Timed Automata – From Theory to Implementation

Patricia Bouyer

LSV – CNRS & ENS de Cachan
France

Roadmap

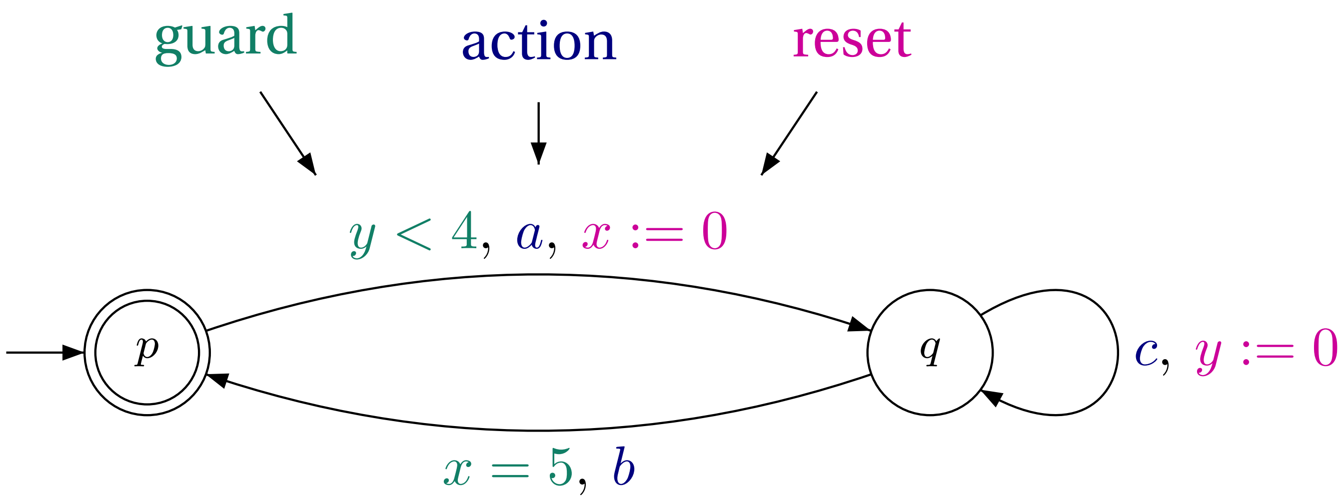
- Timed automata, decidability issues
- Some extensions of the model
- Implementation of timed automata

Timed automata, decidability issues

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

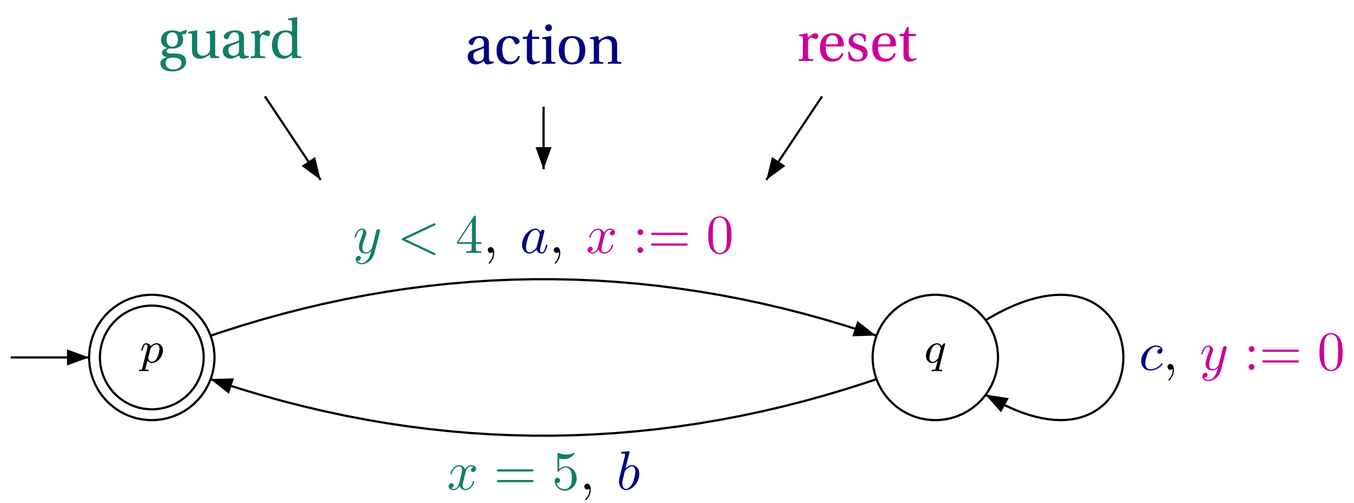
Timed automata

x, y: clocks [Alur & Dill - 1990's]



Timed automata

x, y: clocks [Alur & Dill - 1990's]



	p	$\xrightarrow[3.2]{a}$	q	$\xrightarrow[5.1]{c}$	q	$\xrightarrow[8.2]{b}$	p	$\ldots$
value of x	0		0		1.9		5	$\ldots$
value of y	0		3.2		0		3.1	$\ldots$

➜ timed word $(a, 3.2)(c, 5.1)(b, 8.2)\ldots$

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

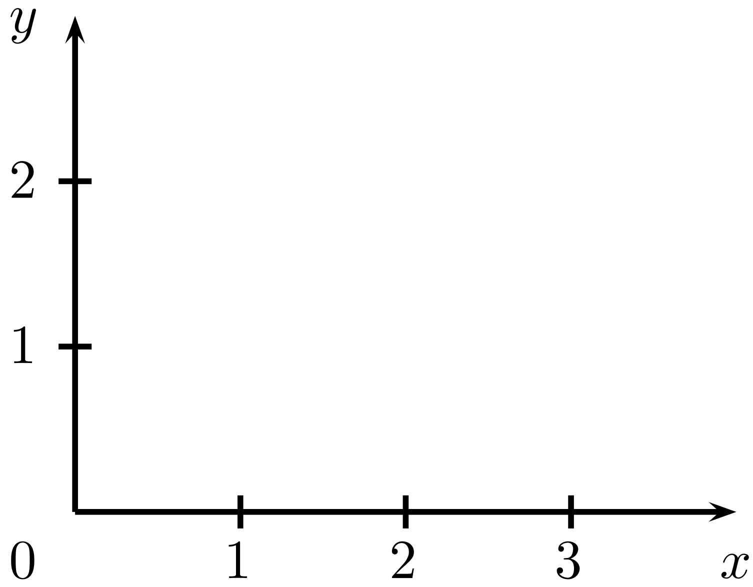
Emptiness problem: is the language accepted by a timed automaton empty?

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

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

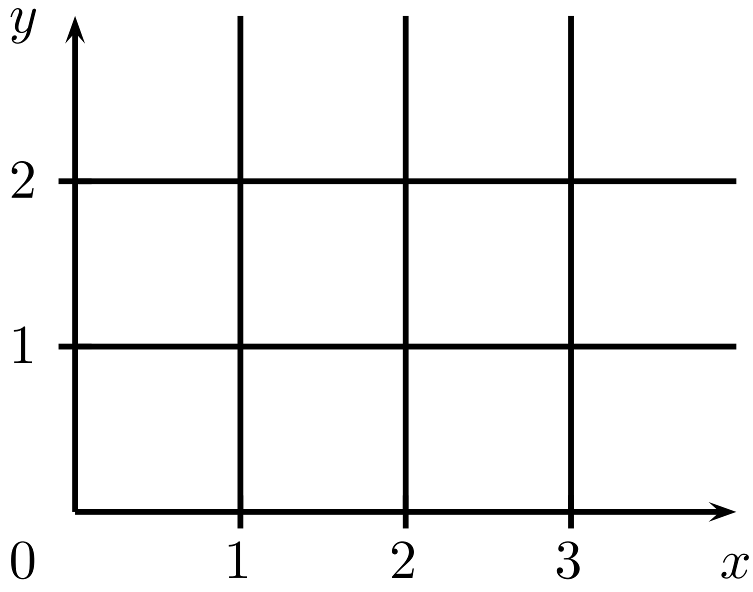
[Alur & Dill 1990's]

The region abstraction



Equivalence of finite index

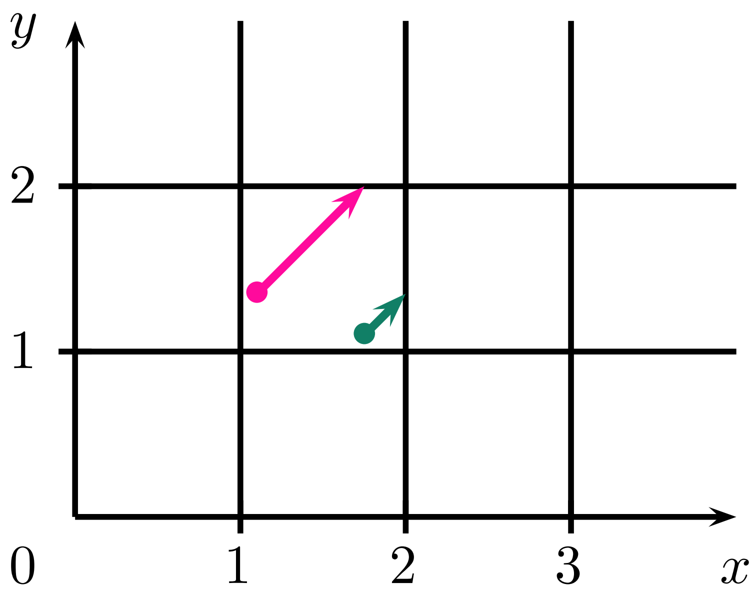
The region abstraction



Equivalence of finite index

- "compatibility" between regions and constraints

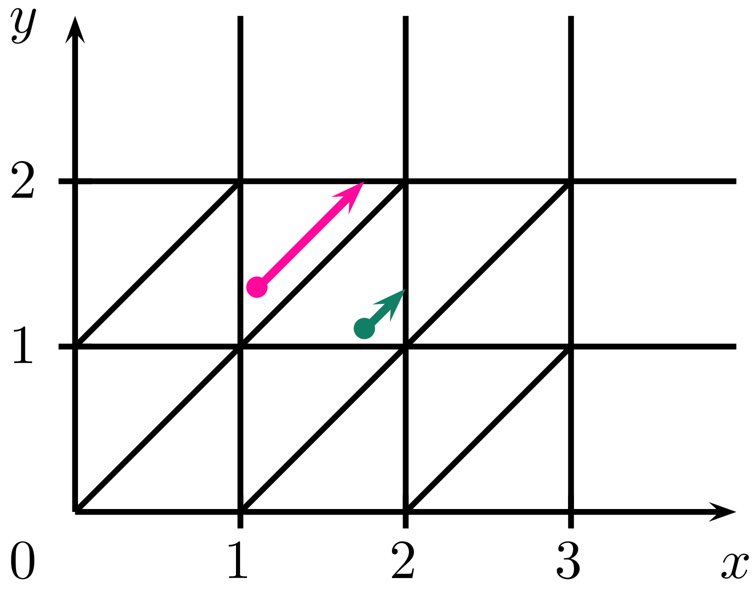
The region abstraction



Equivalence of finite index

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

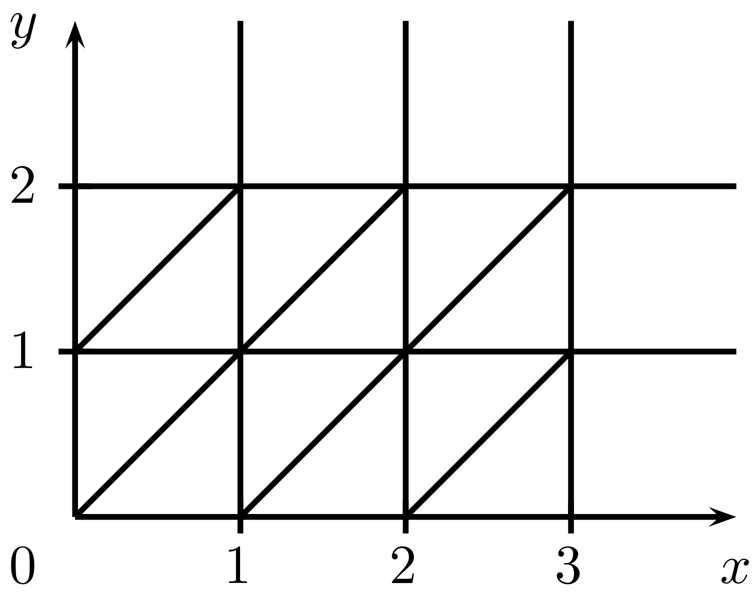
The region abstraction



Equivalence of finite index

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

The region abstraction

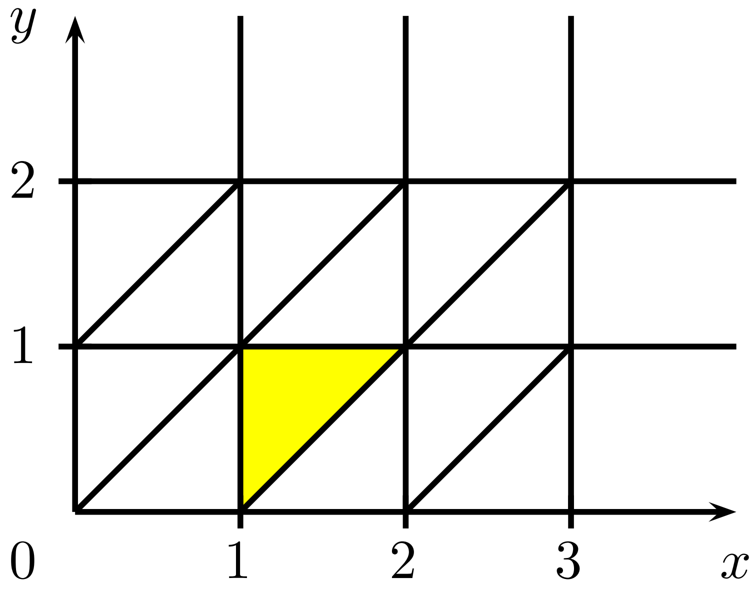


Equivalence of finite index

- "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 automaton

timed automaton $\bigotimes$ region partition

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

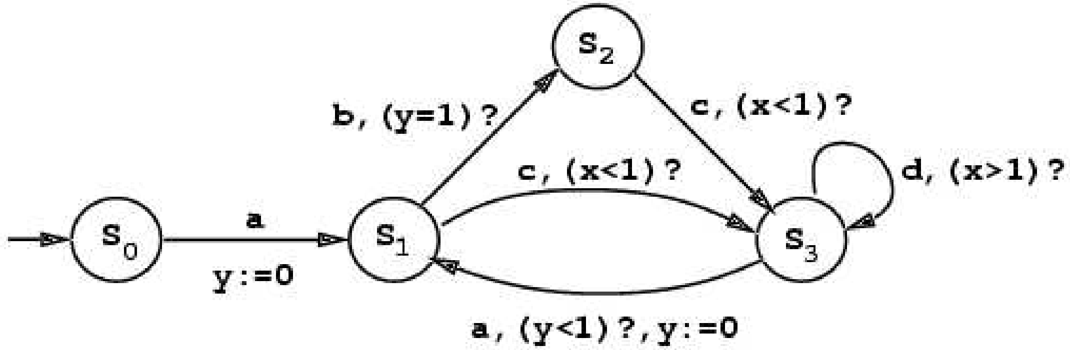
$(q, R) \xrightarrow{a} (q', R')$ if there exists $R'' \in \text{Succ}_t^*(R)$ s.t.

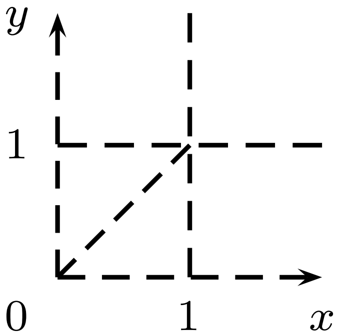
- $R'' \subseteq g$
- $[C \leftarrow 0]R'' \subseteq R'$

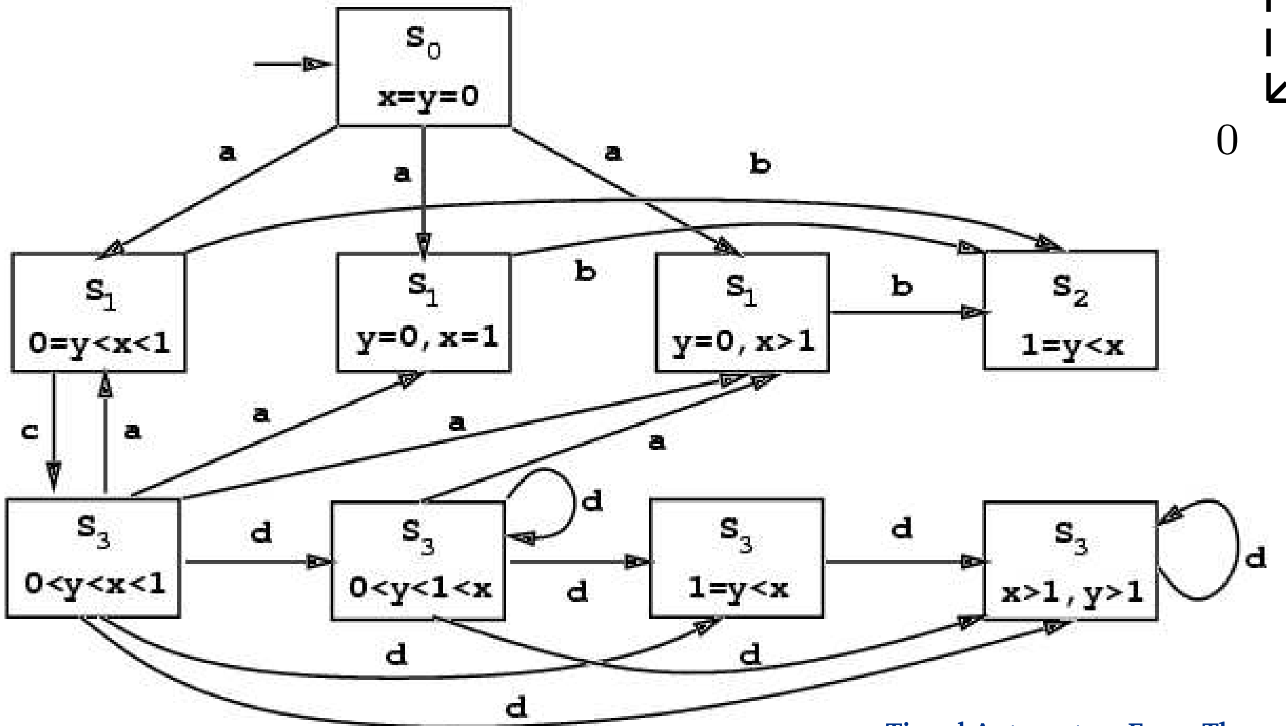
$\mathcal{L}$(reg. aut.) = UNTIME($\mathcal{L}$(timed aut.))

where 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

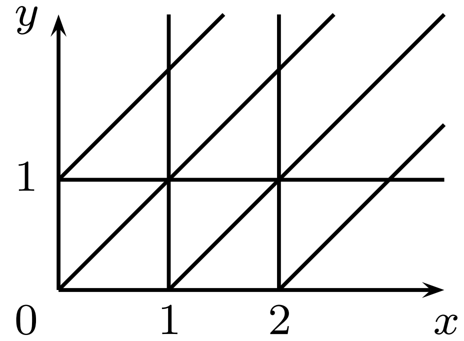
Some extensions of the model

- 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

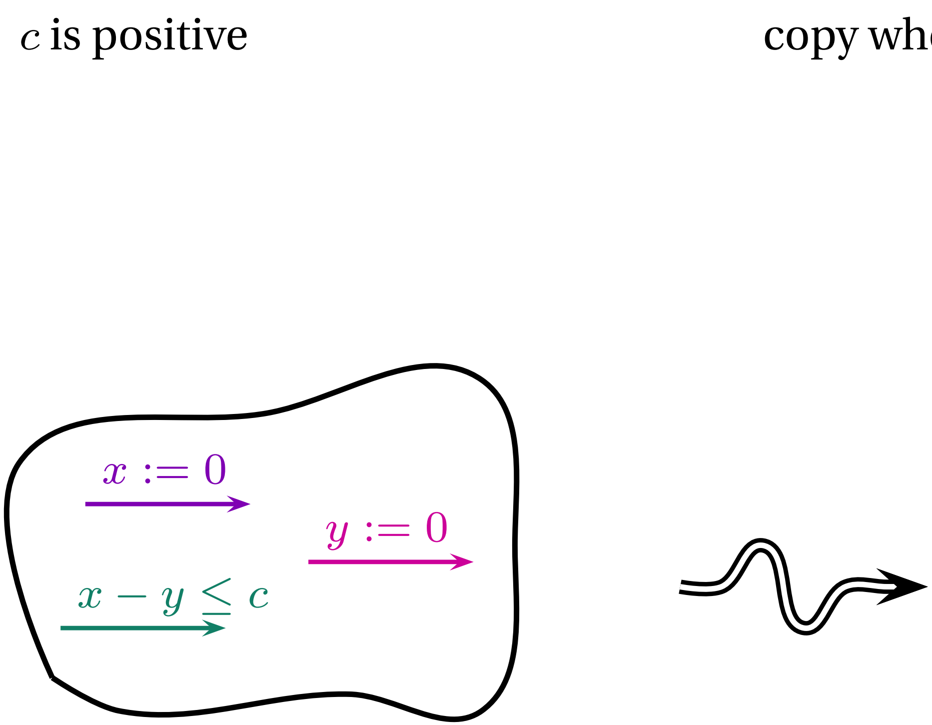
$$\boxed{x - y \sim c \quad \text{and} \quad x \sim c}$$

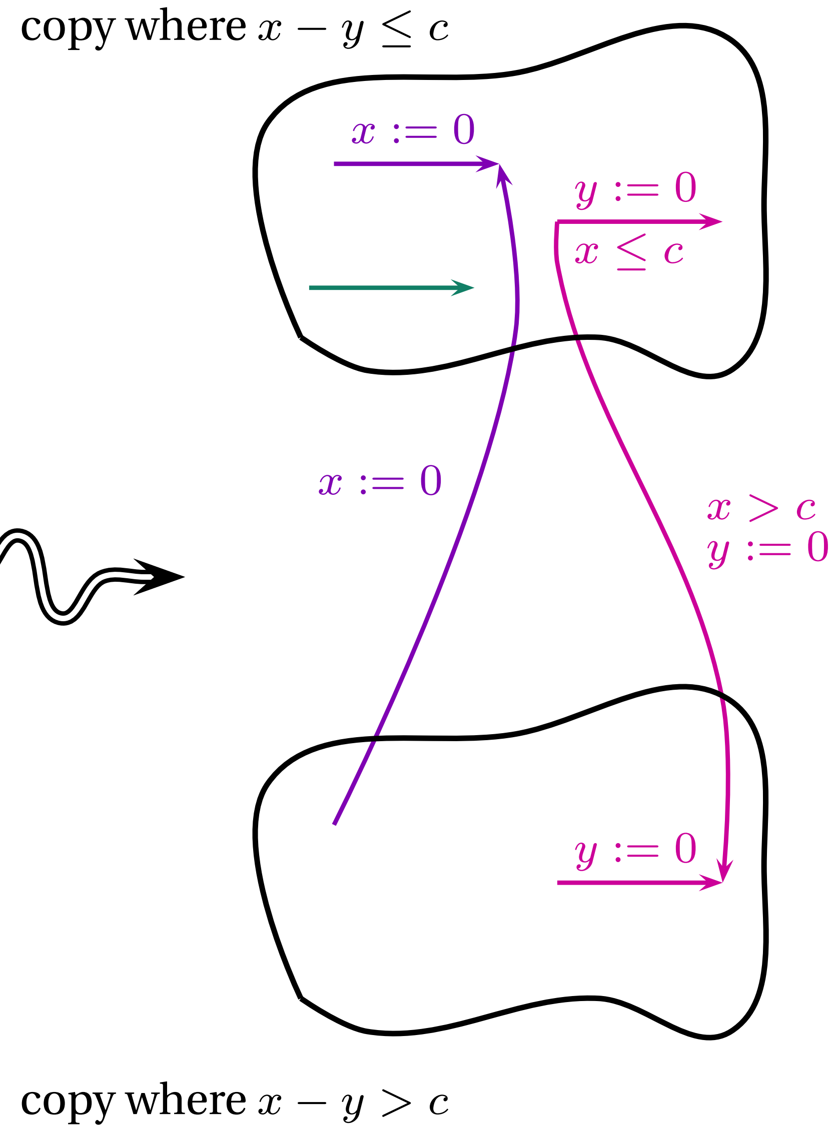
- **Decidability:** yes, using the region abstraction



- **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

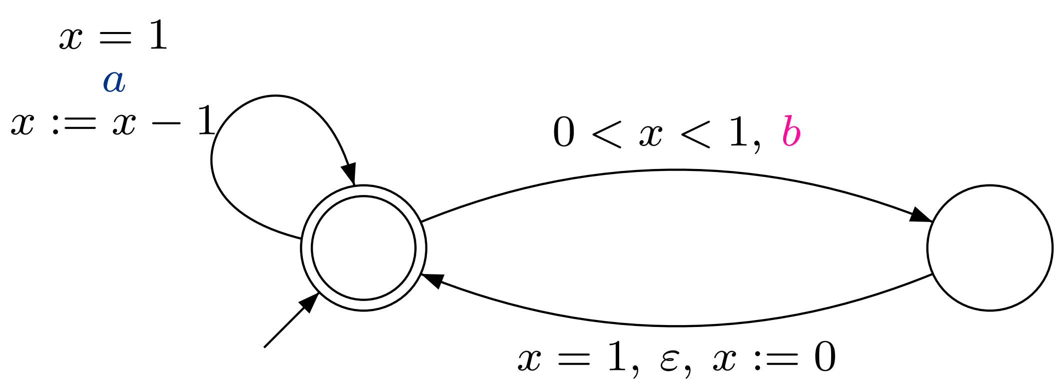
are explonentially more concise than diagonal-free timed automata.

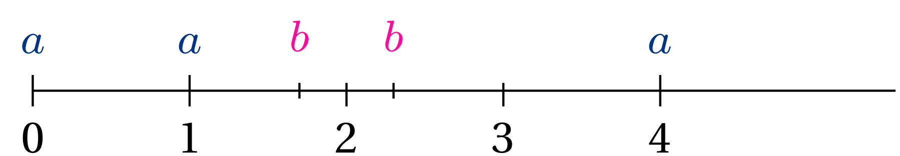
Adding silent actions

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

[Bérard,Diekert,Gastin,Petit 1998]

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

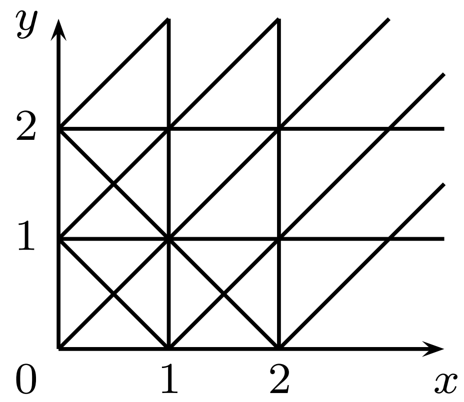




Adding constraints of the form $x + y \sim c$

$\boxed{x + y \sim c \quad \text{and} \quad x \sim c}$ [Bérard,Dufourd 2000]

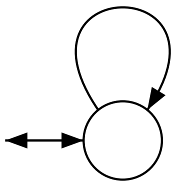
- **Decidability:** - for two clocks, decidable using the abstraction



  - for four clocks (or more), undecidable!

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

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



The two-counter machine

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

- Incrementation:

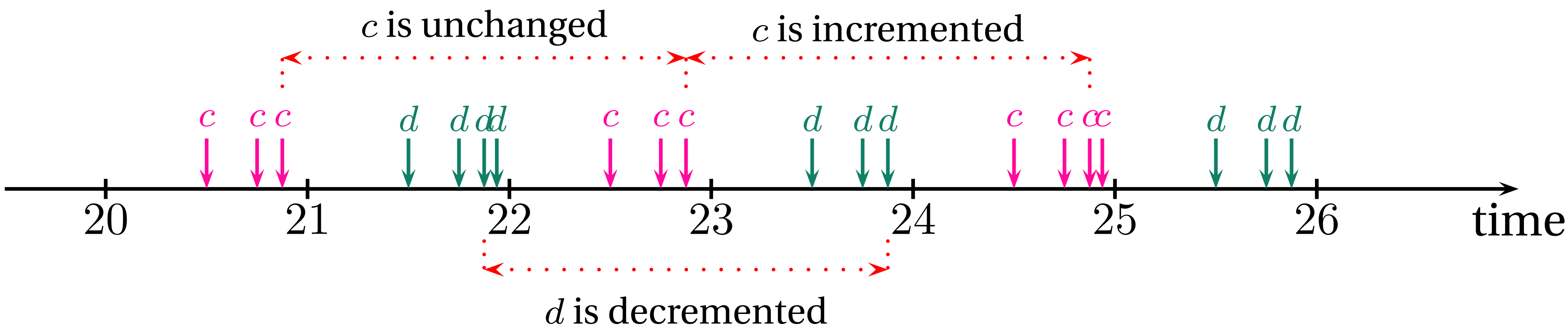
  `(p):` $x := x + 1$; `goto (q)`

- 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

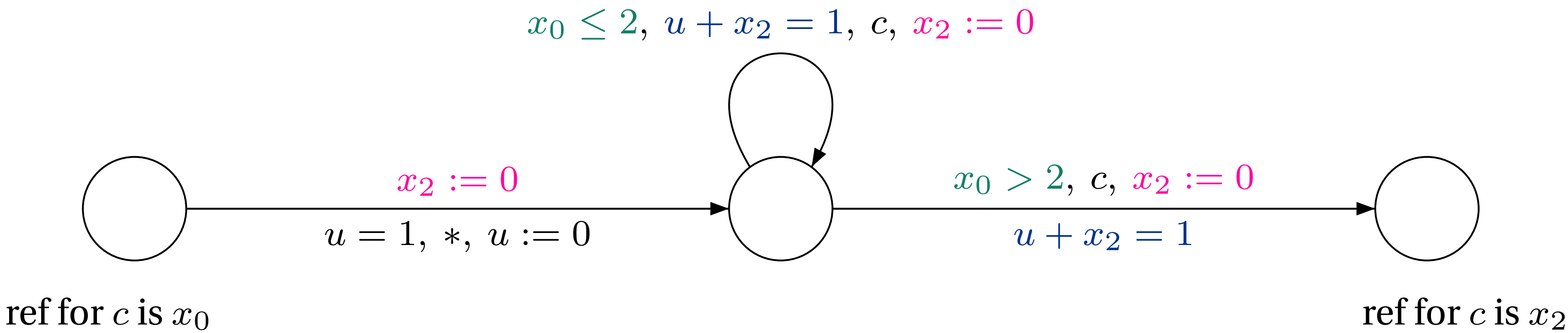
We will use 4 clocks:
- u, “tic” clock (each time unit)
- x_0, x_1, x_2: reference clocks for the two counters

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

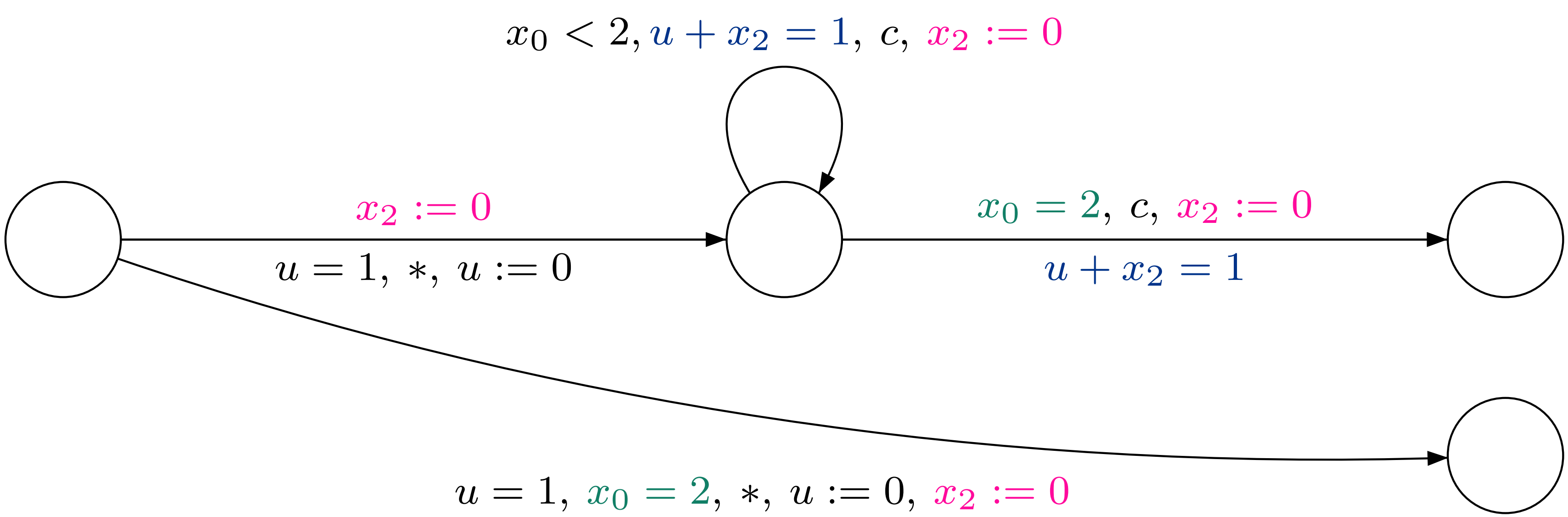
[Bérard,Dufourd 2000]

Undecidability proof (cont.)

- **Increment of counter c:**

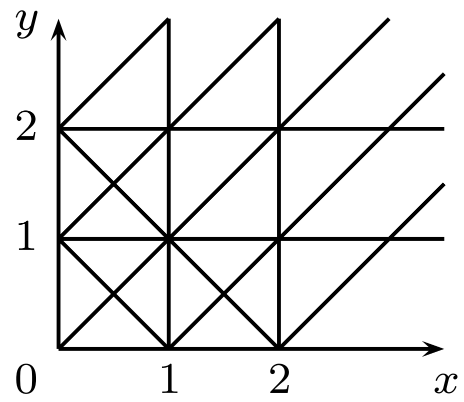


- **Decrement of counter c:**



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

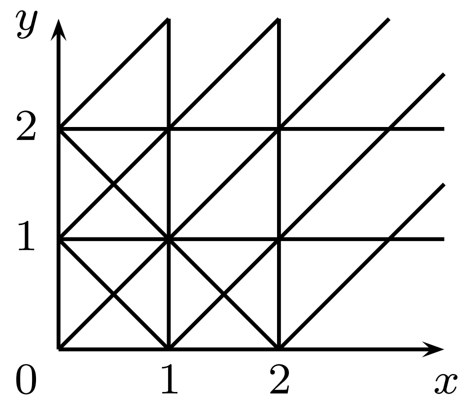
- Two clocks: decidable! using the abstraction



- 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

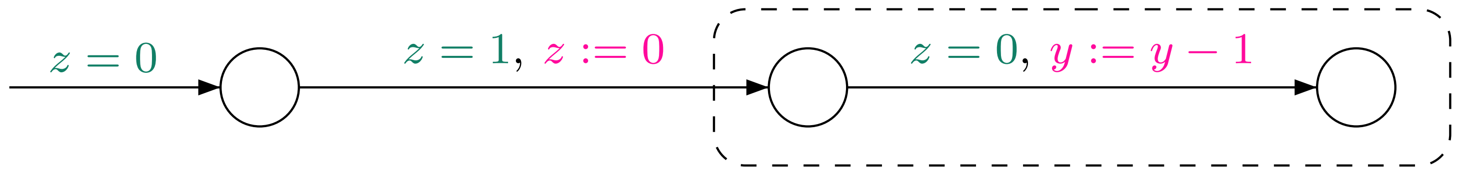
Several types of updates: $x := y + c$, $x :< c$, $x :> c$, etc...

- The general model is undecidable.

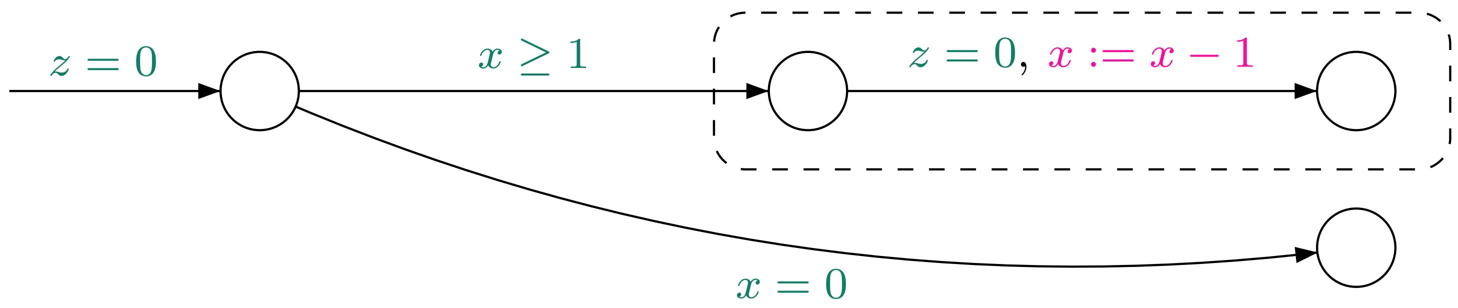
  (simulation of a two-counter machine)

- Only decrementation also leads to undecidability

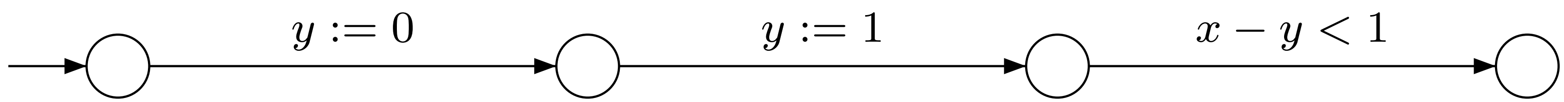
  – **Incrementation of counter x**



  – **Decrementation of counter x**



Decidability



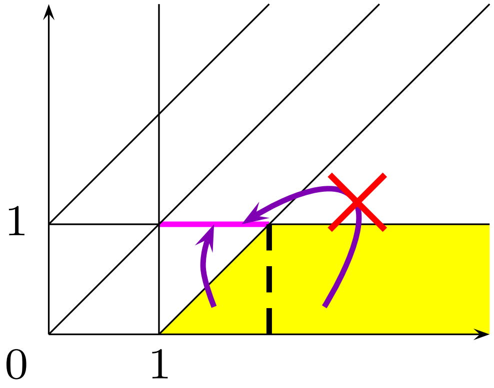


image by $y := 1$

➜ the bisimulation property is not met

The classical region automaton construction is not correct.

Decidability (cont.)

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

$\rightsquigarrow$ is there a solution?

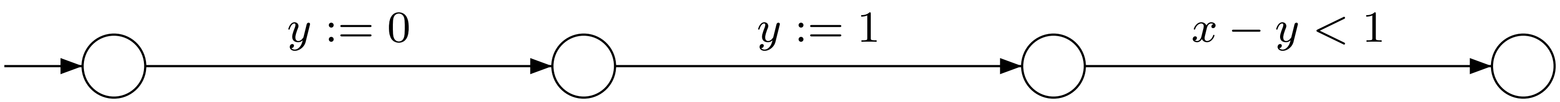
$\rightsquigarrow$ if yes, belongs to a decidable class

Examples:

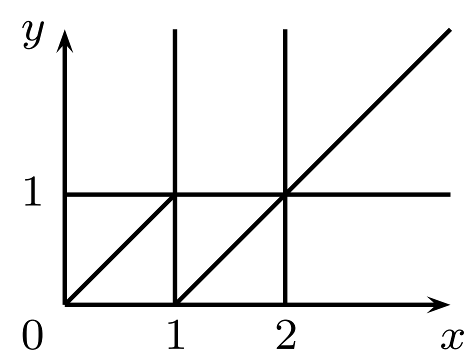
- constraint $x \sim c$ $\qquad c \leq \max_x$
- constraint $x - y \sim c$ $\qquad c \leq \max_{x,y}$
- update $x :\sim y + c$ $\qquad \max_x \leq \max_y + c$
  and for each clock z, $\max_{x,z} \geq \max_{y,z} + c$, $\max_{z,x} \geq \max_{z,y} - c$
- update $x :< c$ $\qquad c \leq \max_x$
  and for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



$$\left\{\begin{array}{l} \max_y \geq 0 \\ \max_x \geq 0 + \max_{x,y} \\ \max_y \geq 1 \\ \max_x \geq 1 + \max_{x,y} \\ \max_{x,y} \geq 1 \end{array}\right. \quad \Longrightarrow \quad \left\{\begin{array}{l} \max_x = 2 \\ \max_y = 1 \\ \max_{x,y} = 1 \\ \max_{y,x} = -1 \end{array}\right.$$

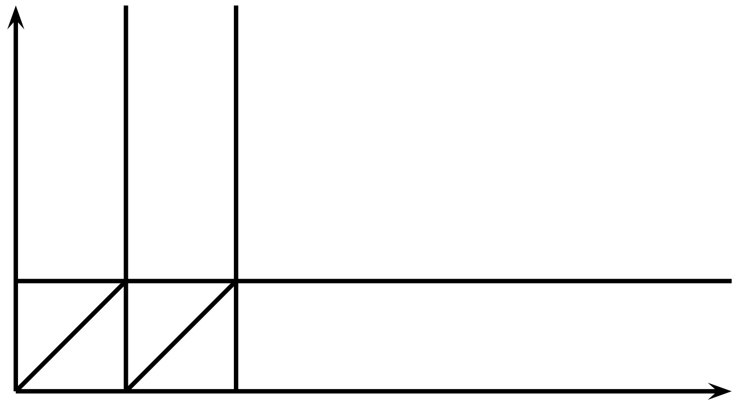


The bisimulation property is met.

What's wrong when undecidable?

Decrementation $x := x - 1$

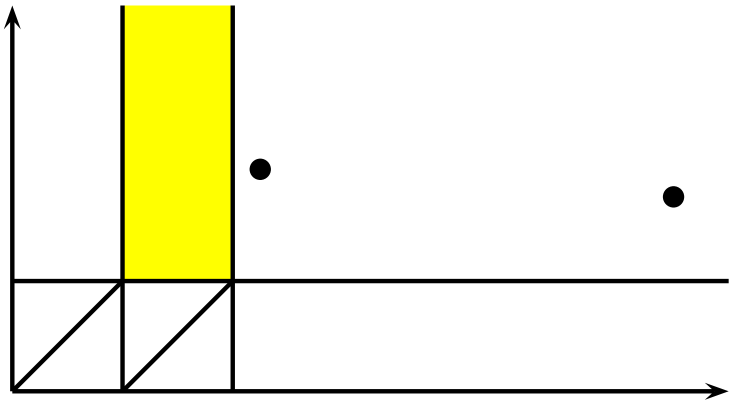
$$\max_x \leq \max_x - 1$$



What's wrong when undecidable?

Decrementation $x := x - 1$

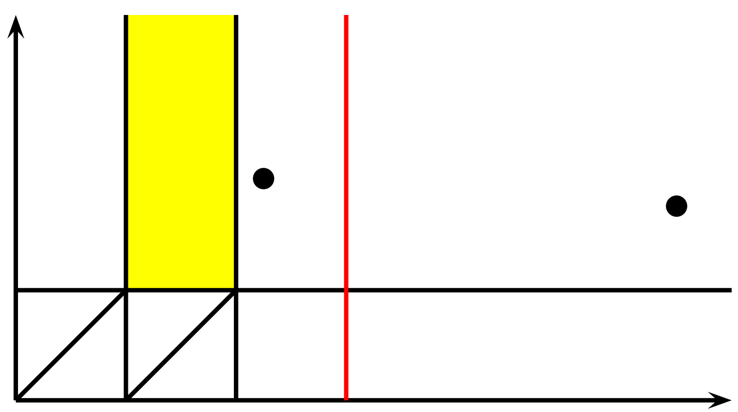
$$\max_x \leq \max_x - 1$$



What's wrong when undecidable?

Decrementation $x := x - 1$

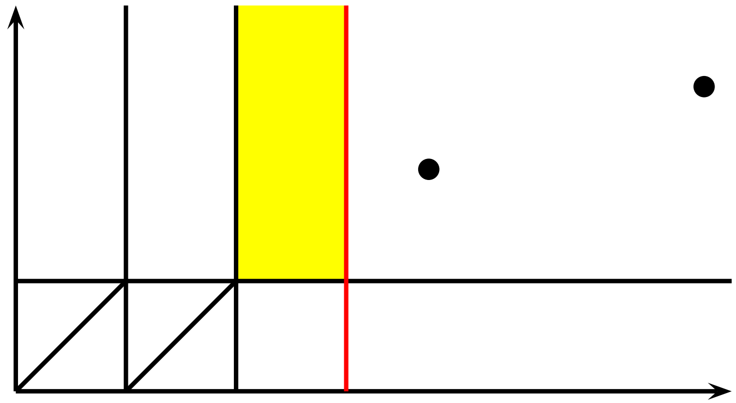
$$\max_x \leq \max_x - 1$$



What's wrong when undecidable?

Decrementation $x := x - 1$

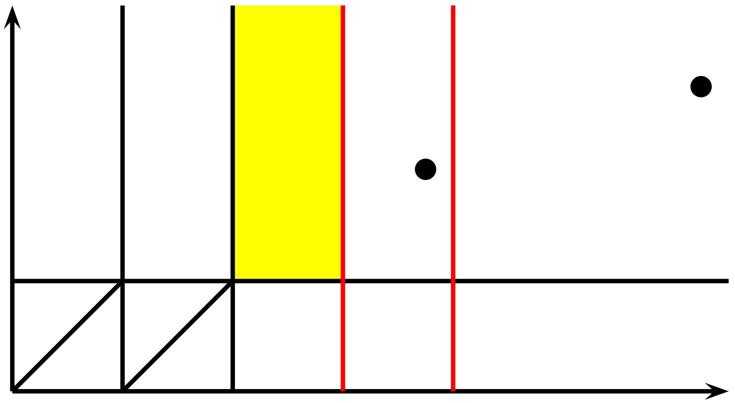
$$\max_x \leq \max_x - 1$$



What's wrong when undecidable?

Decrementation $x := x - 1$

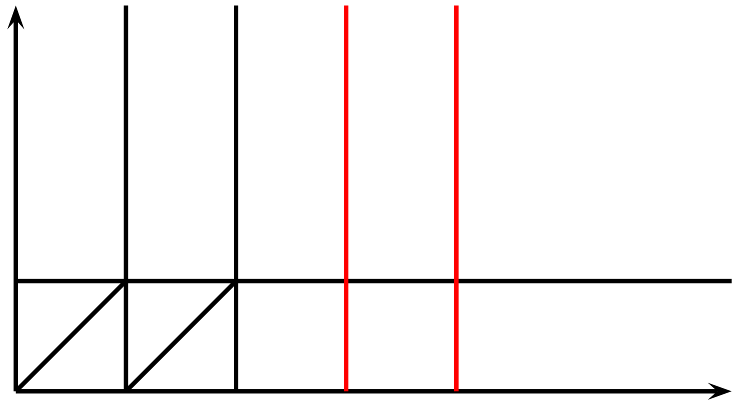
$$\max_x \leq \max_x - 1$$



What's wrong when undecidable?

Decrementation $x := x - 1$

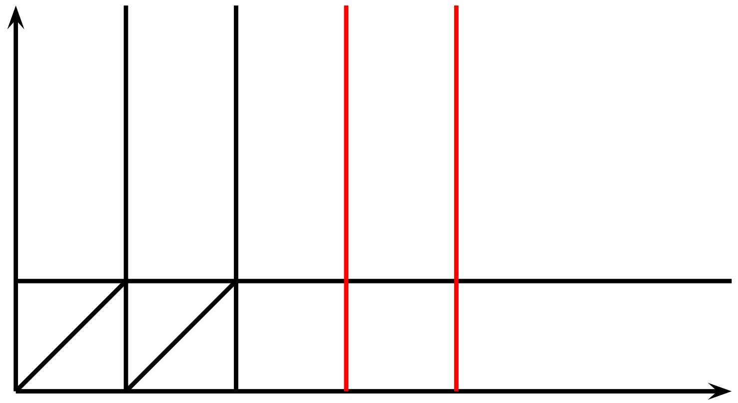
$$\max_x \leq \max_x - 1$$



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]

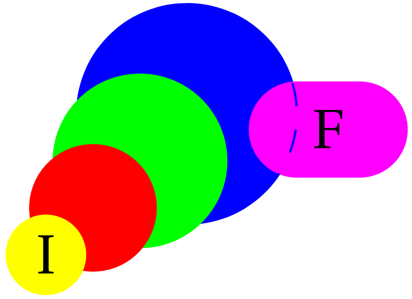
...but **on-the-fly technics** are preferred.

Reachability analysis

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

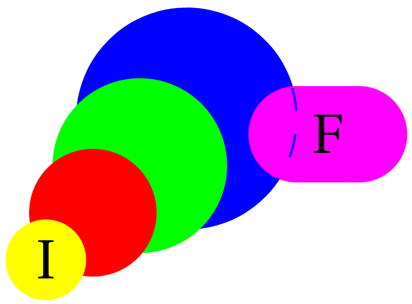
Reachability analysis

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

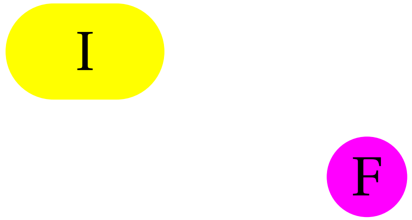


Reachability analysis

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

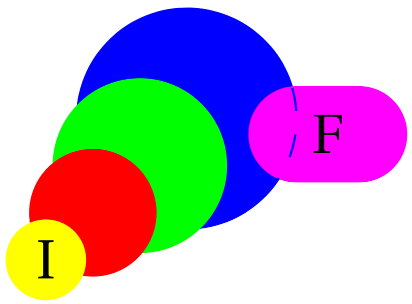


- **backward analysis algorithm:**
  compute the predecessors of final configurations

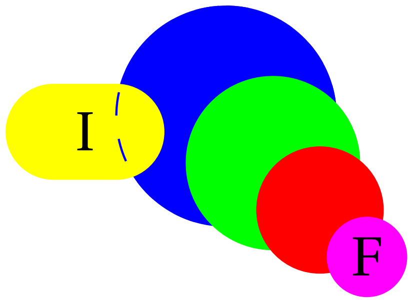


Reachability analysis

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



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



Note on the backward analysis of TA

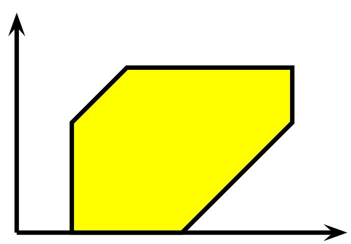
$$\ell \xrightarrow{\quad g,\ a,\ C := 0 \quad} \ell'$$

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

Note on the backward analysis of TA

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

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

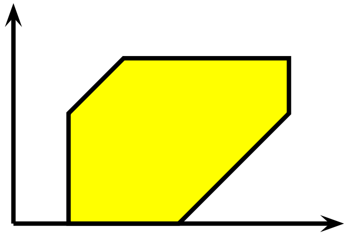


Z

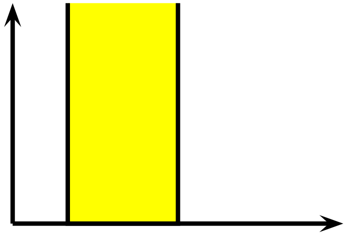
Note on the backward analysis of TA

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

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



Z

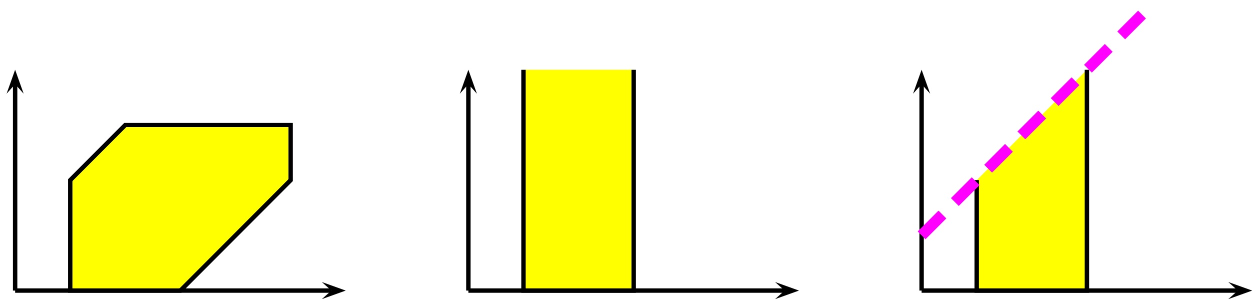


$[C \leftarrow 0]^{-1}(Z \cap (C = 0))$

Note on the backward analysis of TA

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

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

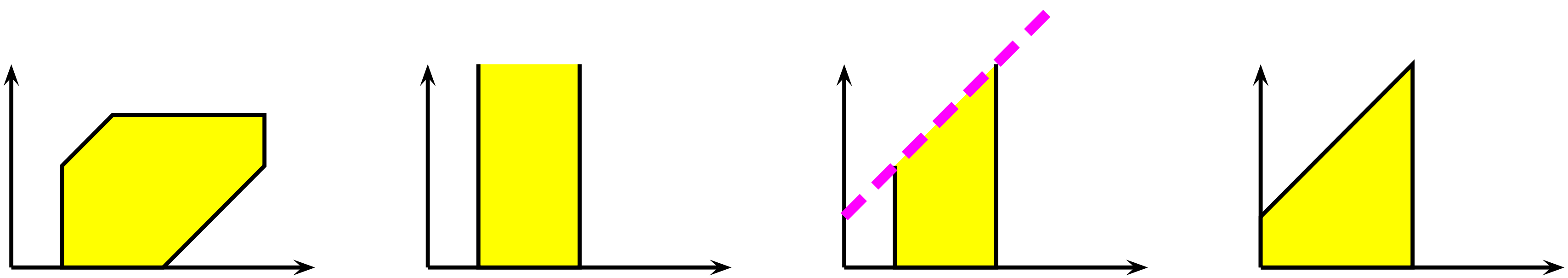


Z $\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$

Note on the backward analysis of TA

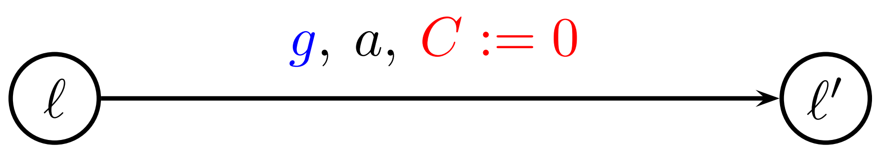
$$\ell \xrightarrow{g,\ a,\ C := 0} \ell'$$

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

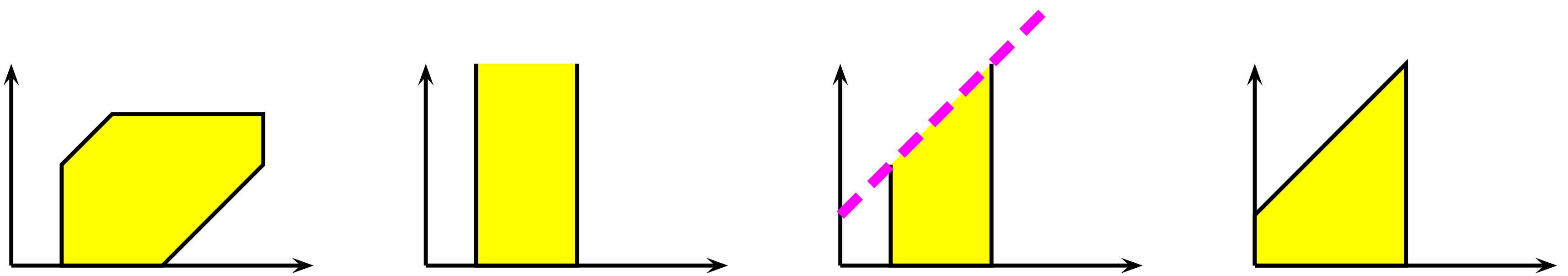


Z $\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$ $\qquad$ $\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$

Note on the backward analysis of TA



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

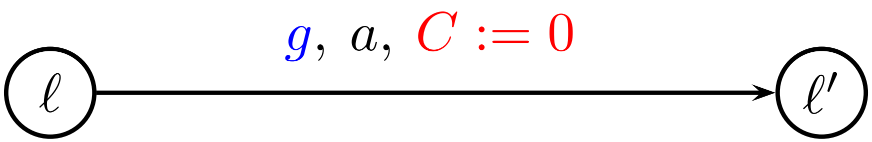
$$\ell \xrightarrow{\quad g,\ a,\ C := 0 \quad} \ell'$$

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

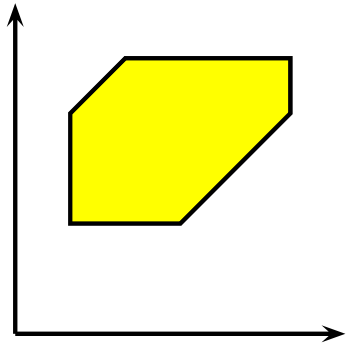
A zone is a set of valuations defined by a clock constraint

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

Forward analysis of TA



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

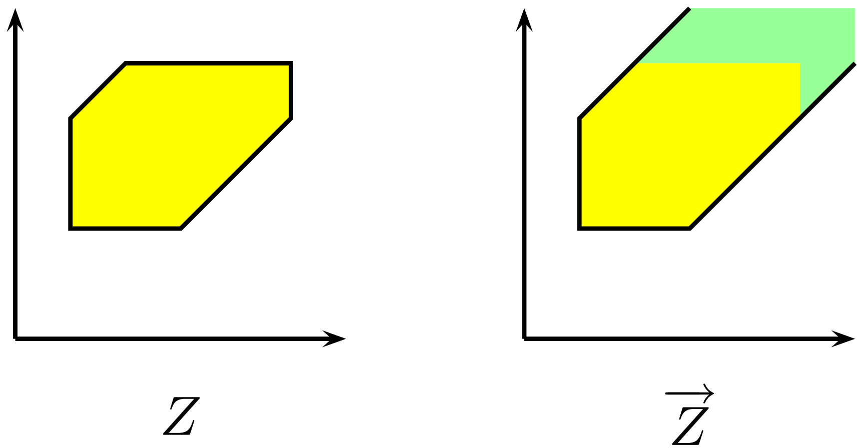


Z

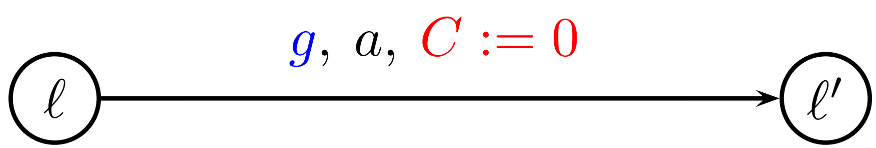
Forward analysis of TA

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

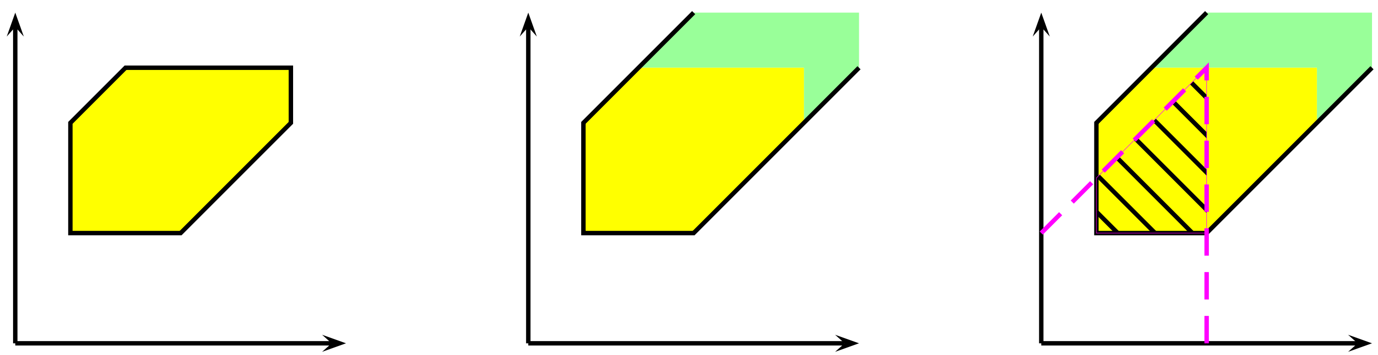
zones $\quad Z \quad\quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$



Forward analysis of TA

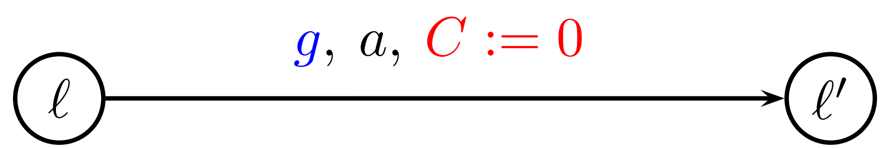


zones Z $[C \leftarrow 0](\overrightarrow{Z} \cap g)$

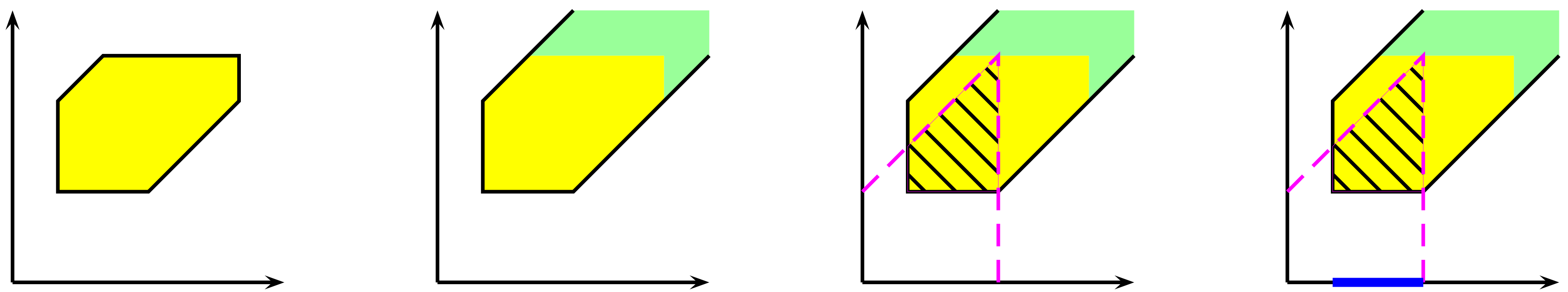


Z $\overrightarrow{Z}$ $\overrightarrow{Z} \cap g$

Forward analysis of TA



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

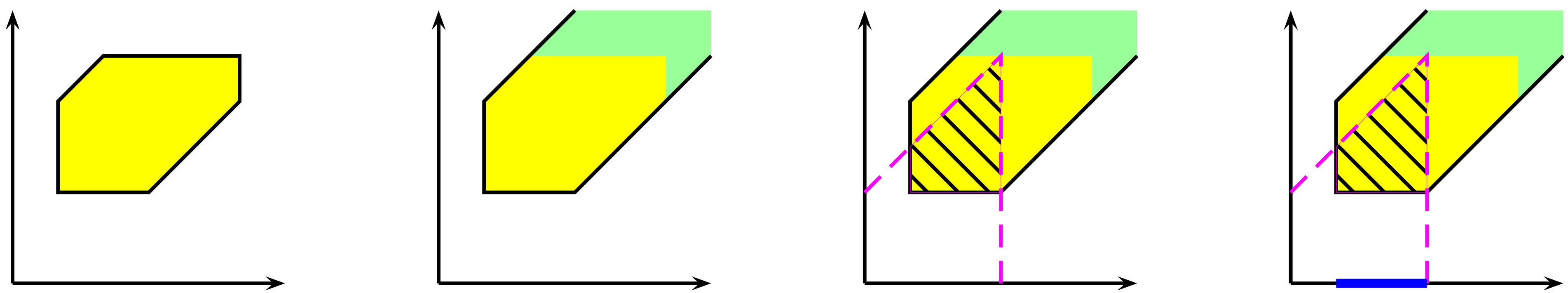


$Z \qquad \overrightarrow{Z} \qquad \overrightarrow{Z} \cap g \qquad [y \leftarrow 0](\overrightarrow{Z} \cap g)$

Forward analysis of TA

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

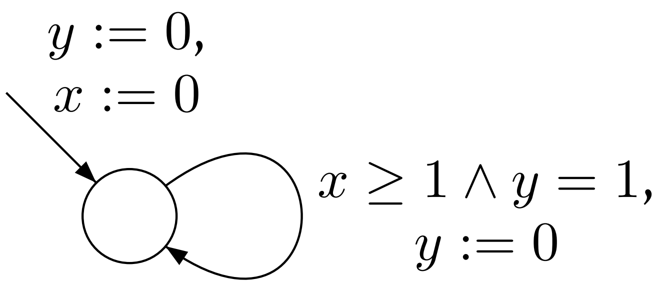
zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

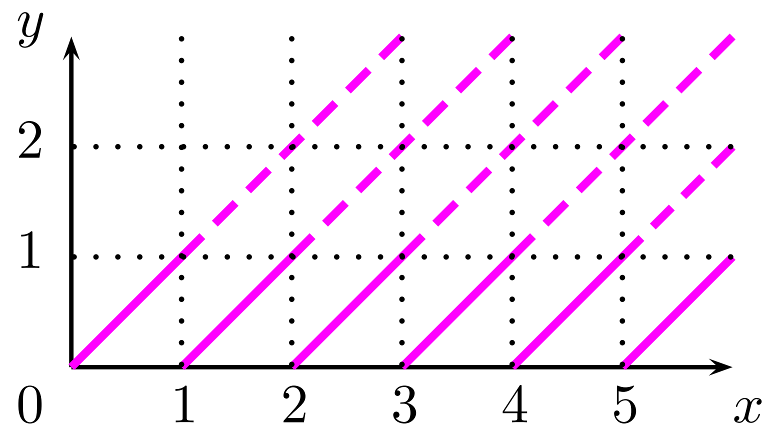


$Z \qquad \overrightarrow{Z} \qquad \overrightarrow{Z} \cap g \qquad [y \leftarrow 0](\overrightarrow{Z} \cap g)$

➜ 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

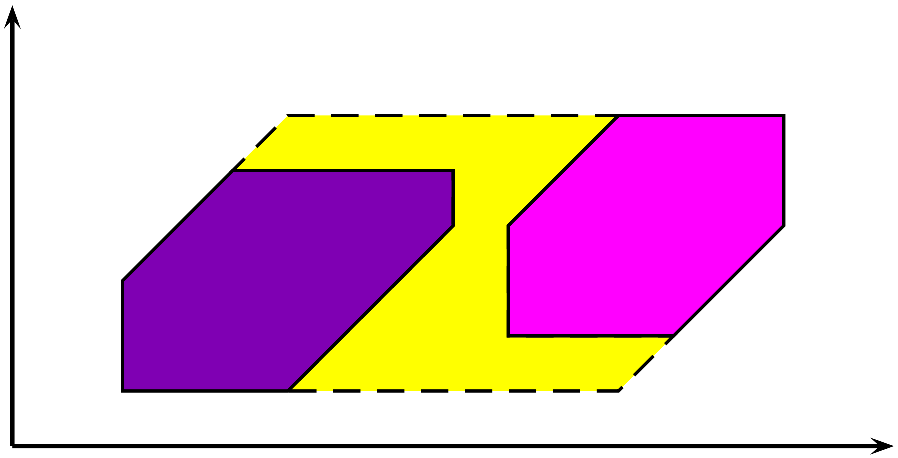
$$q \xrightarrow{g,a,C:=0} q' \quad \Longrightarrow \quad \mathrm{Act}(q) = \mathrm{clocks}(g) \cup (\mathrm{Act}(q') \setminus C)$$

...

“Solutions” to this problem (cont.)

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

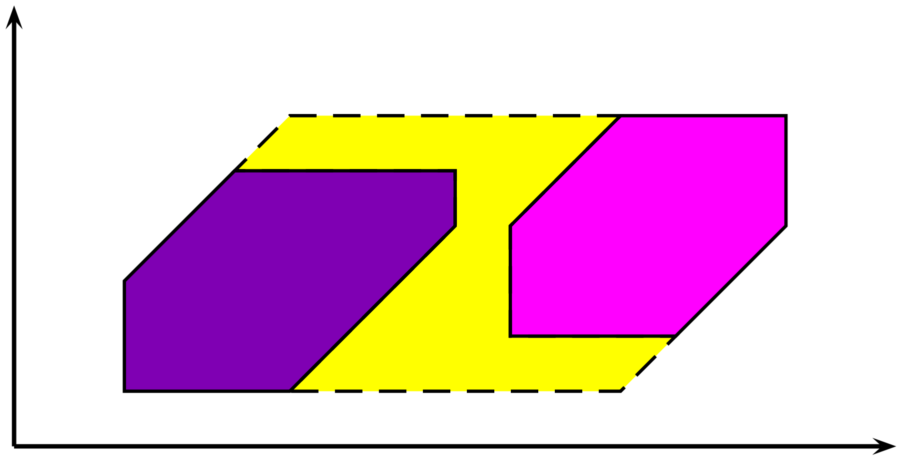
  ➜ “semi-correct” w.r.t. reachability



“Solutions” to this problem (cont.)

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

  ➜ “semi-correct” w.r.t. reachability



- **extrapolation**, a widening operator on zones

The DBM data structure

DBM (Difference Bounded Matrice) data structure [Dill89]

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

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

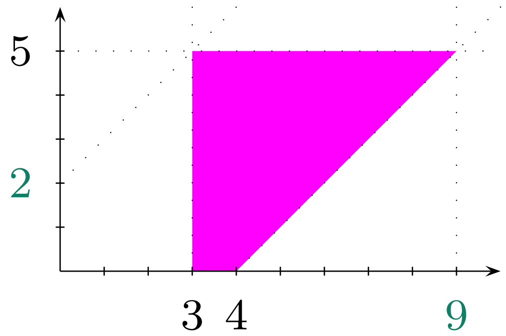
The DBM data structure

DBM (Difference Bounded Matrice) data structure [Dill89]

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

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

- Existence of a normal form



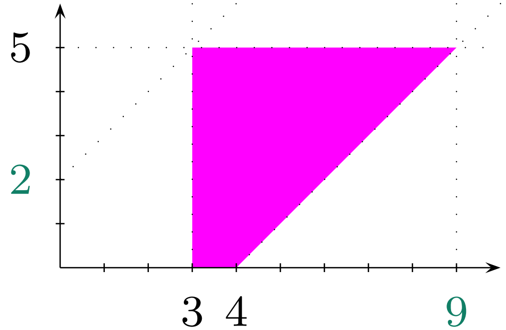
$$\begin{pmatrix} 0 & \mathbf{-3} & 0 \\ 9 & 0 & \mathbf{4} \\ \mathbf{5} & 2 & 0 \end{pmatrix}$$

The DBM data structure

DBM (Difference Bounded Matrice) data structure [Dill89]

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

- Existence of a normal form



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

- All previous operations on zones can be computed using DBMs

The extrapolation operator

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

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

- “intuitively”, erase non-relevant constraints

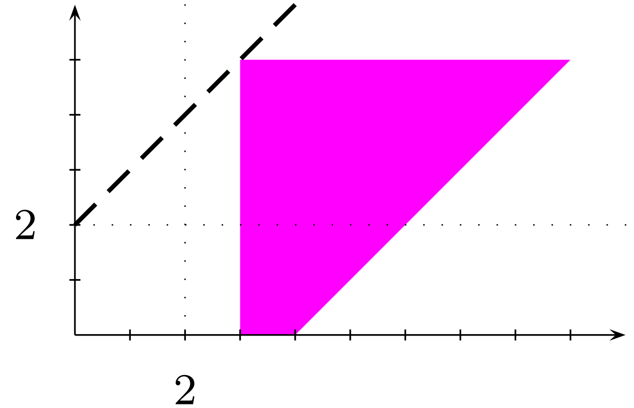
►► ensures termination

The extrapolation operator

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

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

- “intuitively”, erase non-relevant constraints



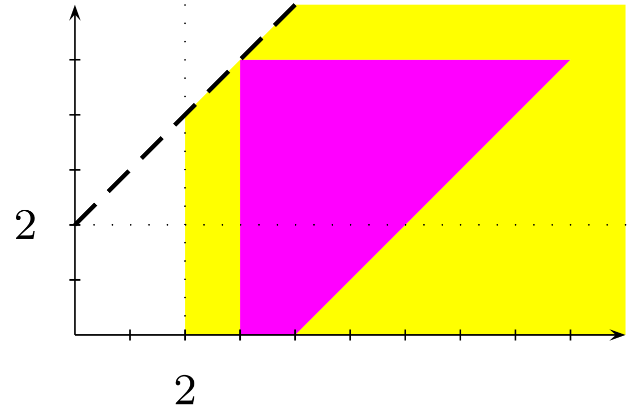
►► ensures termination

The extrapolation operator

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

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

- “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:

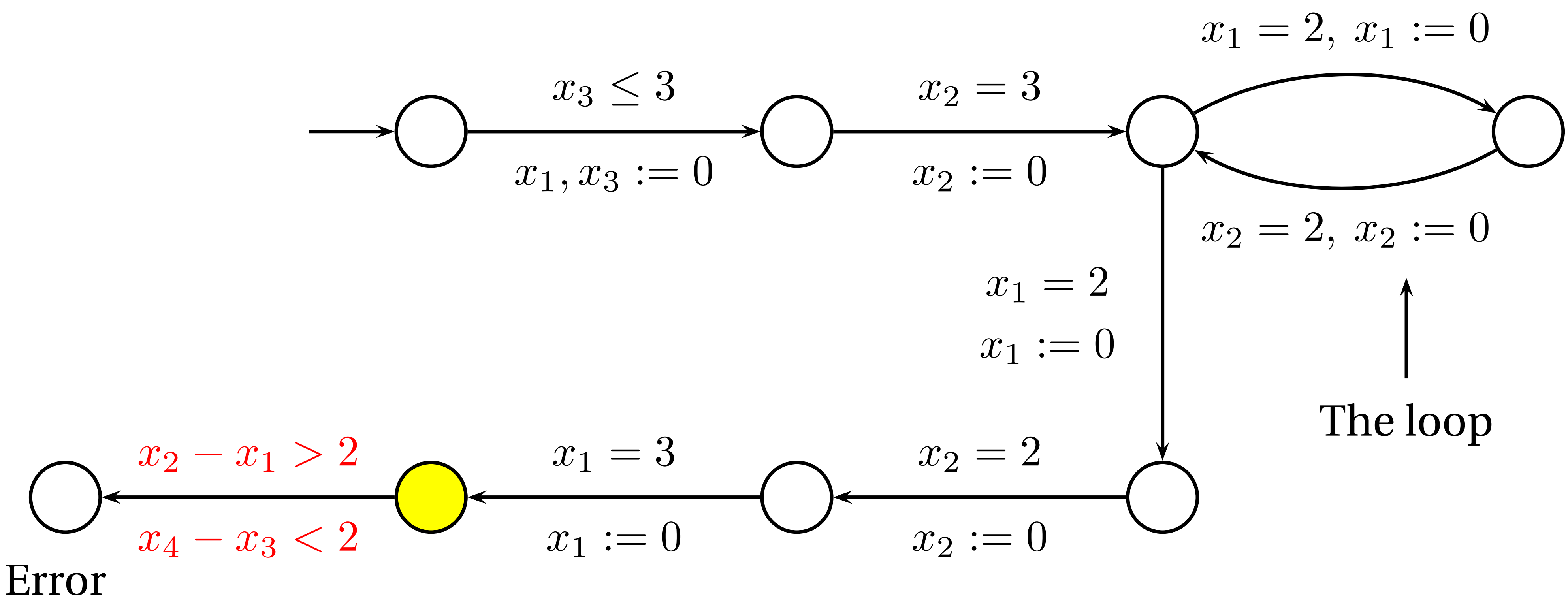
- keep the correctness of the forward computation

Solution by the past: maximal constant appearing in the automaton

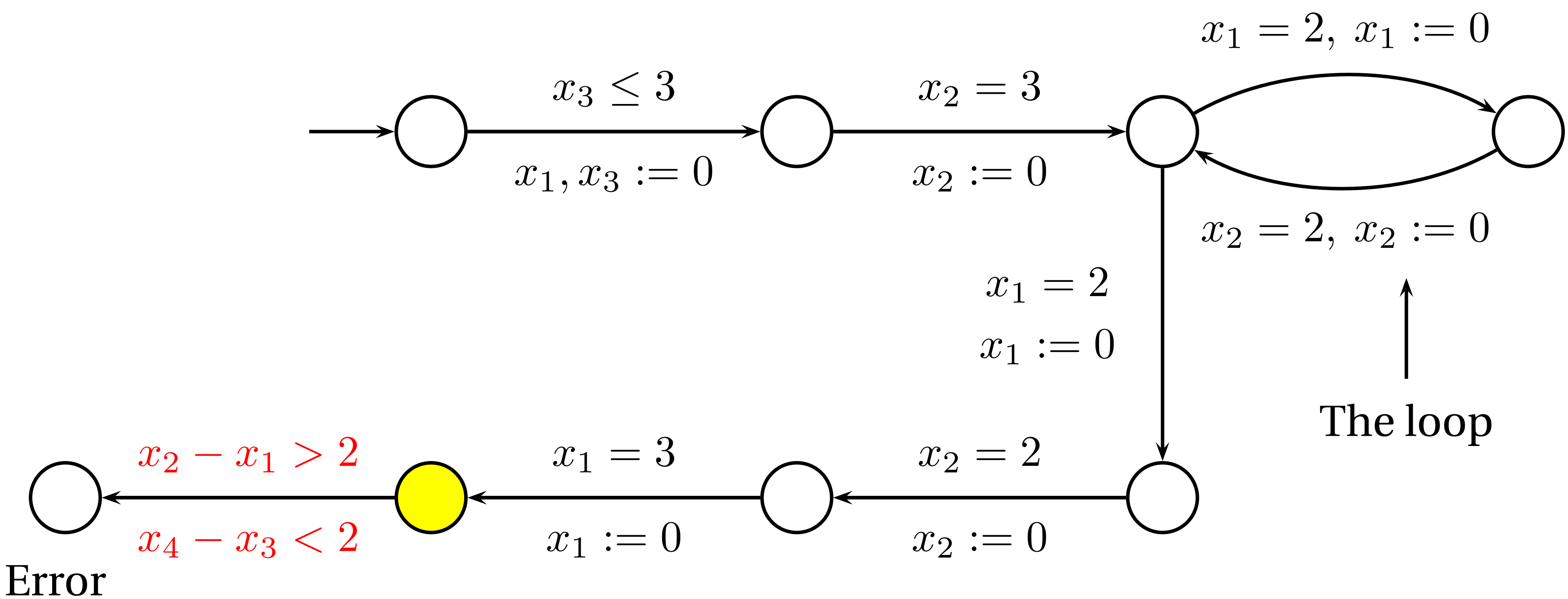
- Several correctness proofs can be found
- Implemented in tools like UPPAAL, KRONOS, RT-SPIN...
- Successfully used on real-life examples

However...

A problematic automaton

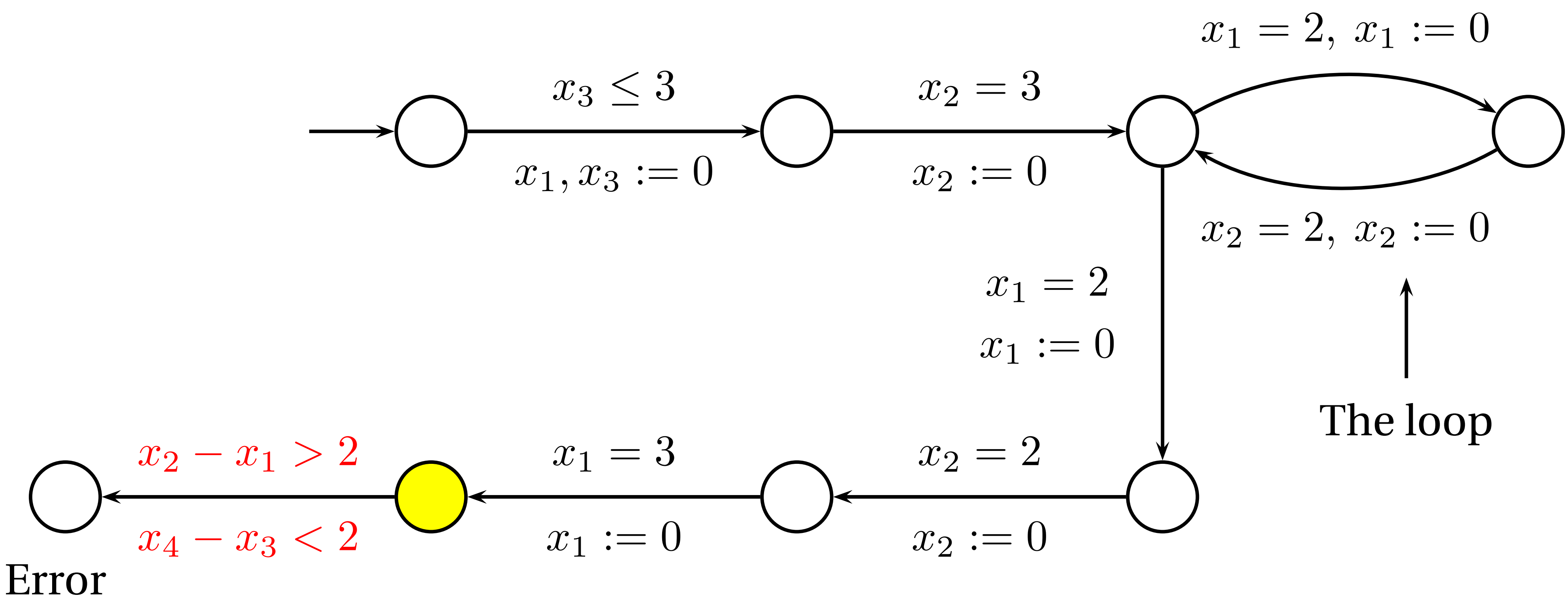


A problematic automaton

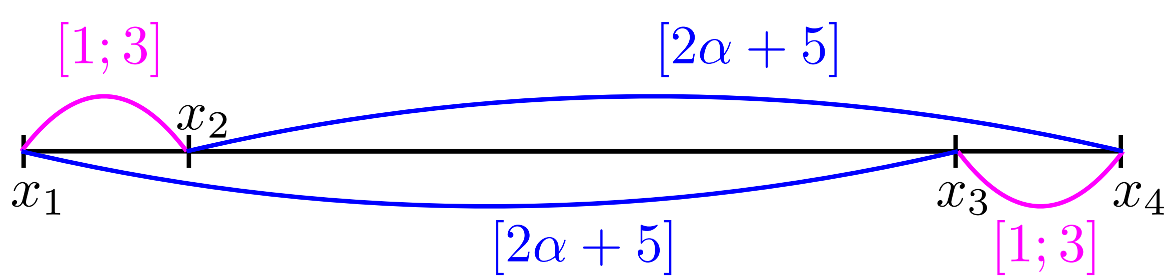


$$\begin{cases} v(x_1) = 0 \\ v(x_2) = d \\ v(x_3) = 2\alpha + 5 \\ v(x_4) = 2\alpha + 5 + d \end{cases}$$

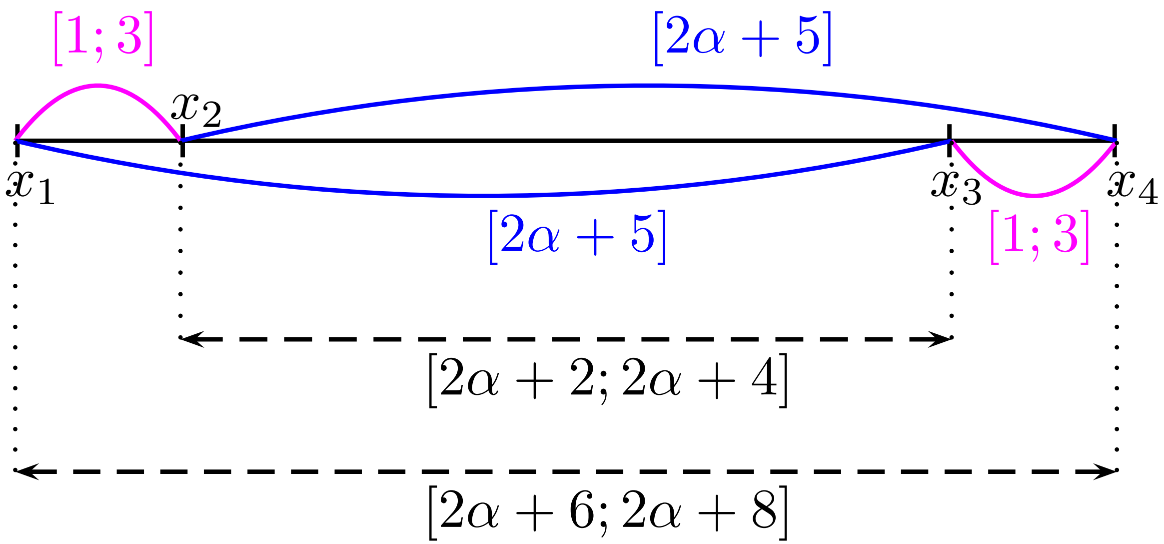
A problematic automaton



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

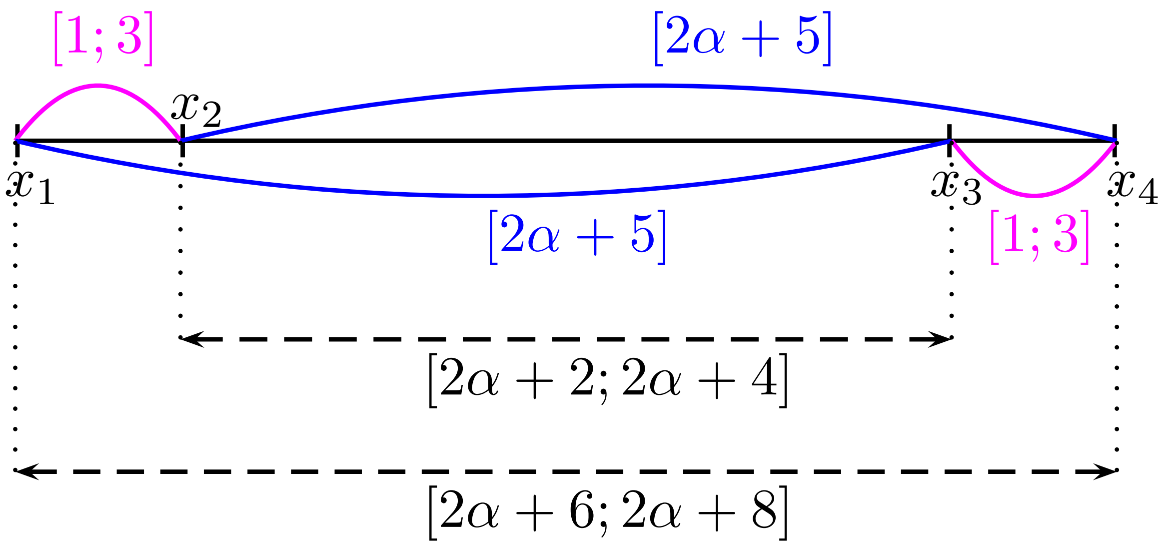


The problematic zone



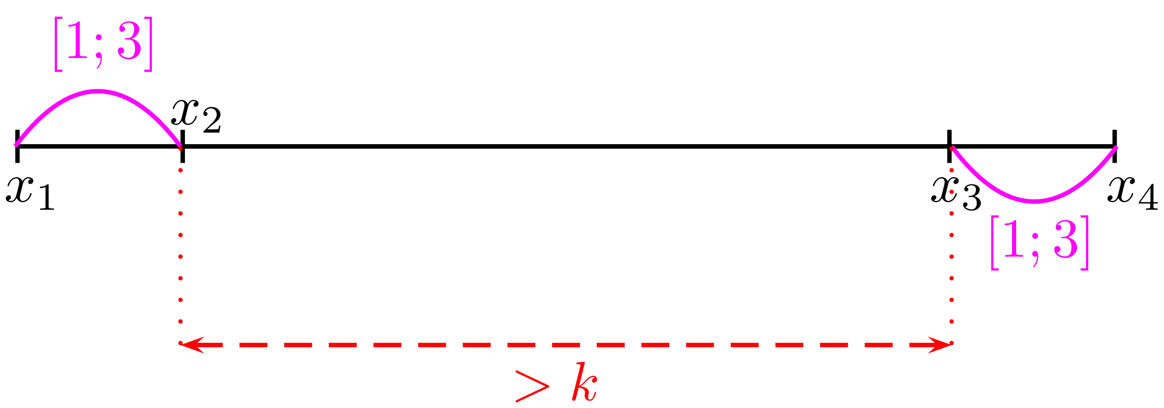
implies $x_1 - x_2 = x_3 - x_4$.

The problematic zone



implies $x_1 - x_2 = x_3 - x_4$.

If α is sufficiently large, after extrapolation:



does not imply

$x_1 - x_2 = x_3 - x_4$.

General abstractions

Criteria for a good abstraction operator Abs:

[Bouyer03]

General abstractions

Criteria for a good abstraction operator Abs**:**

- easy computation [Effectiveness]
  $\mathrm{Abs}(Z)$ is a zone if Z is a zone

[Bouyer03]

General abstractions

Criteria for a good abstraction operator Abs:

- easy computation [Effectiveness]
  $\mathrm{Abs}(Z)$ is a zone if Z is a zone
- finiteness of the abstraction [Termination]
  $\{\mathrm{Abs}(Z) \mid Z \text{ zone}\}$ is finite

[Bouyer03]

General abstractions

Criteria for a good abstraction operator Abs:

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

[Bouyer03]

General abstractions

Criteria for a good abstraction operator Abs:

- easy computation [Effectiveness]
  $\mathrm{Abs}(Z)$ is a zone if Z is a zone
- finiteness of the abstraction [Termination]
  $\{\mathrm{Abs}(Z) \mid Z \text{ zone}\}$ 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

[Bouyer03]

General abstractions

Criteria for a good abstraction operator Abs:

- easy computation [Effectiveness]
  $\mathrm{Abs}(Z)$ is a zone if Z is a zone
- finiteness of the abstraction [Termination]
  $\{\mathrm{Abs}(Z) \mid Z \text{ zone}\}$ 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

For the previous automaton,

no abstraction operator can satisfy all these criteria!

[Bouyer03]

Why that?

Assume there is a “nice” operator Abs.

The set $\{M$ DBM representing a zone $\mathrm{Abs}(Z)\}$ 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.

[Bouyer03]

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...)

[Bouyer03]

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.* To appear in JLAP 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.)

[Bouyer03] Bouyer. *Untameable Timed Automata!* To appear in STACS'03.

[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/`