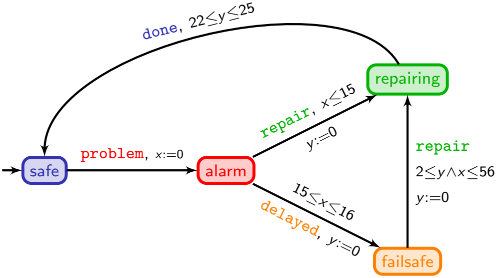
Timed automata – Decidability issues

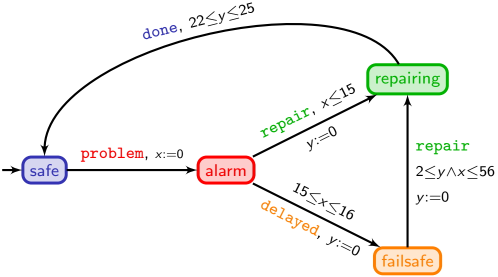
Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France

An example of a timed automaton

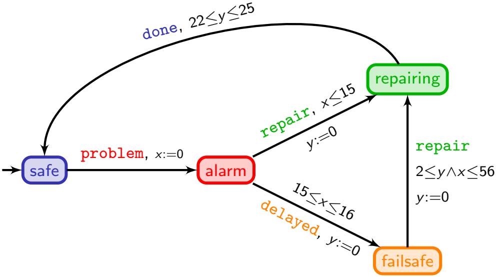


An example of a timed automaton



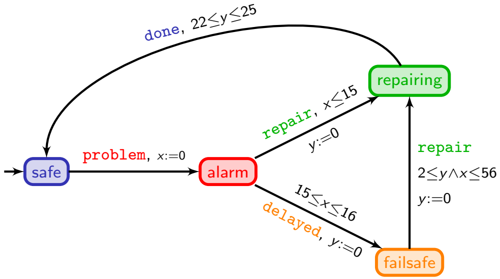
	safe
x	0
y	0

An example of a timed automaton



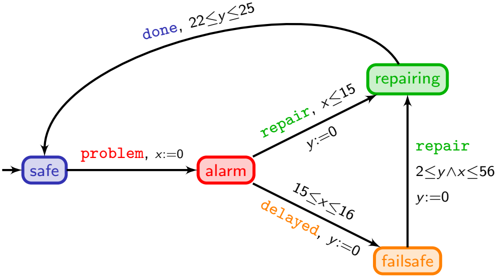
	safe	$\xrightarrow{23}$	safe
x	0		23
y	0		23

An example of a timed automaton



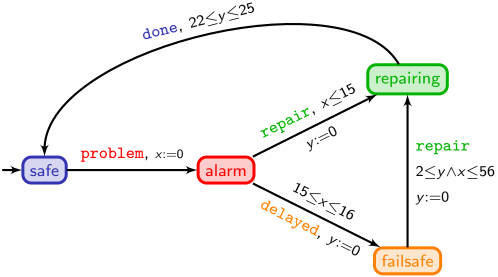
	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm
x	0		23		0
y	0		23		23

An example of a timed automaton



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm
x	0		23		0		15.6
y	0		23		23		38.6

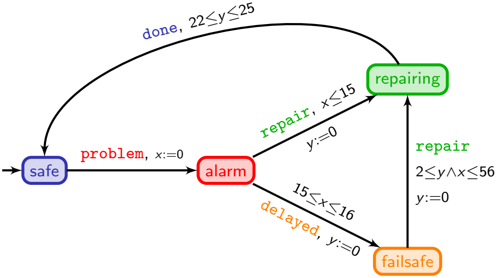
An example of a timed automaton



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	⋯
y	0		23		23		38.6		0	

	failsafe
⋯	15.6
	0

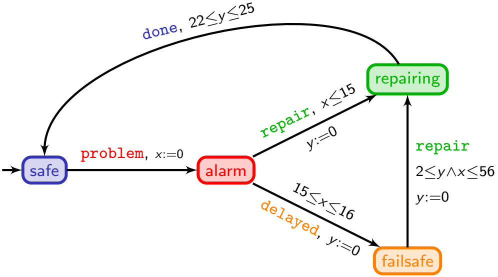
An example of a timed automaton



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	⋯
y	0		23		23		38.6		0	

	failsafe	$\xrightarrow{2.3}$	failsafe
⋯	15.6		17.9
	0		2.3

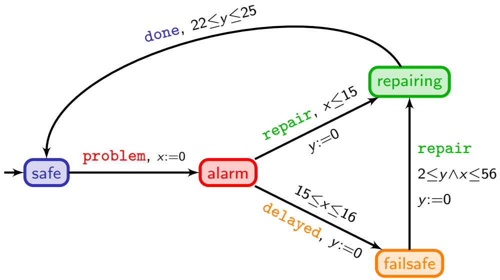
An example of a timed automaton



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	⋯
y	0		23		23		38.6		0	

	failsafe	$\xrightarrow{2.3}$	failsafe	$\xrightarrow{\texttt{repair}}$	repairing
⋯	15.6		17.9		17.9
	0		2.3		0

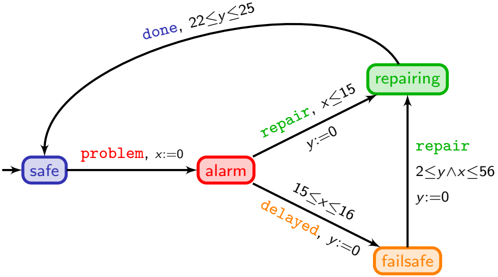
An example of a timed automaton



$$\begin{array}{cccccccccccc}
 & \text{safe} & \xrightarrow{23} & \text{safe} & \xrightarrow{\texttt{problem}} & \text{alarm} & \xrightarrow{15.6} & \text{alarm} & \xrightarrow{\texttt{delayed}} & \text{failsafe} & \\
\text{x} & 0 & & 23 & & 0 & & 15.6 & & 15.6 & \cdots \\
\text{y} & 0 & & 23 & & 23 & & 38.6 & & 0 &
\end{array}$$

$$\begin{array}{cccccccc}
 & \text{failsafe} & \xrightarrow{2.3} & \text{failsafe} & \xrightarrow{\texttt{repair}} & \text{repairing} & \xrightarrow{22.1} & \text{repairing} \\
\cdots & 15.6 & & 17.9 & & 17.9 & & 40 \\
 & 0 & & 2.3 & & 0 & & 22.1
\end{array}$$

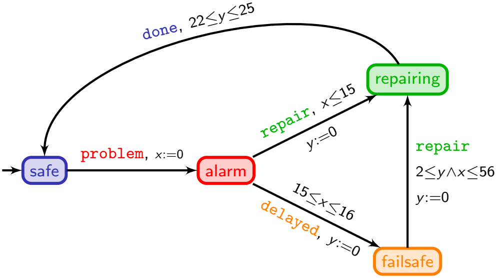
An example of a timed automaton



$$\begin{array}{cccccccccccc}
 & \text{safe} & \xrightarrow{23} & \text{safe} & \xrightarrow{\texttt{problem}} & \text{alarm} & \xrightarrow{15.6} & \text{alarm} & \xrightarrow{\texttt{delayed}} & \text{failsafe} & \\
x & 0 & & 23 & & 0 & & 15.6 & & 15.6 & \cdots \\
y & 0 & & 23 & & 23 & & 38.6 & & 0 &
\end{array}$$

$$\begin{array}{cccccccccccc}
 & \text{failsafe} & \xrightarrow{2.3} & \text{failsafe} & \xrightarrow{\texttt{repair}} & \text{repairing} & \xrightarrow{22.1} & \text{repairing} & \xrightarrow{\texttt{done}} & \text{safe} \\
\cdots & 15.6 & & 17.9 & & 17.9 & & 40 & & 40 \\
 & 0 & & 2.3 & & 0 & & 22.1 & & 22.1
\end{array}$$

An example of a timed automaton



	safe	$\xrightarrow{23}$	safe	$\xrightarrow{\texttt{problem}}$	alarm	$\xrightarrow{15.6}$	alarm	$\xrightarrow{\texttt{delayed}}$	failsafe	
x	0		23		0		15.6		15.6	…
y	0		23		23		38.6		0	

	failsafe	$\xrightarrow{2.3}$	failsafe	$\xrightarrow{\texttt{repair}}$	repairing	$\xrightarrow{22.1}$	repairing	$\xrightarrow{\texttt{done}}$	safe
…	15.6		17.9		17.9		40		40
	0		2.3		0		22.1		22.1

This run reads the timed word
$(\texttt{problem}, 23)(\texttt{delayed}, 38.6)(\texttt{repair}, 40.9), (\texttt{done}, 63)$.

Outline

1. Decidability of basic properties
2. Equivalence (or preorder) checking
3. Some extensions of timed automata

Verification

Emptiness problem

Is the language accepted by a timed automaton empty?

- basic reachability/safety properties (final states)
- basic liveness properties (ω-regular conditions)

Verification

Emptiness problem

Is the language accepted by a timed automaton empty?

- Problem: the set of configurations is infinite
  $\rightsquigarrow$ classical methods for finite-state systems cannot be applied

Verification

Emptiness problem

Is the language accepted by a timed automaton empty?

- Problem: the set of configurations is infinite
  $\rightsquigarrow$ classical methods for finite-state systems cannot be applied
- Positive key point: variables (clocks) increase at the same speed

Verification

Emptiness problem

Is the language accepted by a timed automaton empty?

- Problem: the set of configurations is infinite
  $\rightsquigarrow$ classical methods for finite-state systems cannot be applied
- Positive key point: variables (clocks) increase at the same speed

Theorem [AD90,AD94]

The emptiness problem for timed automata is decidable and PSPACE-complete.

[AD90] Alur, Dill. Automata for modeling real-time systems *(ICALP'90)*.
[AD94] Alur, Dill. A theory of timed automata *(Theoretical Computer Science)*.

Verification

Emptiness problem

Is the language accepted by a timed automaton empty?

- Problem: the set of configurations is infinite
  $\rightsquigarrow$ classical methods for finite-state systems cannot be applied
- Positive key point: variables (clocks) increase at the same speed

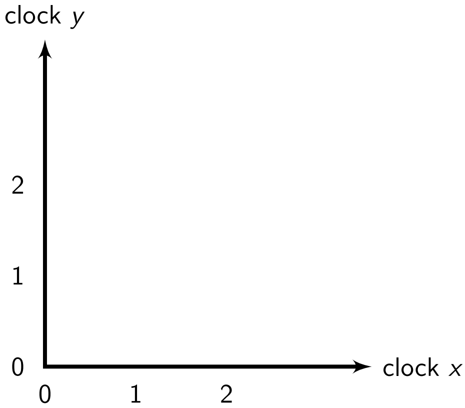
Theorem [AD90,AD94]

The emptiness problem for timed automata is decidable and PSPACE-complete.

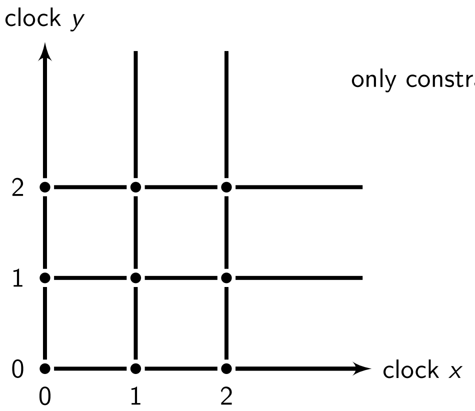
Method: construct a finite abstraction

[AD90] Alur, Dill. Automata for modeling real-time systems *(ICALP'90)*.
[AD94] Alur, Dill. A theory of timed automata *(Theoretical Computer Science)*.

The region abstraction



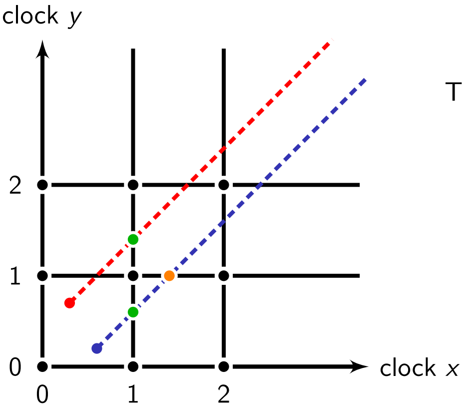
The region abstraction



only constraints: $x \sim c$ with $c \in \{0, 1, 2\}$
$y \sim c$ with $c \in \{0, 1, 2\}$

- "compatibility" between regions and constraints

The region abstraction

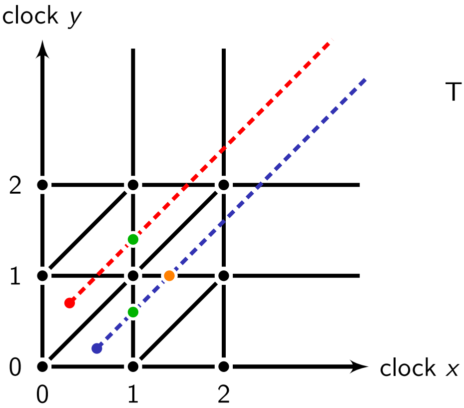


The path 

- can be fired from
- cannot be fired from

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

The region abstraction

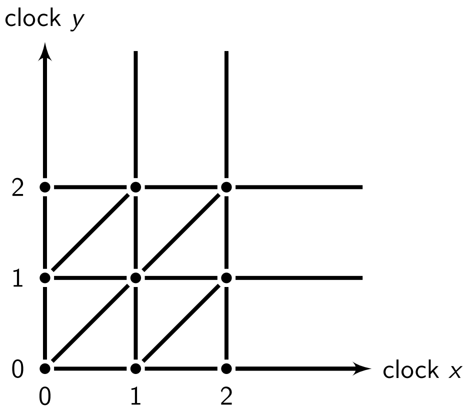


The path 

- can be fired from ●
- cannot be fired from ●

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

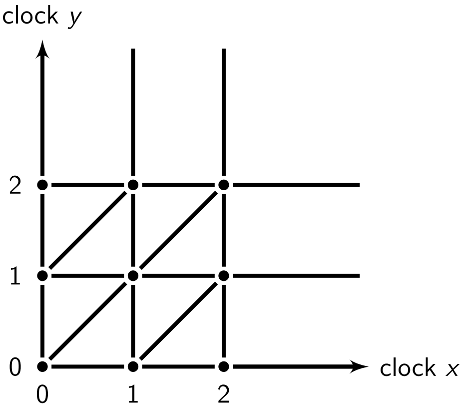
The region abstraction



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

$\rightsquigarrow$ an equivalence of finite index

The region abstraction



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

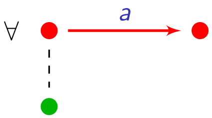
$\rightsquigarrow$ an equivalence of finite index
a time-abstract bisimulation

Time-abstract bisimulation

This is a relation between • and • such that:

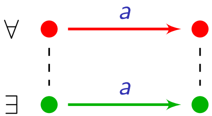
Time-abstract bisimulation

This is a relation between • and • such that:



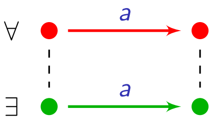
Time-abstract bisimulation

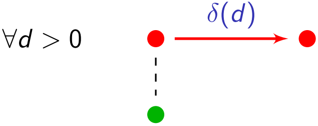
This is a relation between • and • such that:



Time-abstract bisimulation

This is a relation between • and • such that:





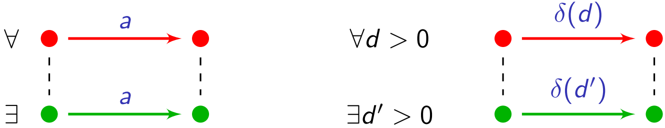
Time-abstract bisimulation

This is a relation between • and • such that:

$$\begin{tikzcd} \forall & \bullet \arrow[r, "a"] \arrow[d, dashed, no head] & \bullet \arrow[d, dashed, no head] \\ \exists & \bullet \arrow[r, "a"] & \bullet \end{tikzcd} \qquad \begin{tikzcd} \forall d > 0 & \bullet \arrow[r, "\delta(d)"] \arrow[d, dashed, no head] & \bullet \arrow[d, dashed, no head] \\ \exists d' > 0 & \bullet \arrow[r, "\delta(d')"] & \bullet \end{tikzcd}$$

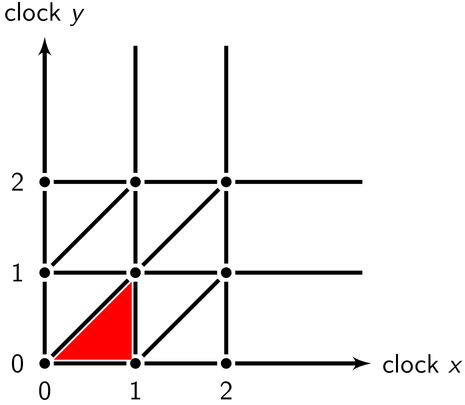
Time-abstract bisimulation

This is a relation between • and • such that:



... and vice-versa (swap • and •).

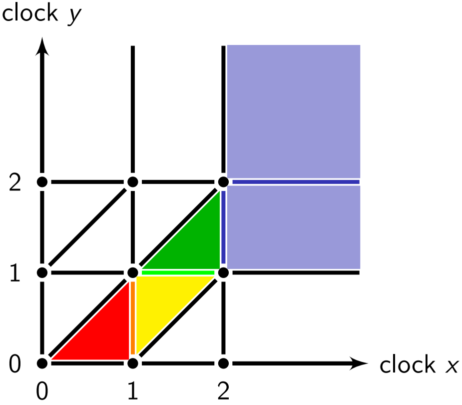
The region abstraction (2)



- region R defined by:

$$\begin{cases} 0 < x < 1 \\ 0 < y < 1 \\ y < x \end{cases}$$

The region abstraction (2)

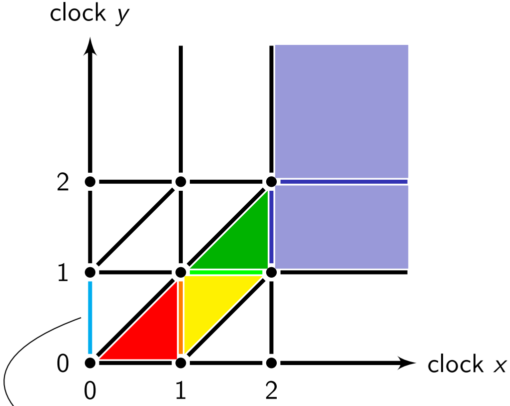


- region R defined by:

$$\begin{cases} 0 < x < 1 \\ 0 < y < 1 \\ y < x \end{cases}$$

- time successors of R

The region abstraction (2)



- region R defined by:

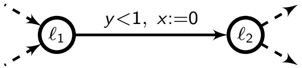
$$\begin{cases} 0 < x < 1 \\ 0 < y < 1 \\ y < x \end{cases}$$

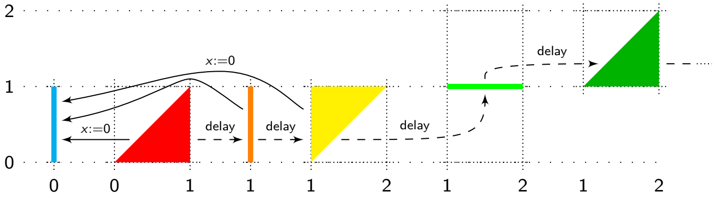
- time successors of R

image of R when resetting clock x

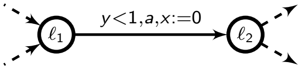
The construction of the region graph

It "mimicks" the behaviours of the clocks.



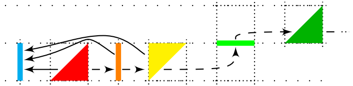


Region automaton $\equiv$ finite bisimulation quotient



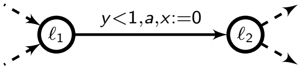
timed automaton

$\otimes$



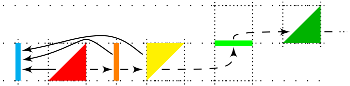
region graph

Region automaton $\equiv$ finite bisimulation quotient

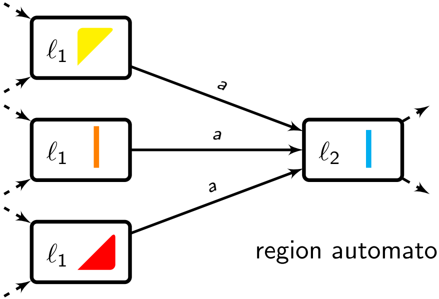


timed automaton

$\otimes$

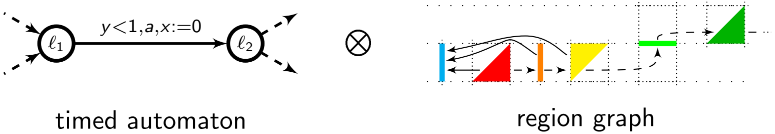


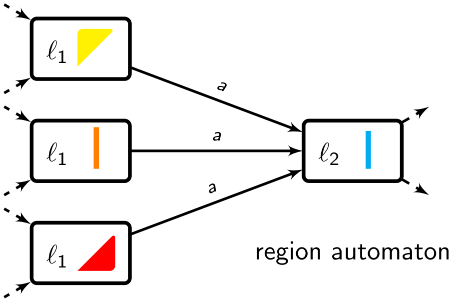
region graph



region automaton

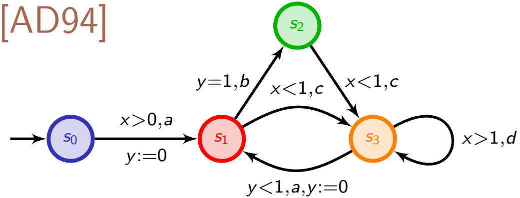
Region automaton ≡ finite bisimulation quotient



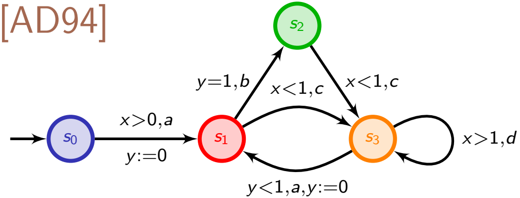


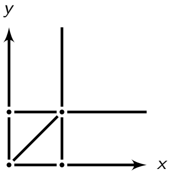
$\mathcal{L}(\text{reg. aut.}) = \text{UNTIME}(\mathcal{L}(\text{timed aut.}))$

An example [AD94]

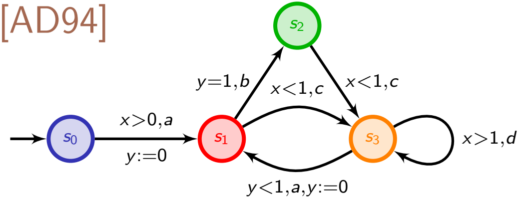


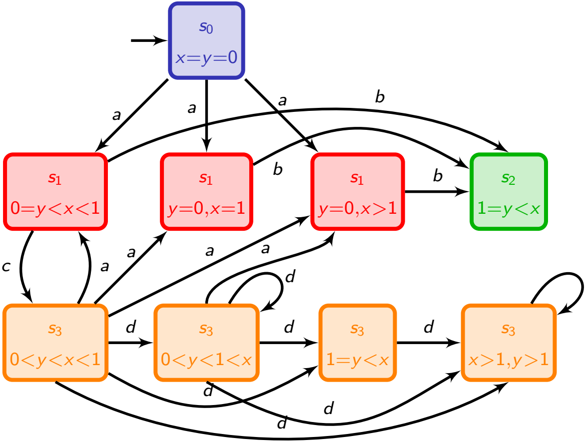
An example [AD94]

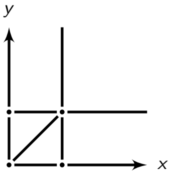


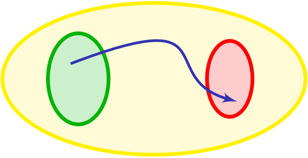


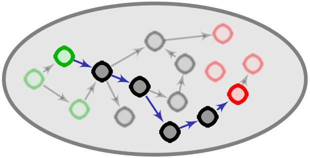
An example [AD94]





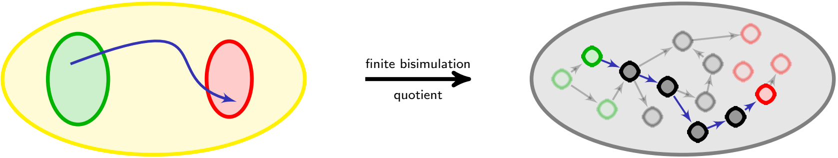






timed automaton

large (but finite) automaton
(region automaton)

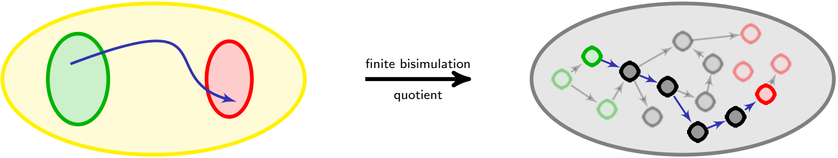


timed automaton

large (but finite) automaton
(region automaton)

- large: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

$$\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}$$



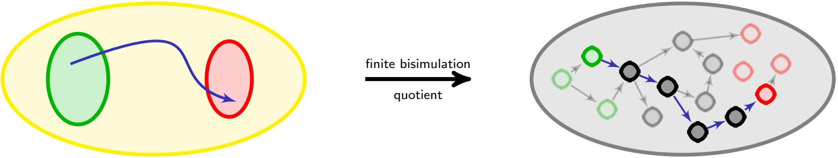
timed automaton

large (but finite) automaton
(region automaton)

- large: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

$$\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}$$

- It can be used to check for:



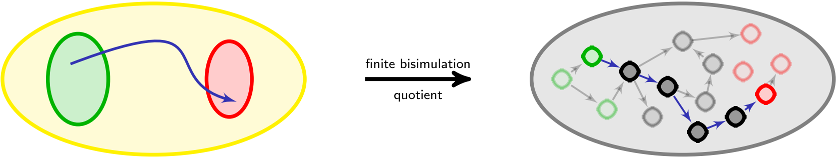
timed automaton

large (but finite) automaton
(region automaton)

- large: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

$$\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}$$

- It can be used to check for:
  - reachability/safety properties



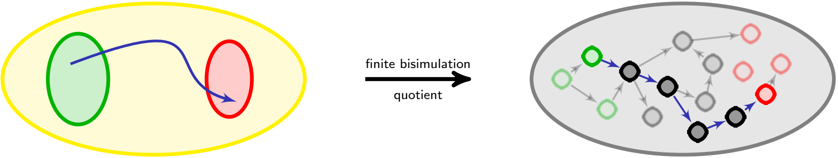
timed automaton

large (but finite) automaton
(region automaton)

- large: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

$$\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}$$

- It can be used to check for:
  - reachability/safety properties
  - liveness properties (like Büchi properties)



timed automaton

large (but finite) automaton
(region automaton)

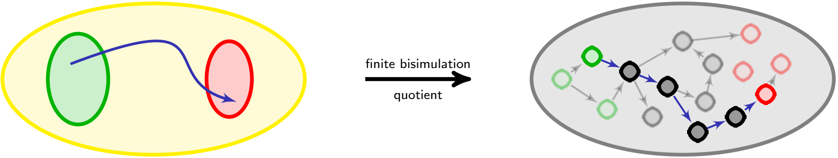
- large: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

$$\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}$$

- It can be used to check for:
  - reachability/safety properties
  - liveness properties (like Büchi properties)

$\rightsquigarrow$ problems with Zeno behaviours?

(infinitely many actions in bounded time)



timed automaton

large (but finite) automaton
(region automaton)

- large: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

$$\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}$$

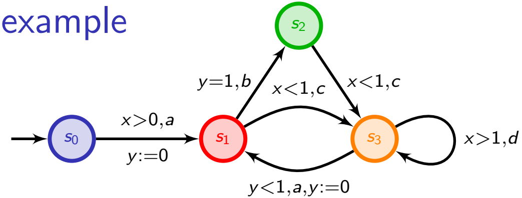
- It can be used to check for:
  - reachability/safety properties
  - liveness properties (like Büchi properties)

$\rightsquigarrow$ problems with Zeno behaviours?

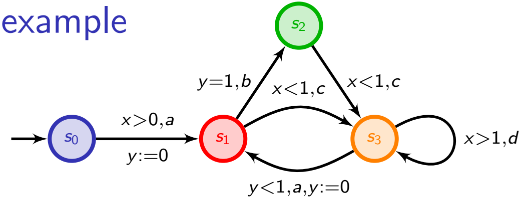
(infinitely many actions in bounded time)

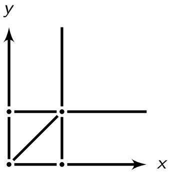
NB: standard problem in timed automata...

Back to the example

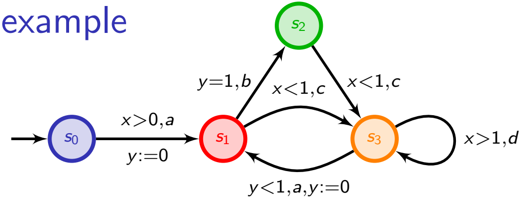


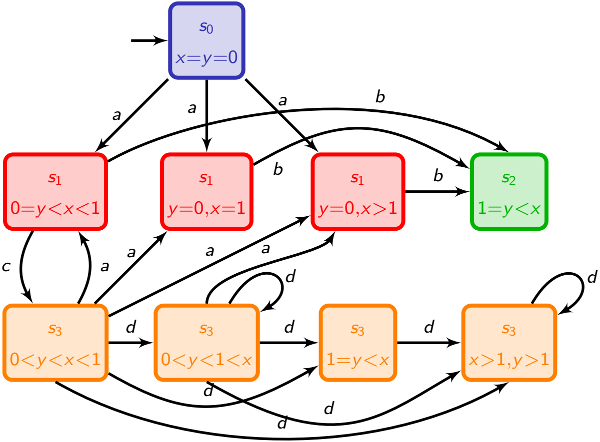
Back to the example

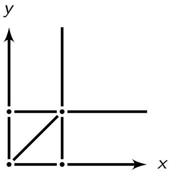




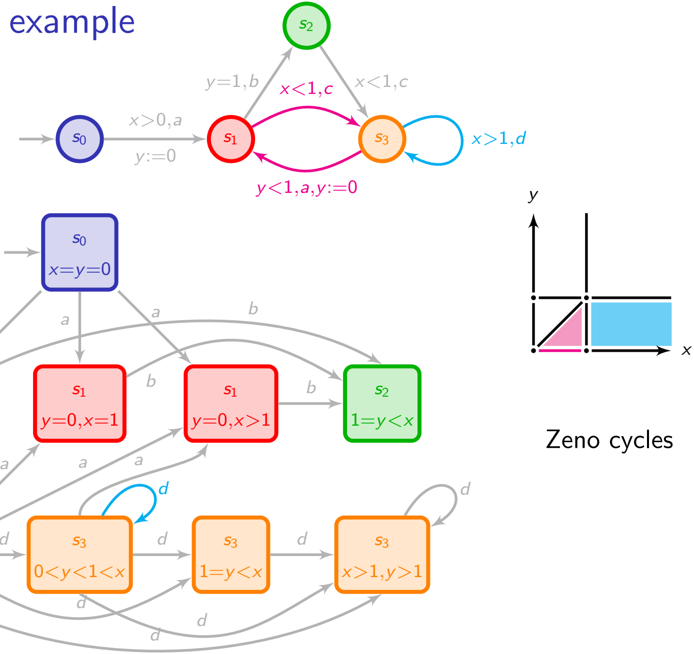
Back to the example





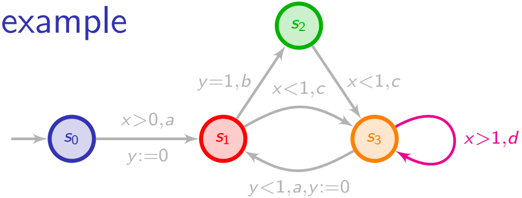


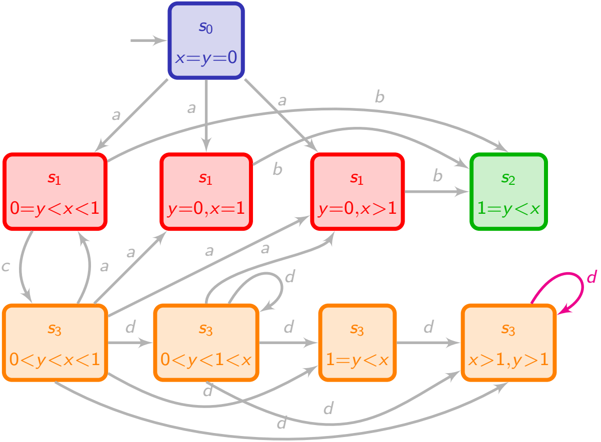
Back to the example

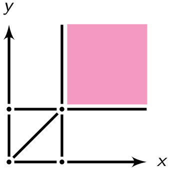


Zeno cycles

Back to the example







Cycles with non-Zeno behaviours

Outline

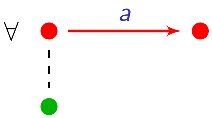
1. Decidability of basic properties
2. Equivalence (or preorder) checking
3. Some extensions of timed automata

Strong timed (bi)simulation

This is a relation between • and • such that:

Strong timed (bi)simulation

This is a relation between • and • such that:



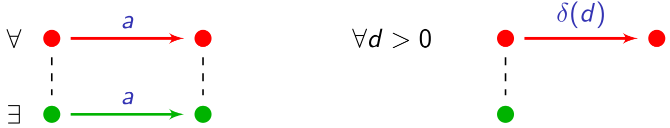
Strong timed (bi)simulation

This is a relation between • and • such that:

$$
\begin{tikzcd}
\forall & \bullet \arrow[r, "a"] \arrow[d, dash, dashed] & \bullet \arrow[d, dash, dashed] \\
\exists & \bullet \arrow[r, "a"] & \bullet
\end{tikzcd}
$$

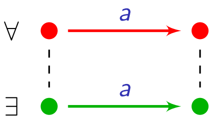
Strong timed (bi)simulation

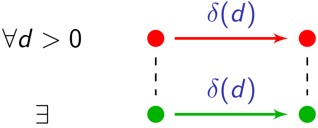
This is a relation between • and • such that:



Strong timed (bi)simulation

This is a relation between • and • such that:





Strong timed (bi)simulation

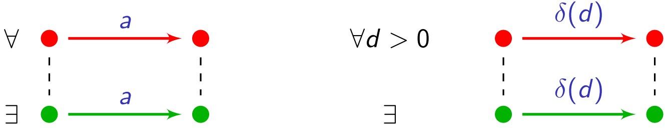
This is a relation between $\bullet$ and $\bullet$ such that:

$$\begin{tikzcd}
\forall & \bullet \arrow[r, "a"] \arrow[d, dash, dashed] & \bullet \arrow[d, dash, dashed] & & \forall d > 0 & \bullet \arrow[r, "\delta(d)"] \arrow[d, dash, dashed] & \bullet \arrow[d, dash, dashed] \\
\exists & \bullet \arrow[r, "a"] & \bullet & & \exists & \bullet \arrow[r, "\delta(d)"] & \bullet
\end{tikzcd}$$

... and vice-versa (swap $\bullet$ and $\bullet$) for the bisimulation relation.

Strong timed (bi)simulation

This is a relation between • and • such that:



... and vice-versa (swap • and •) for the bisimulation relation.

Theorem

Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete.

(see later for a simple proof of the upper bound)

Language (or trace) equivalence and inclusion

Question

Given two timed automata $\mathcal{A}$ and $\mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$ (resp. $L(\mathcal{A}) \subseteq L(\mathcal{B})$)?

Language (or trace) equivalence and inclusion

Question

Given two timed automata $\mathcal{A}$ and $\mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$ (resp. $L(\mathcal{A}) \subseteq L(\mathcal{B})$)?

Theorem [AD90,AD94]

Language equivalence and language inclusion are undecidable in timed automata.

... as a special case of the universality problem (are all timed words accepted by the automaton?).

[AD90] Alur, Dill. Automata for modeling real-time systems *(ICALP'90)*.
[AD94] Alur, Dill. A theory of timed automata *(Theoretical Computer Science)*.

Language (or trace) equivalence and inclusion

Question

Given two timed automata $\mathcal{A}$ and $\mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$ (resp. $L(\mathcal{A}) \subseteq L(\mathcal{B})$)?

Theorem [AD90,AD94]

Language equivalence and language inclusion are undecidable in timed automata.

... as a special case of the universality problem (are all timed words accepted by the automaton?).

$\rightsquigarrow$ Proof by reduction from the recurring problem of a two-counter machine

[AD90] Alur, Dill. Automata for modeling real-time systems *(ICALP'90)*.
[AD94] Alur, Dill. A theory of timed automata *(Theoretical Computer Science)*.

Undecidability of universality

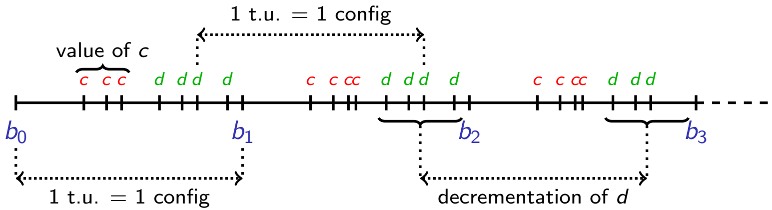
Theorem [AD90,AD94]

Universality of timed automata is undecidable.

Undecidability of universality

Theorem [AD90,AD94]

Universality of timed automata is undecidable.

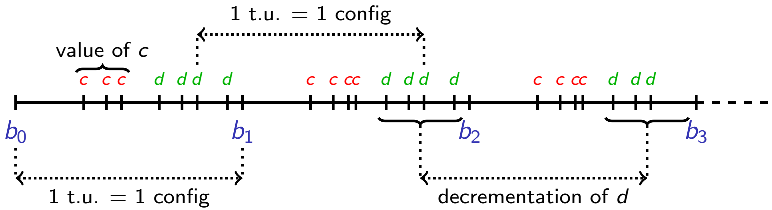


- one configuration is encoded in one time unit
- number of c's: value of counter c
- number of d's: value of counter d
- one time unit between two corresponding c's (resp. d's)

Undecidability of universality

Theorem [AD90,AD94]

Universality of timed automata is undecidable.

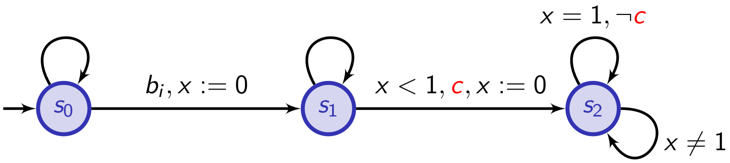


- one configuration is encoded in one time unit
- number of c's: value of counter c
- number of d's: value of counter d
- one time unit between two corresponding c's (resp. d's)

$\rightsquigarrow$ We encode "non-behaviours" of a two-counter machine

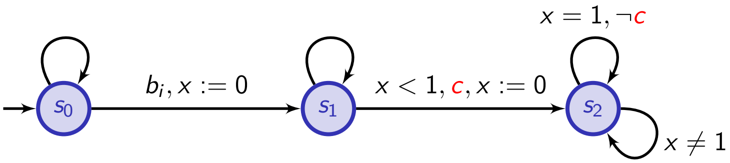
Example

Module to check that if instruction i does not decrease counter c, then all actions c appearing less than 1 t.u. after b_i has to be followed by an other c 1 t.u. later.



Example

Module to check that if instruction i does not decrease counter c, then all actions c appearing less than 1 t.u. after b_i has to be followed by an other c 1 t.u. later.



The union of all small modules is not universal
iff
The two-counter machine has a recurring computation

Bad news

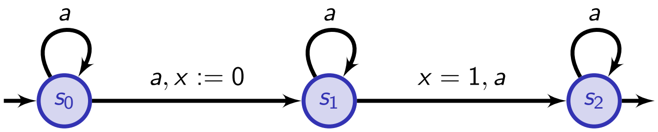
- Language inclusion is undecidable [AD90,AD94]
  (Bad news for the application to verification)
- Complementability is undecidable [Tri03,Fin06]
- ...

[Tri03] Tripakis. Folk theorems on the determinization and minimization of timed automata *(FORMATS'03)*.
[Fin06] Finkel. Undecidable problems about timed automata *(FORMATS'06)*.

Bad news

- Language inclusion is undecidable [AD90,AD94]
  (Bad news for the application to verification)
- Complementability is undecidable [Tri03,Fin06]
- ...

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

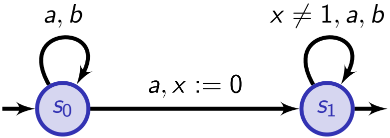


[Tri03] Tripakis. Folk theorems on the determinization and minimization of timed automata *(FORMATS'03)*.
[Fin06] Finkel. Undecidable problems about timed automata *(FORMATS'06)*.

Bad news

- Language inclusion is undecidable [AD90,AD94]
  (Bad news for the application to verification)
- Complementability is undecidable [Tri03,Fin06]
- ...

An example of non-determinizable/non-complementable aut.: [AM04]

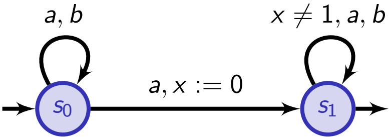


[Tri03] Tripakis. Folk theorems on the determinization and minimization of timed automata *(FORMATS'03)*.
[Fin06] Finkel. Undecidable problems about timed automata *(FORMATS'06)*.
[AM04] Alur, Madhusudan. Decision problems for timed automata: A survey *(SFM-04:RT))*.

Bad news

- Language inclusion is undecidable [AD90,AD94]
  (Bad news for the application to verification)
- Complementability is undecidable [Tri03,Fin06]
- ...

An example of non-determinizable/non-complementable aut.: [AM04]



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

[Tri03] Tripakis. Folk theorems on the determinization and minimization of timed automata *(FORMATS'03)*.
[Fin06] Finkel. Undecidable problems about timed automata *(FORMATS'06)*.
[AM04] Alur, Madhusudan. Decision problems for timed automata: A survey *(SFM-04:RT))*.

Outline

1. Decidability of basic properties
2. Equivalence (or preorder) checking
3. Some extensions of timed automata

What if we extend the clock constraints?

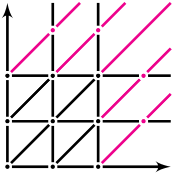
- Diagonal constraints (*i.e.* $x - y \leq 2$)

What if we extend the clock constraints?

- Diagonal constraints (*i.e.* $x - y \leq 2$)
  - decidable (with the same complexity)

What if we extend the clock constraints?

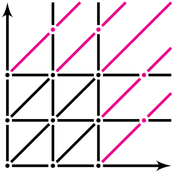
- Diagonal constraints (*i.e.* $x - y \leq 2$)
  - decidable (with the same complexity)



is also a time-abstract bisimulation!

What if we extend the clock constraints?

- Diagonal constraints (*i.e.* $x - y \leq 2$)
  - decidable (with the same complexity)

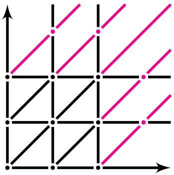


is also a time-abstract bisimulation!

- Linear constraints (*i.e.* $2x + 3y \sim 5$)

What if we extend the clock constraints?

- Diagonal constraints (*i.e.* $x - y \leq 2$)
  - decidable (with the same complexity)

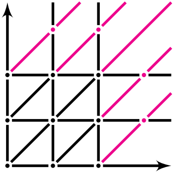


is also a time-abstract bisimulation!

- Linear constraints (*i.e.* $2x + 3y \sim 5$)
  - undecidable in general

What if we extend the clock constraints?

- Diagonal constraints (*i.e.* $x - y \leq 2$)
  - decidable (with the same complexity)

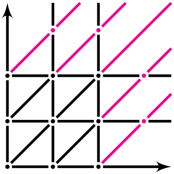


is also a time-abstract bisimulation!

- Linear constraints (*i.e.* $2x + 3y \sim 5$)
  - undecidable in general
  - only decidable in few cases

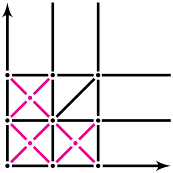
What if we extend the clock constraints?

- Diagonal constraints (*i.e.* $x - y \leq 2$)
  - decidable (with the same complexity)



is also a time-abstract bisimulation!

- Linear constraints (*i.e.* $2x + 3y \sim 5$)
  - undecidable in general
  - only decidable in few cases



is a time-abstract bisimulation (when two clocks x and y and constraints $x + y \sim c$)!

What if we allow more operations on clocks?

- that can be transfer operations (*i.e.* $x := y$), or reinitialization operations (*i.e.* $x := 4$), or ... [BDFP04]

[BDFP04] Bouyer, Dufourd, Fleury, Petit. Updatable Timed Automata *(Theoretical Computer Science)*.

What if we allow more operations on clocks?

- that can be transfer operations (*i.e.* $x := y$), or reinitialization operations (*i.e.* $x := 4$), or ... [BDFP04]

	simple constraints	+ diagonal constraints
$x := c$, $x := y$		
$x := x + 1$		
$x := y + c$		
$x := x - 1$		
$x :< c$		
$x :> c$		
$x :\sim y + c$		
$y + c <: x :< y + d$		
$y + c <: x :< z + d$		

[BDFP04] Bouyer, Dufourd, Fleury, Petit. Updatable Timed Automata *(Theoretical Computer Science)*.

What if we allow more operations on clocks?

- that can be transfer operations (*i.e.* $x := y$), or reinitialization operations (*i.e.* $x := 4$), or ... [BDFP04]

	simple constraints	+ diagonal constraints
$x := c,\ x := y$	decidable	decidable
$x := x + 1$		undecidable
$x := y + c$		
$x := x - 1$	undecidable	
$x :< c$	decidable	decidable
$x :> c$		undecidable
$x :\sim y + c$		
$y + c <: x :< y + d$		
$y + c <: x :< z + d$	undecidable	

[BDFP04] Bouyer, Dufourd, Fleury, Petit. Updatable Timed Automata *(Theoretical Computer Science)*.

What if we allow more operations on clocks?

- that can be transfer operations (*i.e.* $x := y$), or reinitialization operations (*i.e.* $x := 4$), or ... [BDFP04]

<table>
<tr><th></th><th>simple constraints</th><th>+ diagonal constraints</th></tr>
<tr><td>$x := c,\ x := y$</td><td rowspan="3">decidable</td><td>decidable</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">decidable</td><td>decidable</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>

$\rightsquigarrow$ need of being very careful when using more operations on clocks!

[BDFP04] Bouyer, Dufourd, Fleury, Petit. Updatable Timed Automata *(Theoretical Computer Science)*.

A note on hybrid automata (see more on Friday)

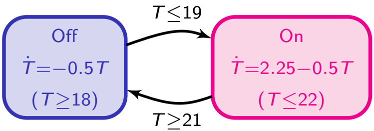
$$\begin{array}{ll} & \text{a discrete control (the mode of the system)} \\ + & \text{continuous evolution of the variables within a mode} \end{array}$$

[HKPV95] Henzinger, Kopke, Puri, Varaiya. What's decidable wbout hybrid automata? *(SToC'95)*.

A note on hybrid automata (see more on Friday)

$$\begin{array}{ll} & \text{a discrete control (the mode of the system)} \\ + & \text{continuous evolution of the variables within a mode} \end{array}$$

The thermostat example

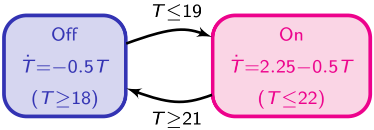


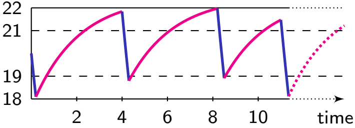
[HKPV95] Henzinger, Kopke, Puri, Varaiya. What's decidable wbout hybrid automata? *(SToC'95)*.

A note on hybrid automata (see more on Friday)

a discrete control (the mode of the system)
+ continuous evolution of the variables within a mode

The thermostat example



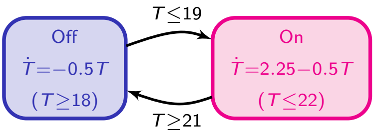


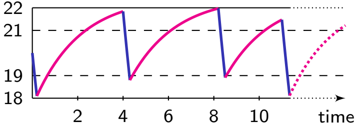
[HKPV95] Henzinger, Kopke, Puri, Varaiya. What's decidable wbout hybrid automata? *(SToC'95)*.

A note on hybrid automata (see more on Friday)

a discrete control (the mode of the system)
+ continuous evolution of the variables within a mode

The thermostat example





Theorem [HKPV95]

The reachability problem is undecidable in hybrid automata, even for stopwatch automata.

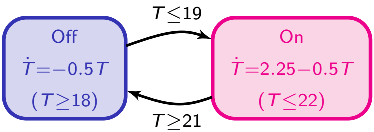
(stopwatch automata: timed automata in which clocks can be stopped)

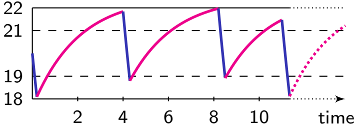
[HKPV95] Henzinger, Kopke, Puri, Varaiya. What's decidable wbout hybrid automata? *(SToC'95)*.

A note on hybrid automata (see more on Friday)

a discrete control (the mode of the system)
$+$ continuous evolution of the variables within a mode

The thermostat example





Theorem [HKPV95]

The reachability problem is undecidable in hybrid automata, even for stopwatch automata.

(stopwatch automata: timed automata in which clocks can be stopped)

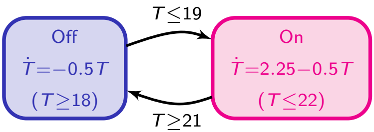
A relevant question

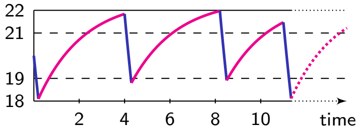
Is there something between timed automata and hybrid automata which is decidable?

A note on hybrid automata (see more on Friday)

$$\begin{array}{rl} & \text{a discrete control (the mode of the system)} \\ + & \text{continuous evolution of the variables within a mode} \end{array}$$

The thermostat example





Theorem [HKPV95]

The reachability problem is undecidable in hybrid automata, even for stopwatch automata.

(stopwatch automata: timed automata in which clocks can be stopped)

A relevant question

Is there something between timed automata and hybrid automata which is decidable? $\rightsquigarrow$ See Nicolas' afternoon lecture