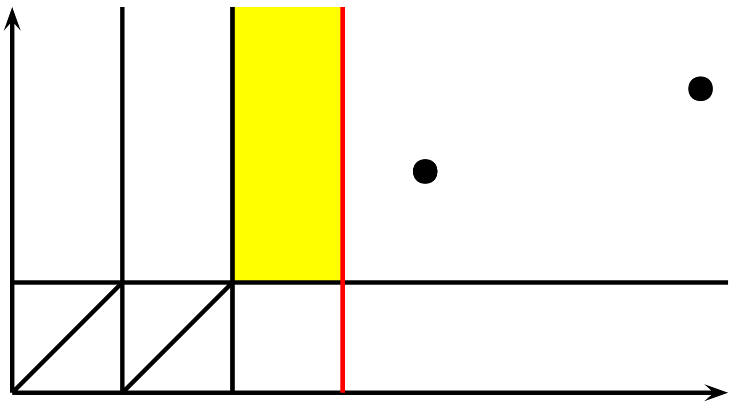
$max_x \leq max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

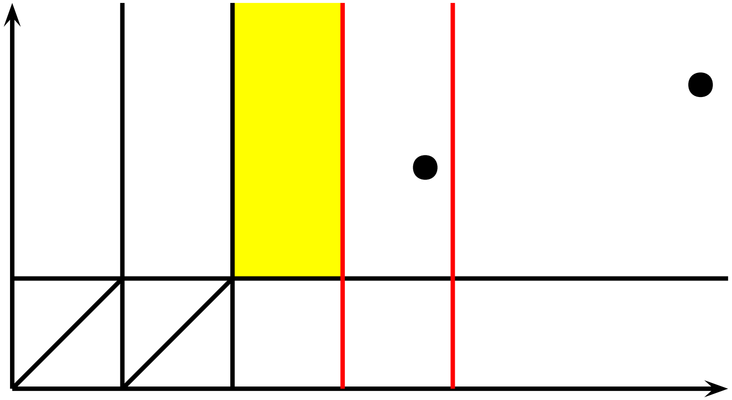
$max_x \leq max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

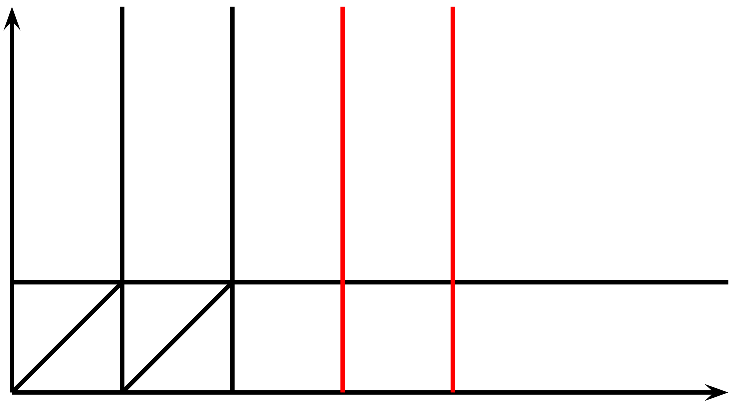
$max_x \leq max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

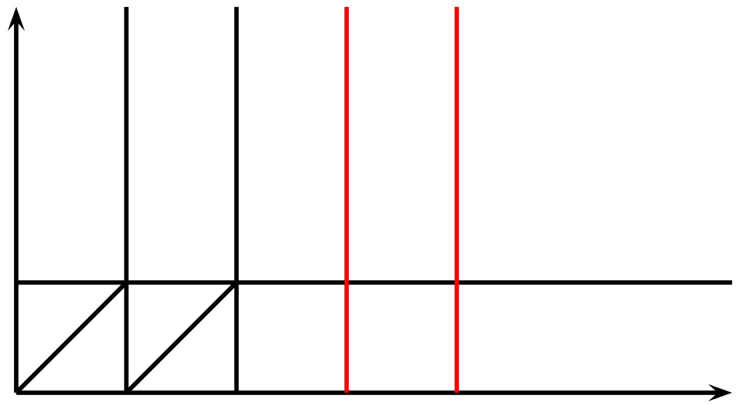
$max_x \leq max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

$max_x \leq max_x - 1$



etc...

Decidability (cont.)

<table>
<tr><th></th><th>Diagonal-free constraints</th><th>General constraints</th></tr>
<tr><td>$x := c, x := y$</td><td rowspan="3">PSPACE-complete</td><td>PSPACE-complete</td></tr>
<tr><td>$x := x + 1$</td><td rowspan="3">Undecidable</td></tr>
<tr><td>$x := y + c$</td></tr>
<tr><td>$x := x - 1$</td><td>Undecidable</td></tr>
<tr><td>$x :< c$</td><td rowspan="4">PSPACE-complete</td><td>PSPACE-complete</td></tr>
<tr><td>$x :> c$</td><td rowspan="4">Undecidable</td></tr>
<tr><td>$x :\sim y + c$</td></tr>
<tr><td>$y + c <: x :< y + d$</td></tr>
<tr><td>$y + c <: x :< z + d$</td><td>Undecidable</td></tr>
</table>

[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

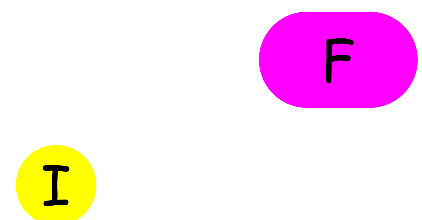
Algorithms for "minimizing" the region automaton have been proposed...

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

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

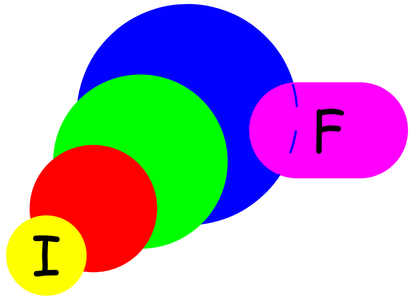
Reachability analysis

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



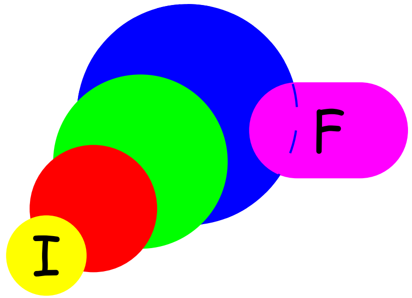
Reachability analysis

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



Reachability analysis

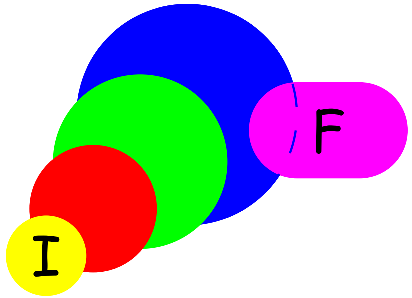
- **forward analysis algorithm:**
  compute the successors of initial configurations



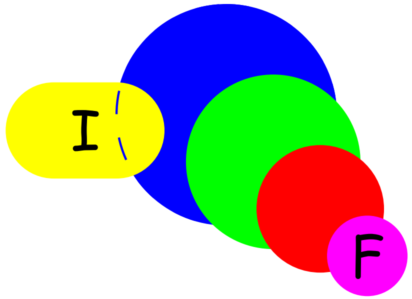
- **backward analysis algorithm:**
  compute the predecessors of final configurations

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

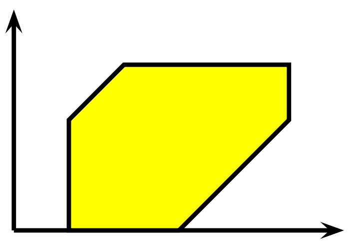
$$\ell \xrightarrow{\;g,\ a,\ C := 0\;} \ell'$$

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

Note on the backward analysis

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

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

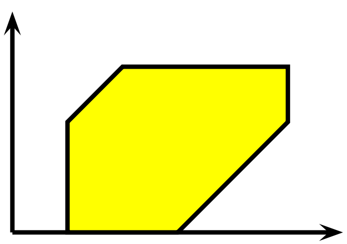


Z

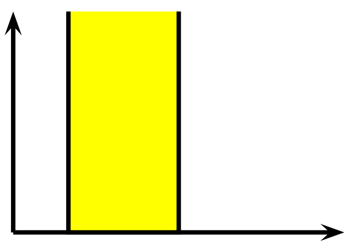
Note on the backward analysis

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

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

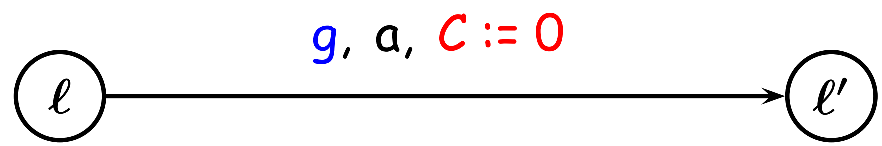


Z

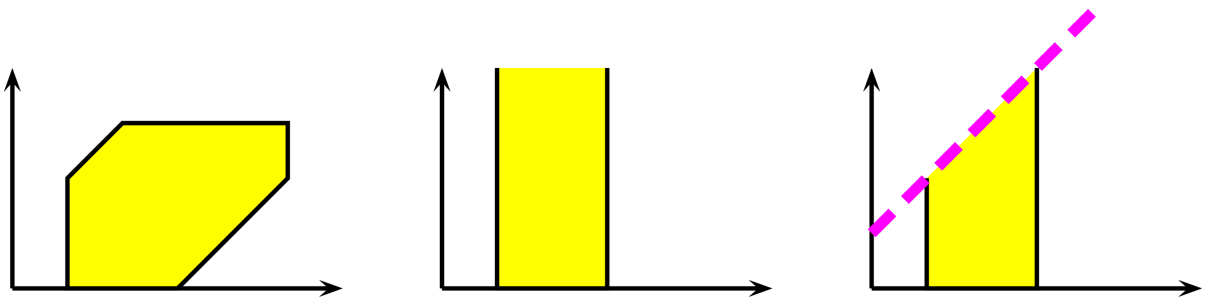


$[C \leftarrow 0]^{-1}(Z \cap (C = 0))$

Note on the backward analysis

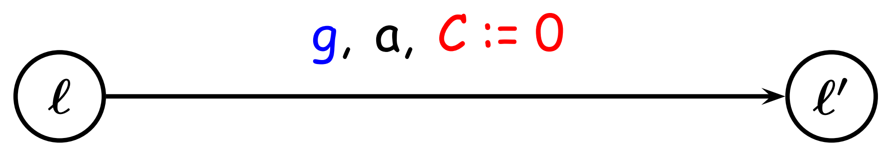


$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\qquad$ Z

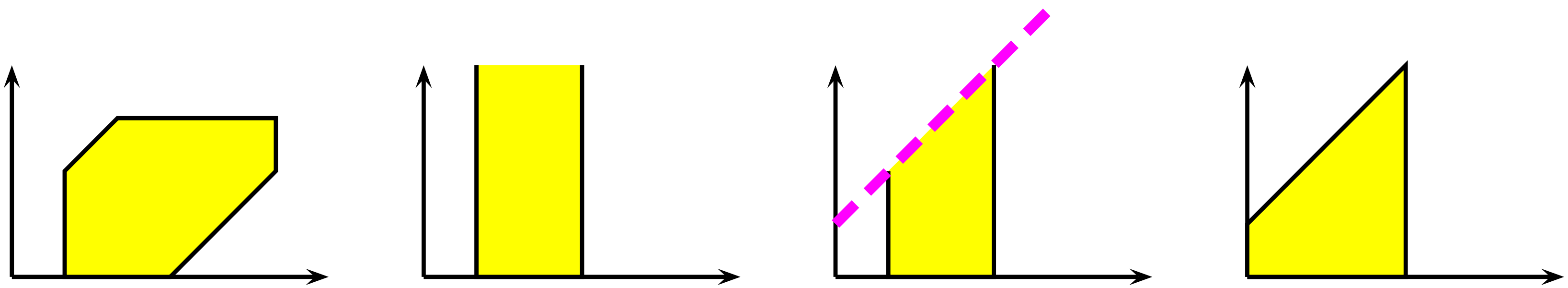


Z $\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$

Note on the backward analysis

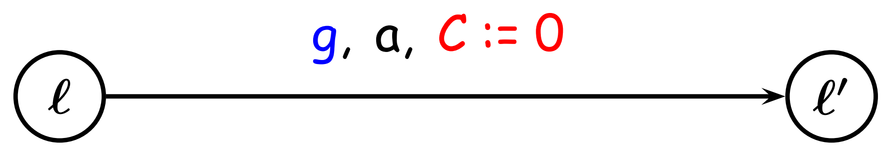


$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\qquad\qquad$ Z

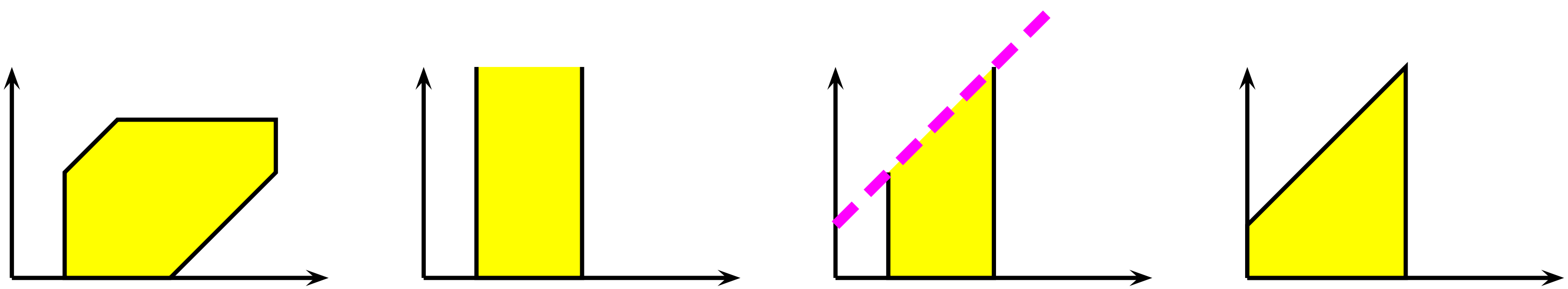


Z $\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$ $\qquad$ $\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$

Note on the backward analysis



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



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

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

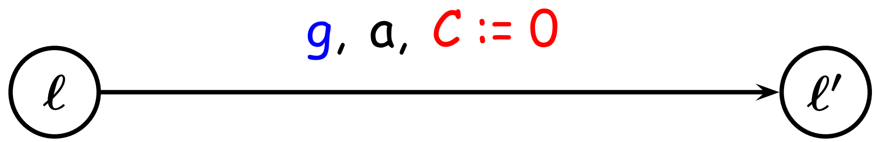
$$\ell \xrightarrow{\;g,\ a,\ C := 0\;} \ell'$$

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

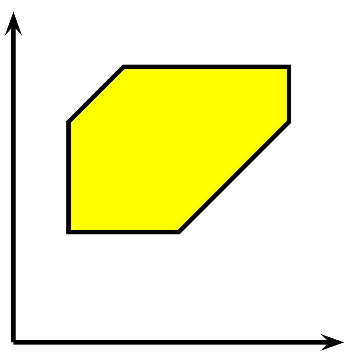
A zone is a set of valuations defined by a clock constraint

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

Forward analysis of TA



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

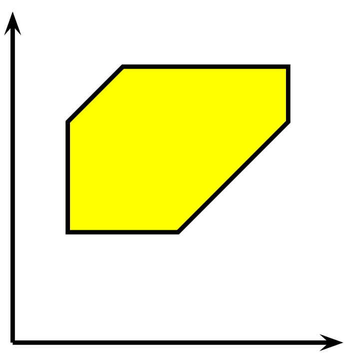


Z

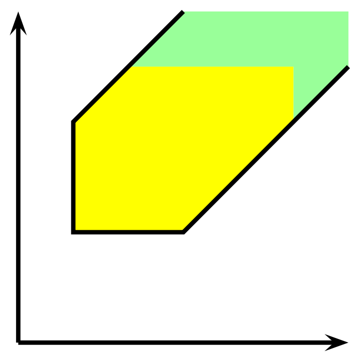
Forward analysis of TA

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

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



Z

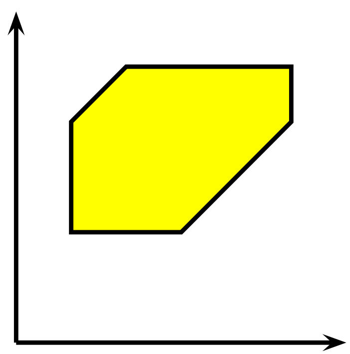


$\overrightarrow{Z}$

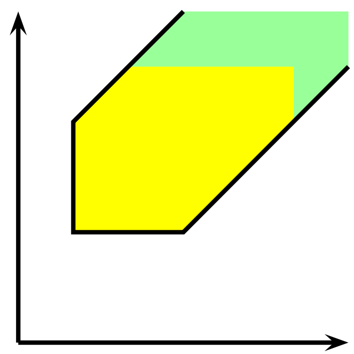
Forward analysis of TA

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

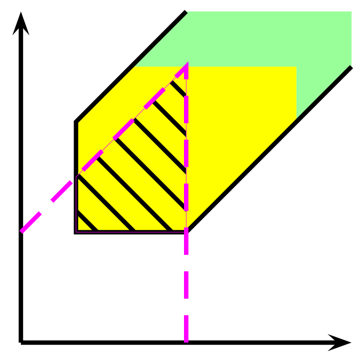
zones $\qquad Z \qquad\qquad [C \leftarrow 0](\overrightarrow{Z} \cap g)$



Z

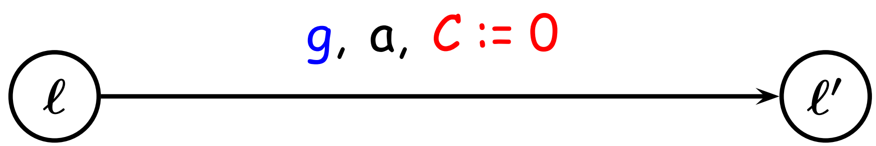


$\overrightarrow{Z}$

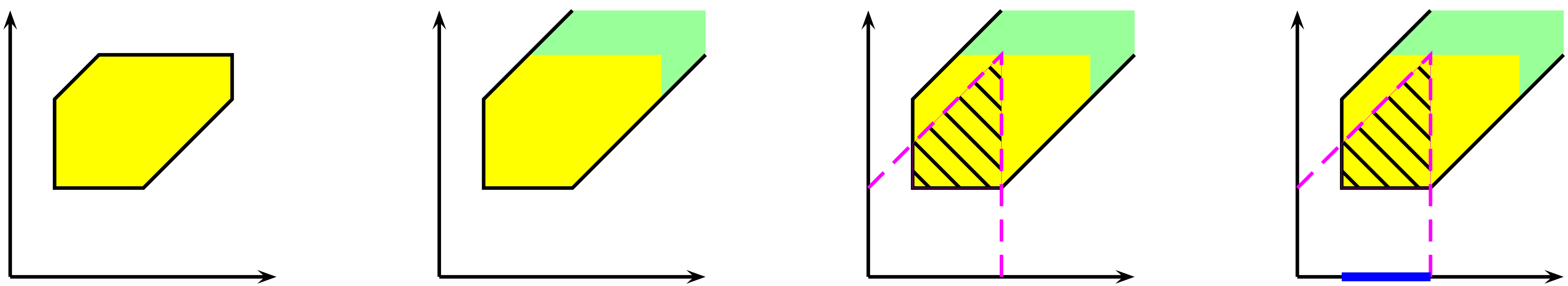


$\overrightarrow{Z} \cap g$

Forward analysis of TA

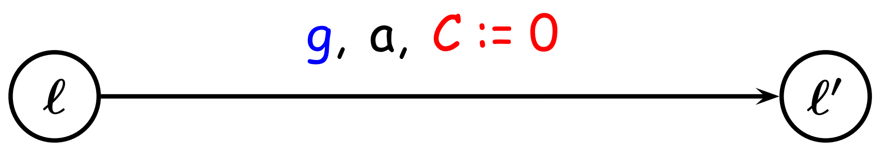


zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

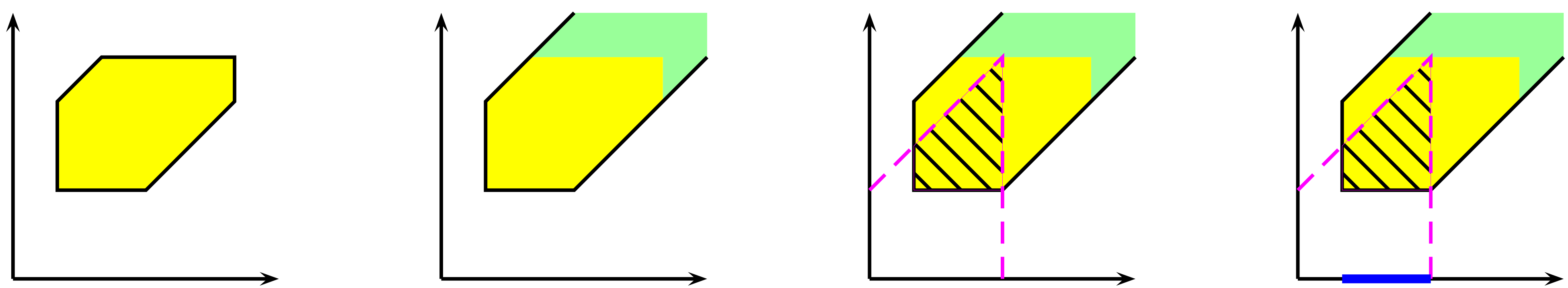


$Z \qquad \overrightarrow{Z} \qquad \overrightarrow{Z} \cap g \qquad [y \leftarrow 0](\overrightarrow{Z} \cap g)$

Forward analysis of TA



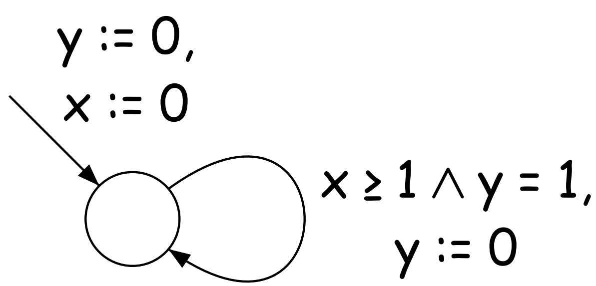
zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

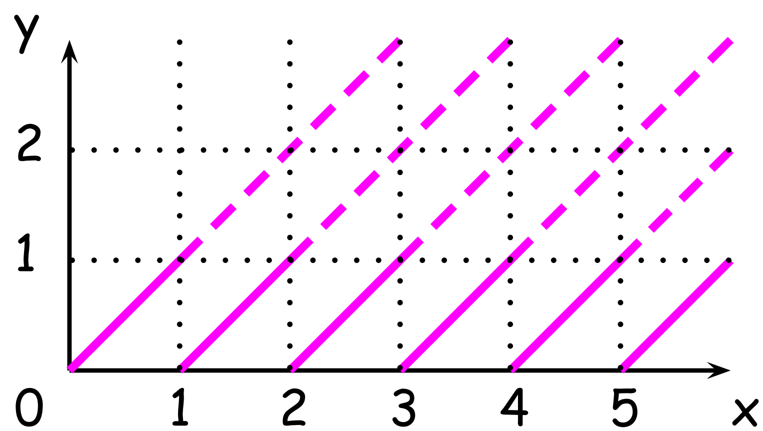


$Z \qquad \overrightarrow{Z} \qquad \overrightarrow{Z} \cap g \qquad [y \leftarrow 0](\overrightarrow{Z} \cap g)$

➜ 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

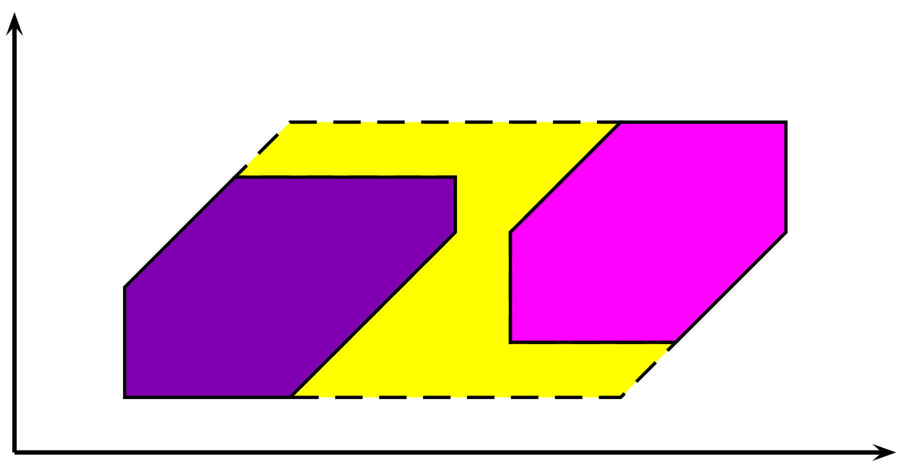
$$q \xrightarrow{g,a,C:=0} q' \quad \Longrightarrow \quad Act(q) = clocks(g) \cup (Act(q') \setminus C)$$

...

"Solutions" to this problem (cont.)

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

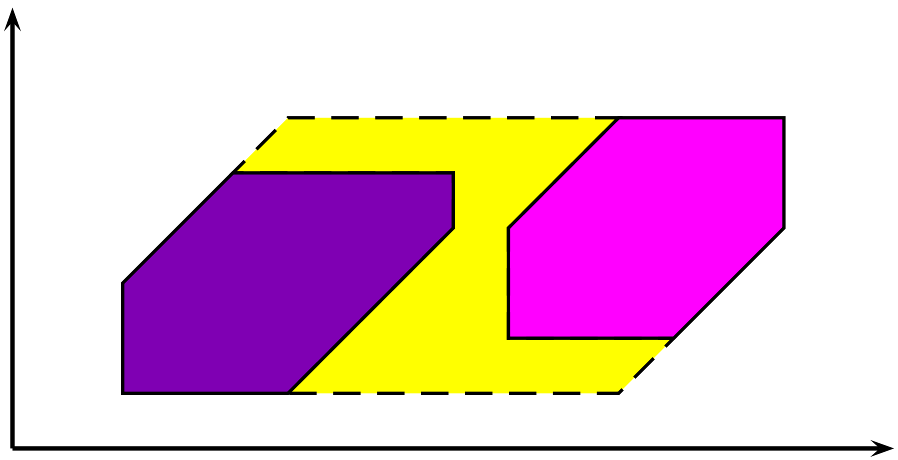
➜ "semi-correct" w.r.t. reachability



“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]

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

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

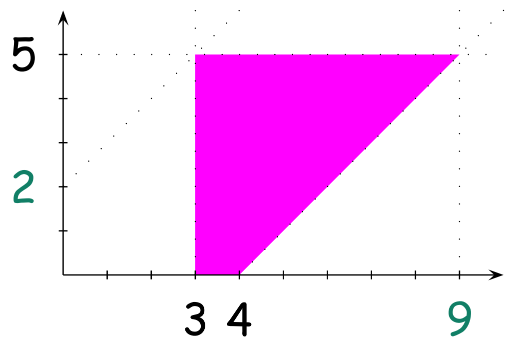
The DBM data structure

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

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

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

✔ Existence of a normal form



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

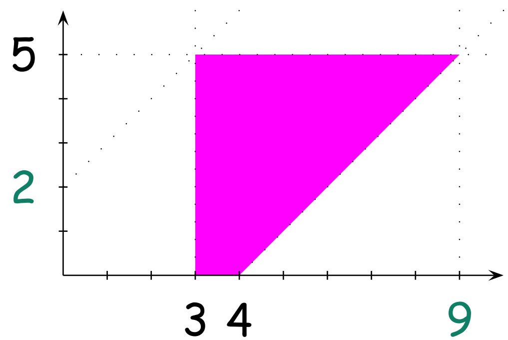
The DBM data structure

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

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

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

- ✔ Existence of a normal form



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

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

The extrapolation operator

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

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

✔ "intuitively", erase non-relevant constraints

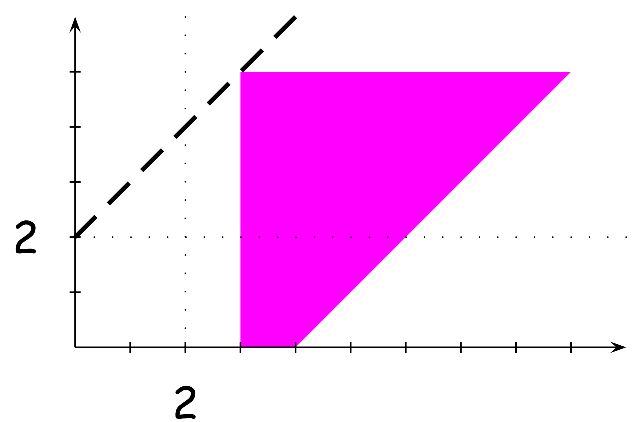
➜ *ensures termination*

The extrapolation operator

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

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

✔ "intuitively", erase non-relevant constraints



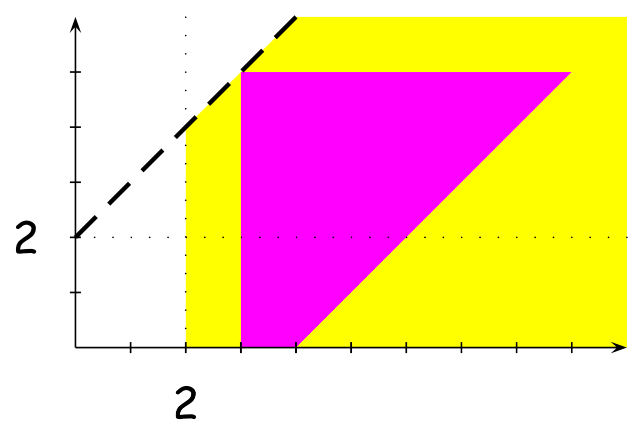
➜ *ensures termination*

The extrapolation operator

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

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

✔ "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:

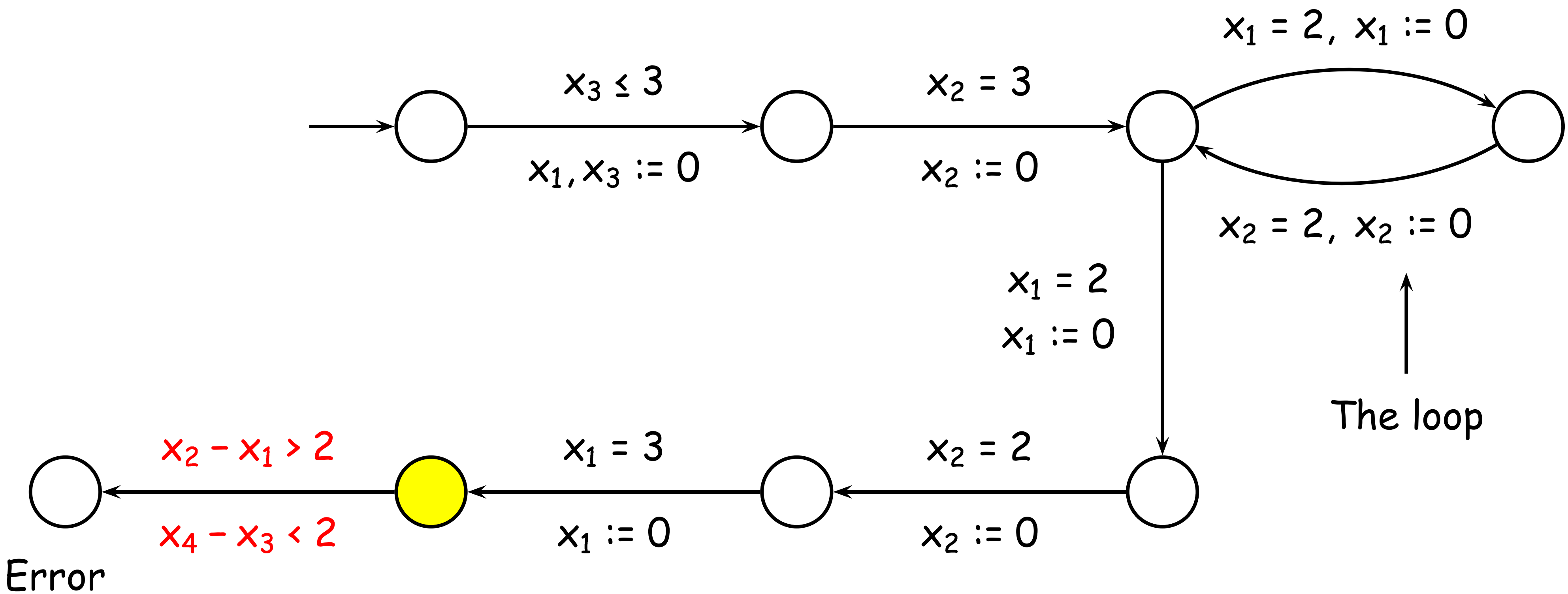
- ✔ keep the correctness of the forward computation

Solution by the past: maximal constant appearing in the automaton

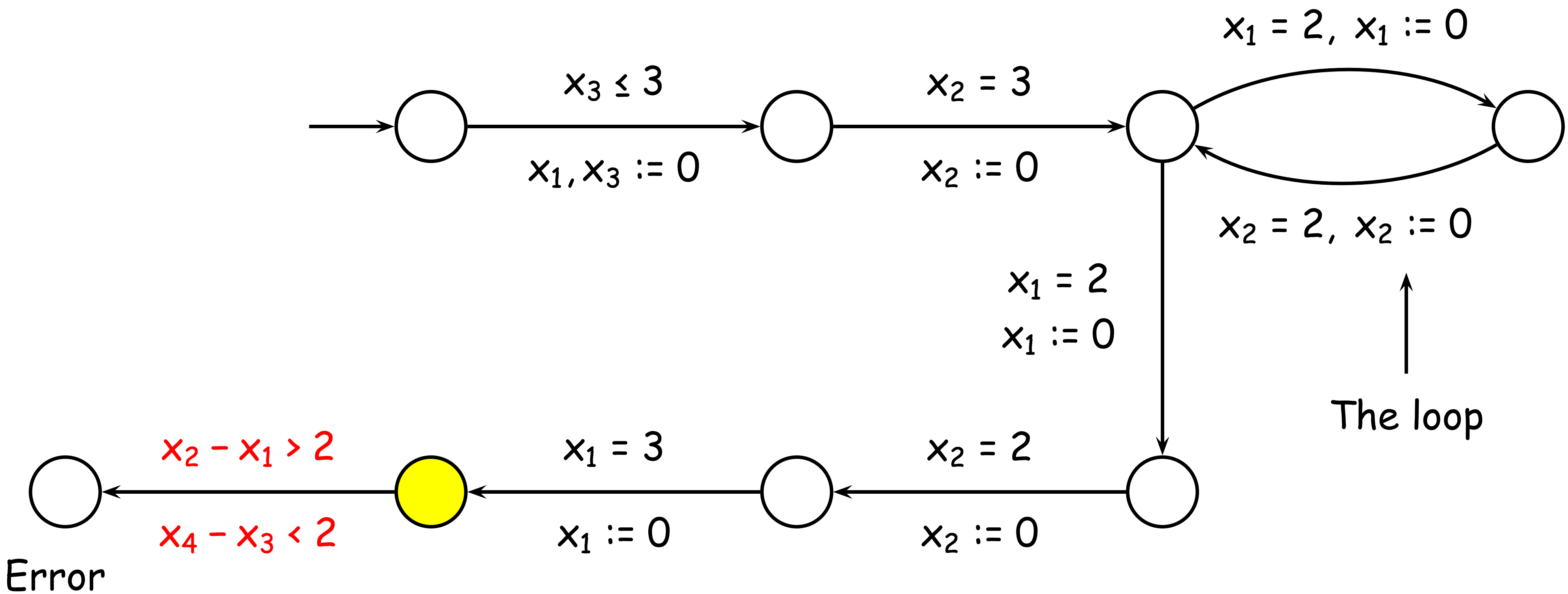
- ✔ Several correctness proofs can be found
- ✔ Implemented in tools like UPPAAL, KRONOS, RT-SPIN...
- ✔ Successfully used on real-life examples

However...

A problematic automaton

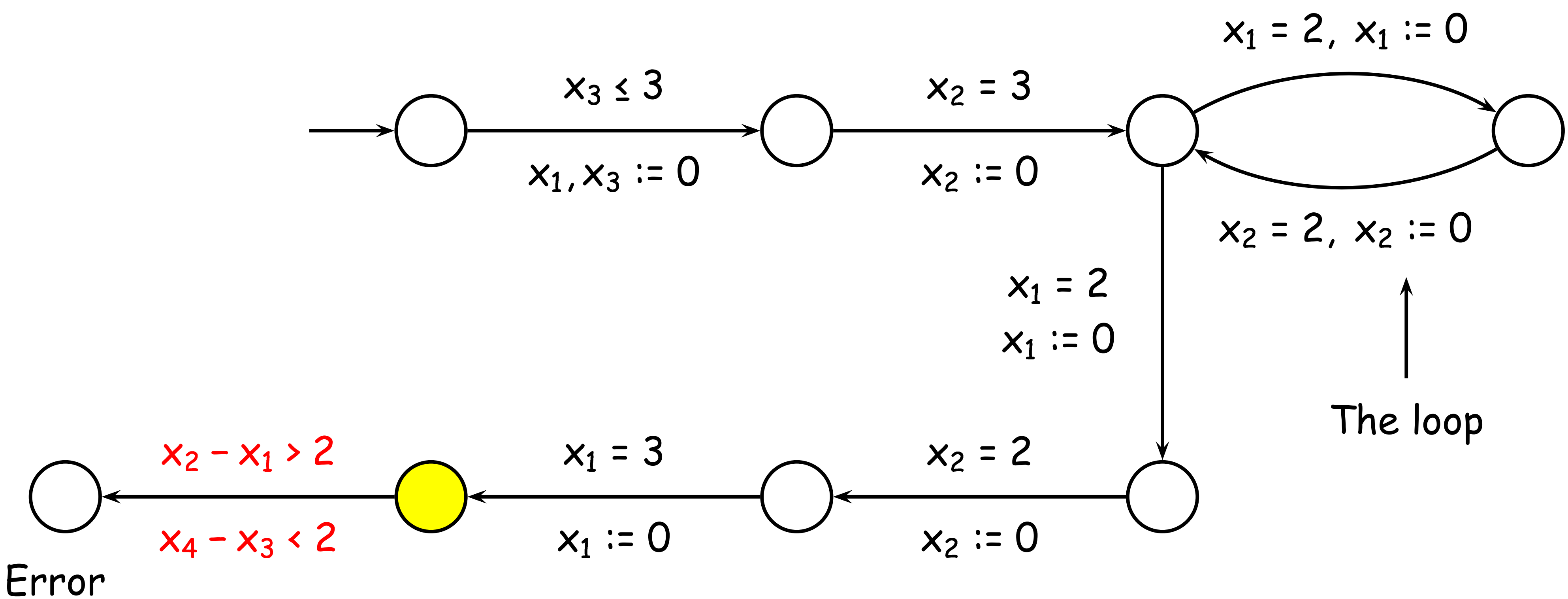


A problematic automaton

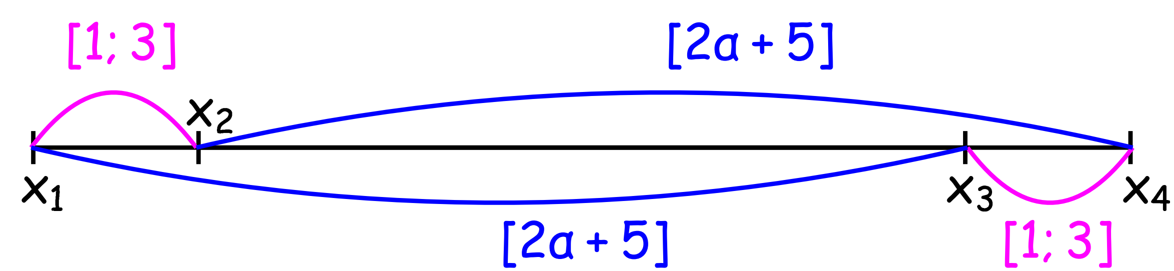


$v(x_1) = 0$
$v(x_2) = d$
$v(x_3) = 2a + 5$
$v(x_4) = 2a + 5 + d$

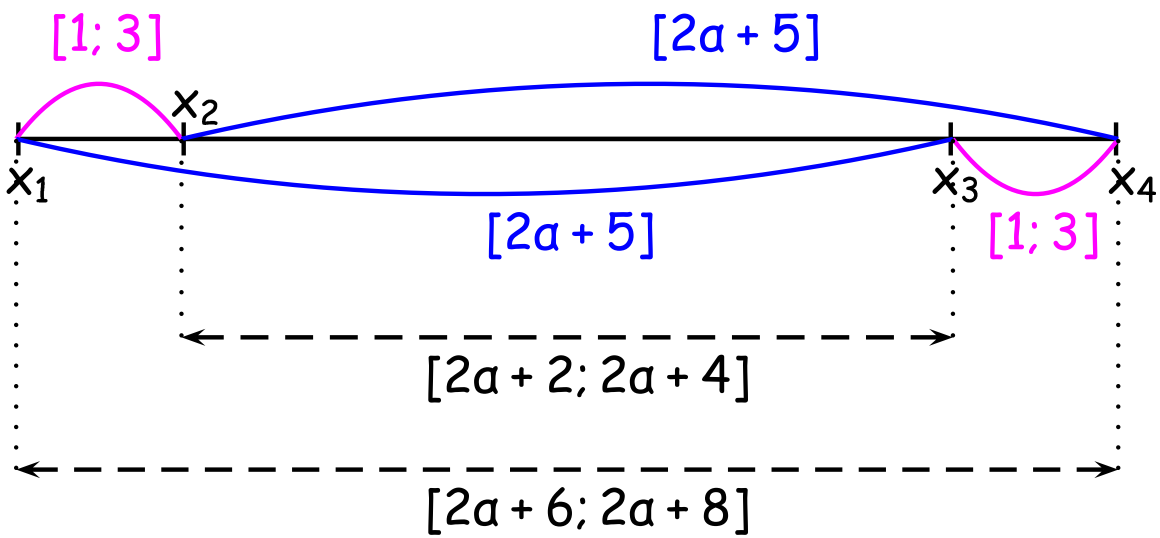
A problematic automaton



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

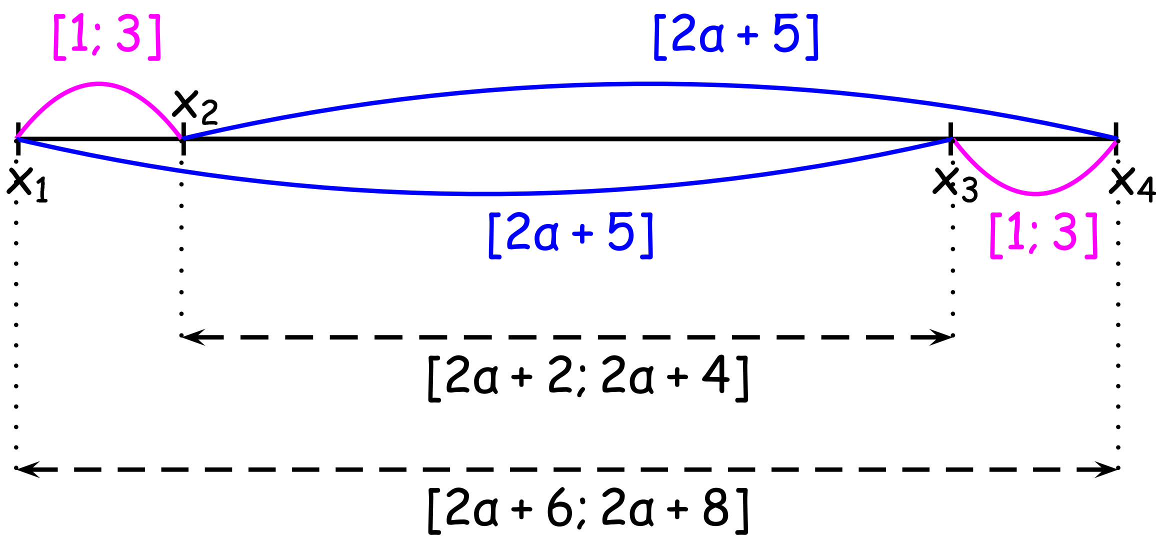


The problematic zone



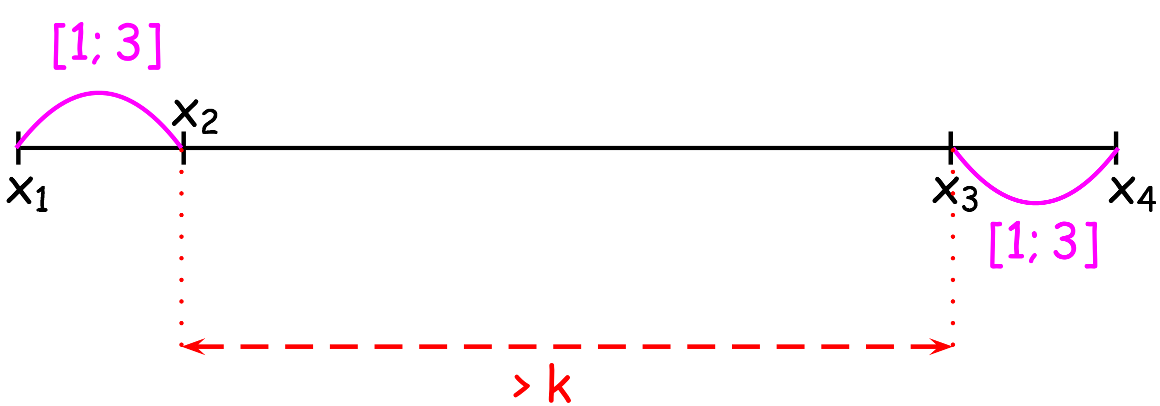
implies $x_1 - x_2 = x_3 - x_4$.

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]
  $Z \subseteq \mathrm{Abs}(Z)$

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]
  $Z \subseteq \mathrm{Abs}(Z)$
- soundness of the abstraction [Soundness]
  the computation of $(\mathrm{Abs} \circ \mathrm{Post})^*$ 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]
  $Z \subseteq \text{Abs}(Z)$
- ✔ soundness of the abstraction [Soundness]
  the computation of $(\text{Abs} \circ \text{Post})^*$ 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,

$$Z \subseteq \mathrm{Extra}_k(Z) \subseteq \mathrm{Abs}(Z)$$

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.
- ✔ A rather big problem with the forward analysis of timed automata needs to be solved.
  - a very unsatisfactory solution for dealing with diagonal constraints.
  - maybe the zones are not the "optimal" objects that we can deal with.

To be continued...

- ✔ Some other current challenges:
  - adding C macros to timed automata
  - reducing the memory consumption *via* new data structures

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).

Bibliography (cont.)

ouyer03] Bouyer. *Untameable Timed Automata!* STACS'03 (LNCS 2607).

ouyer04] Bouyer. *Forward analysis of updatable timed automata.* To appear in FMSD, 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.

insky67] Minsky. *Computation: Finite and Infinite Machines.* 1967.

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

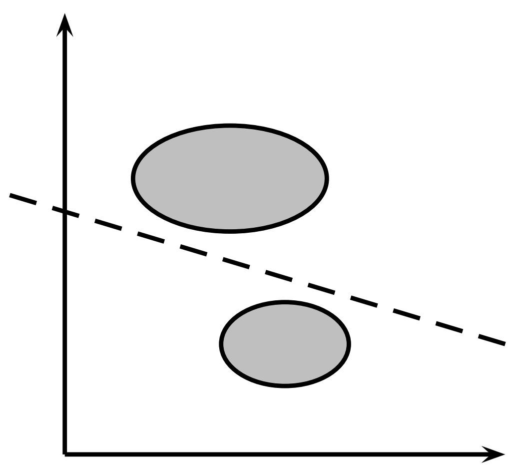
Hytech: `http://www-cad.eecs.berkeley.edu:80/~tah/HyTech/`

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

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

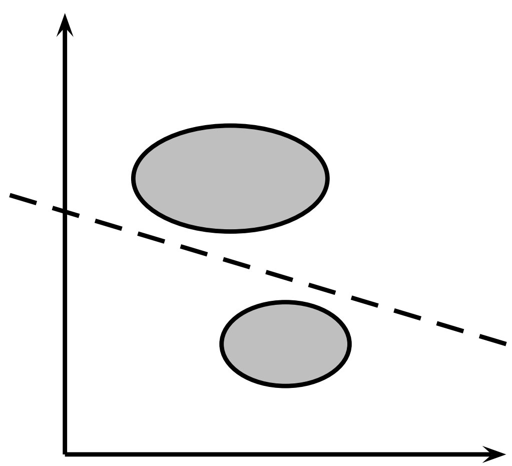
Quizz (1)

1. Let Z_1 and Z_2 be two zones.
   - ✔ $Z_1 \cap Z_2$ is a zone. ❑
   - ✔ $Z_1 \cup Z_2$ is a zone. ❑
   - ✔ The convex hull of $Z_1 \cup Z_2$ is a zone. ❑
2. Let C_1 and C_2 be two disjoint convexes, C_1 is also supposed to be open. Then there exists an hyperplan H that separates C_1 and C_2. ❑



Quizz (1)

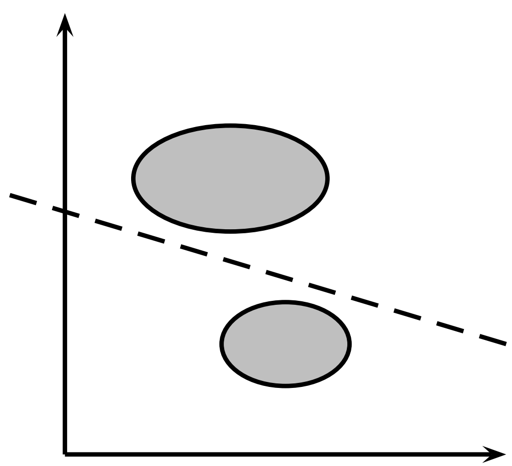
1. Let Z_1 and Z_2 be two zones.
   - ✔ $Z_1 \cap Z_2$ is a zone. ☑
   - ✔ $Z_1 \cup Z_2$ is a zone. ❑
   - ✔ The convex hull of $Z_1 \cup Z_2$ is a zone. ❑
2. Let C_1 and C_2 be two disjoint convexes, C_1 is also supposed to be open. Then there exists an hyperplan H that separates C_1 and C_2. ❑



Quizz (1)

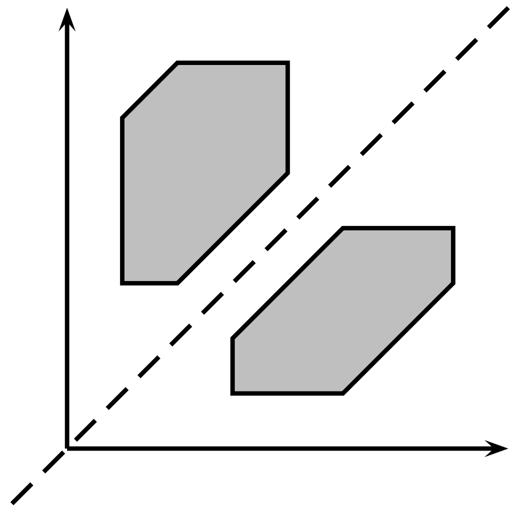
1. Let Z_1 and Z_2 be two zones.
   - ✔ $Z_1 \cap Z_2$ is a zone. ☑
   - ✔ $Z_1 \cup Z_2$ is a zone. ❑
   - ✔ The convex hull of $Z_1 \cup Z_2$ is a zone. ❑
2. Let C_1 and C_2 be two disjoint convexes, C_1 is also supposed to be open. Then there exists an hyperplan H that separates C_1 and C_2. ☑

Hahn-Banach



Quizz (2)

3. Let Z_1 and Z_2 be two disjoint zones. Then there exists an hyperplan H, whose equation is $x - y = c$ or $x = c$ for some clocks x and y and constant c, that separates Z_1 and Z_2. ❑



4. Let Z_1 and Z_2 be two disjoint zones. Then there is a projection π from the set of clocks X on a subset of clocks Y ($2 \leq \#Y < \#X$) such that $\pi(Z_1) \cap \pi(Z_2) = \emptyset$. ❑

Quizz (3)

5. Let Z be a zone and R a region. If $Z \cap R = \emptyset$, then there exists a constraint $x - y \sim c$ defining R (y may be the clock which is always 0) such that $Z \cap (x - y \sim c) = \emptyset$. ❑
6. Let Z be a zone "generated" by a timed automaton. Then for each pair of clocks (x, y), either $Z \cap (x - y < 0) = \emptyset$ or $Z \cap (x - y > 0) = \emptyset$. ❑
7. Let Z_i be zones (such that $\bigcup_i Z_i$ is convex). Then, ❑

$$\mathrm{Approx}_k\Big(\bigcup_i Z_i\Big) = \bigcup_i \big(\mathrm{Approx}_k(Z_i)\big)$$

8. Let $\mathcal{A}$ be a timed automaton. There exists a constant k, syntactically depending on the constraints of $\mathcal{A}$, such that bounding all the clocks by k in the whole automaton does not change the truth or the falsity of the reachability properties. ❑

Quizz (3)

5. Let Z be a zone and R a region. If $Z \cap R = \emptyset$, then there exists a constraint $x-y \sim c$ defining R (y may be the clock which is always 0) such that $Z \cap (x-y \sim c) = \emptyset$. ❑
6. Let Z be a zone "generated" by a timed automaton. Then for each pair of clocks (x, y), either $Z \cap (x-y < 0) = \emptyset$ or $Z \cap (x-y > 0) = \emptyset$. ☑
7. Let Z_i be zones (such that $\bigcup_i Z_i$ is convex). Then, ❑

$$\mathsf{Approx}_k(\bigcup_i Z_i) = \bigcup_i (\mathsf{Approx}_k(Z_i))$$

8. Let $\mathcal{A}$ be a timed automaton. There exists a constant k, syntactically depending on the constraints of $\mathcal{A}$, such that bounding all the clocks by k in the whole automaton does not change the truth or the falsity of the reachability properties. ❑

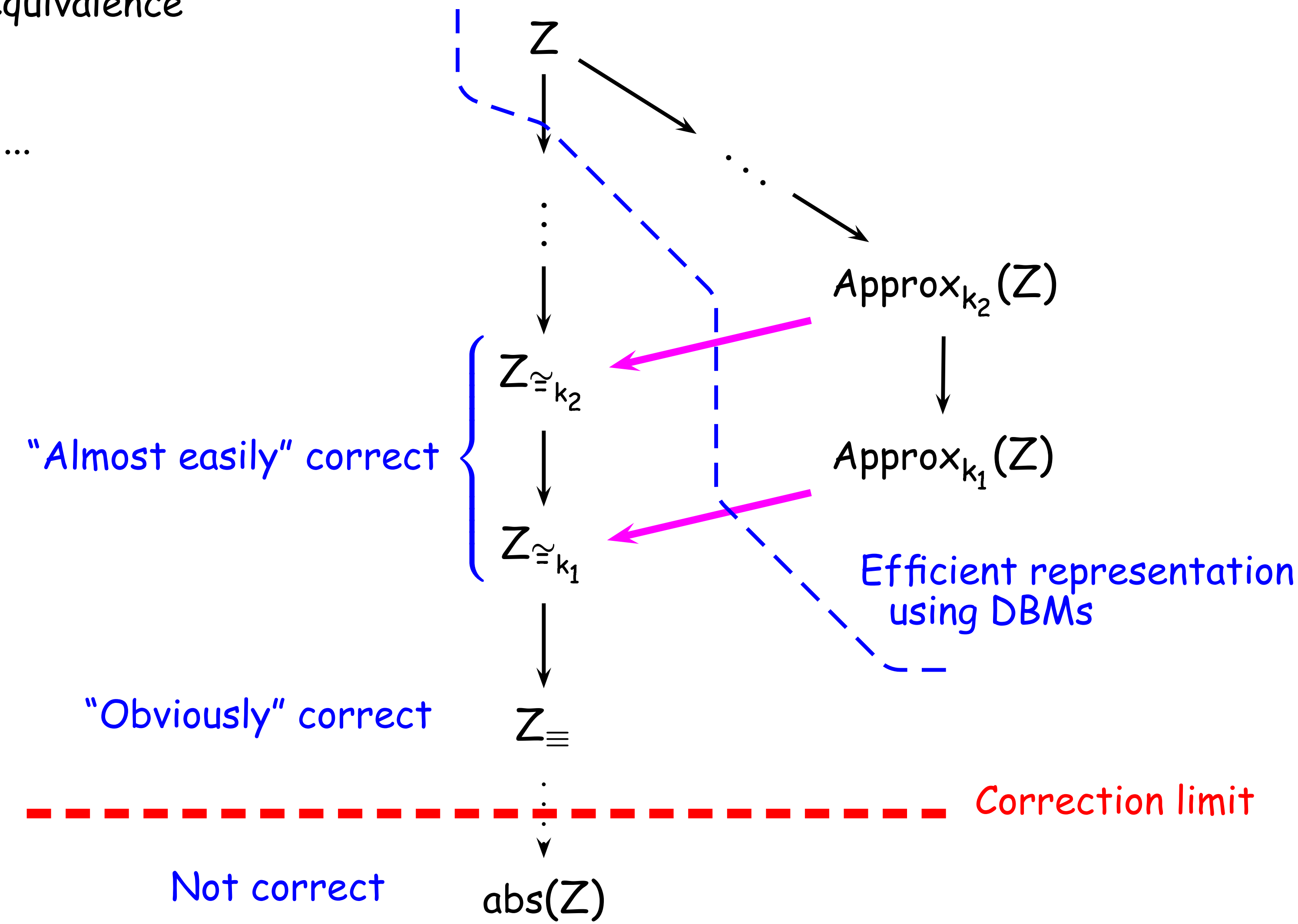
Plan of one of the proofs (2nd proof)

$\equiv$ bisimulation

$\cong_k$ k-equivalence

$k_1 > k_2 > \ldots$

[Behrmann,Bouyer,Fleury,Larsen 2003]



$\mathrm{Approx}_k(Z) \subseteq Z_{\widetilde{=}_k}$

- let $\sigma \in \mathrm{Approx}_k(Z)$
- prove $\{\sigma' \in Z \mid \sigma' \widetilde{=}_k \sigma\}$ is not empty
- this set is defined by:
  - the constraints defining Z,
  - $x = \sigma(x)$ whenever $\sigma(x) \leq k$, stronger than the constraint in Z
  - $x > k$ whenever $\sigma(x) > k$

  $\longrightarrow$ this defines a DBM on the set of real numbers
- use the property that a DBM $(m_{i,j})_{i,j=1\dots n}$ represents the empty set iff there exists a sequence of distinct indices $(i_j)_{j=1\dots p}$ such that

$$m_{i_1,i_2} + \dots + m_{i_{p-1},i_p} + m_{i_p,i_1} < 0$$

- check what can be these negative cycles...

k-equivalence

$$\sigma \cong_k \sigma' \iff \forall x \left| \begin{array}{l} \text{either } \sigma(x) = \sigma'(x) \\ \text{or } \sigma(x) > k \text{ and } \sigma'(x) > k \end{array} \right.$$