Automates temporisés et extensions : frontières de la décidabilité

Patricia Bouyer

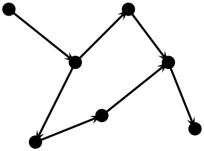
LSV – CNRS & ENS de Cachan

Journées Systèmes Infinis 2005

Model-checking

Does the system satisfy the property?

Modelling

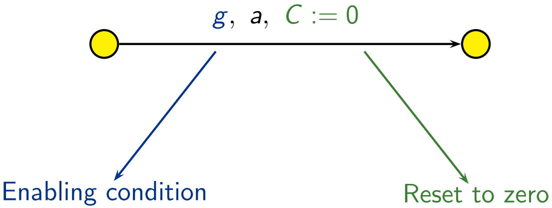


$\models$ φ

Model-checking Algorithm

Timed automata [Alur & Dill 90's]

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



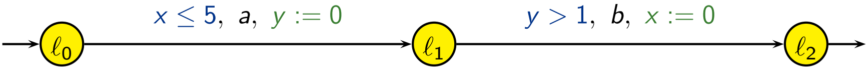
- An enabling condition (or guard) is:

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

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

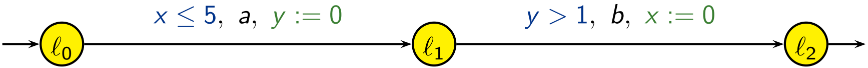
Timed automata (example)

x, y : clocks



Timed automata (example)

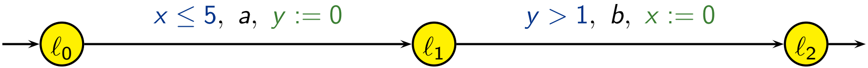
x, y : clocks



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

Timed automata (example)

x, y : clocks

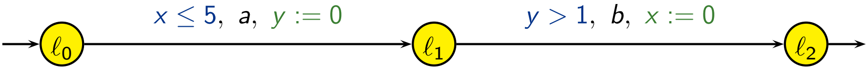


	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

(clock) valuation

Timed automata (example)

x, y : clocks

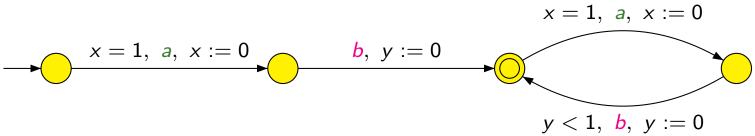


	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

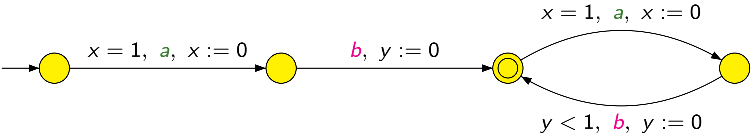
(clock) valuation

➜ timed word $(a, 4.1)(b, 5.5)$

Timed languages

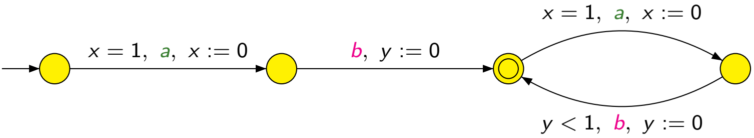


Timed languages



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

Timed languages



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

Verification

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

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

Verification

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

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

Verification

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

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

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

Verification

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

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

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

Theorem **[Alur & Dill 1990's]**

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

Verification

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

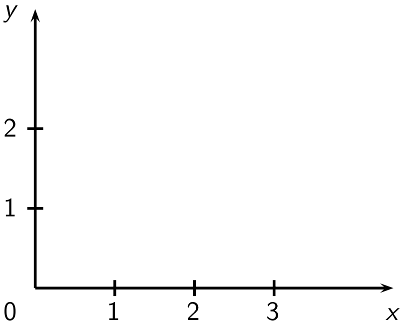
- **Problem:** the set of configurations is infinite
  ➜ classical methods can not be applied
- **Positive key point:** variables (clocks) have the same speed

Theorem [Alur & Dill 1990's]

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

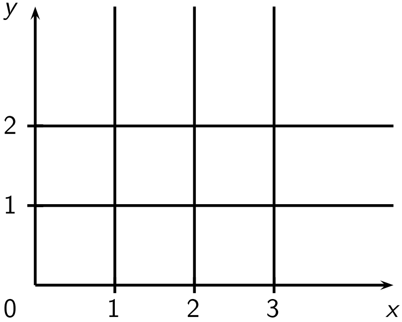
Method: construct a finite abstraction

The region abstraction



Equivalence of finite index

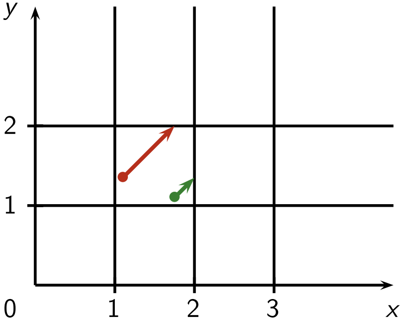
The region abstraction



Equivalence of finite index

- “compatibility” between regions and constraints

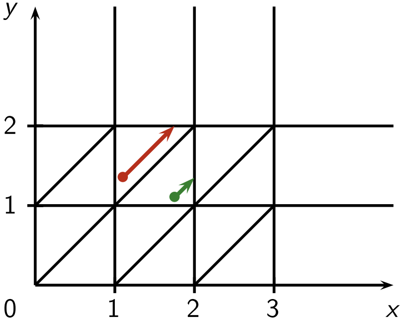
The region abstraction



Equivalence of finite index

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

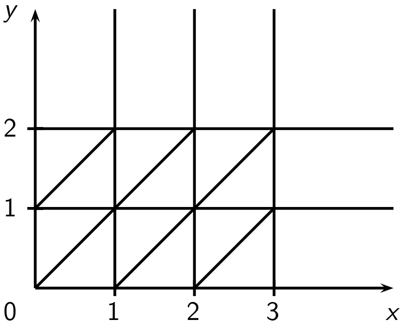
The region abstraction



Equivalence of finite index

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

The region abstraction

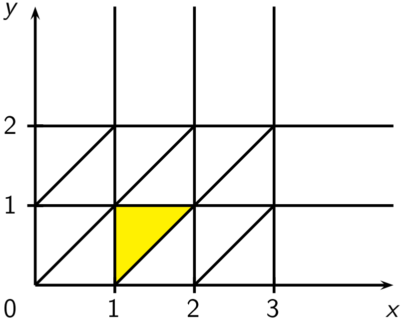


Equivalence of finite index

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

➜ a bisimulation property

The region abstraction



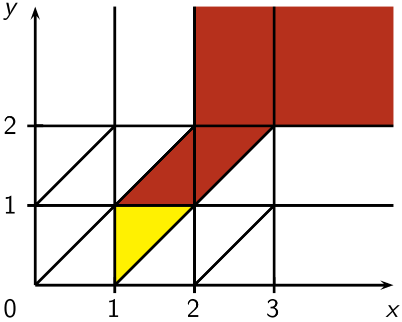
Equivalence of finite index

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

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

➜ a bisimulation property

The region abstraction



Equivalence of finite index

region defined by

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

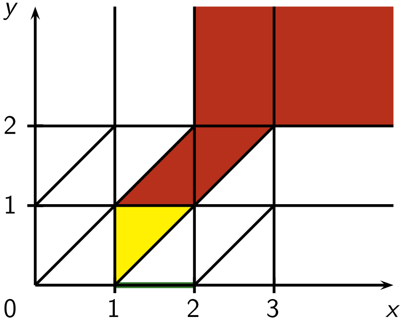
$\{x\} < \{y\}$

delay successors

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

➜ a bisimulation property

The region abstraction



Equivalence of finite index

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

delay successors

successor by reset

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

➜ a bisimulation property

The region automaton

timed automaton $\bigotimes$ region abstraction

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

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

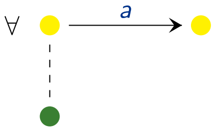
- $R'' \subseteq g$
- $[C \leftarrow 0]R'' \subseteq R'$

➜ **time-abstract bisimulation**

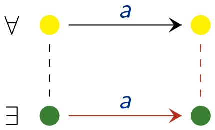
$\mathcal{L}$**(reg. aut.) = UNTIME(**$\mathcal{L}$**(timed aut.))**

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

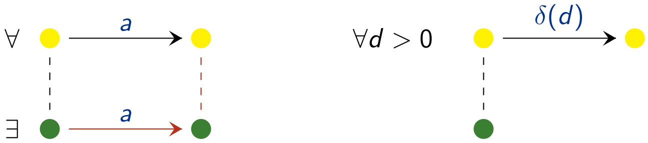
Time-abstract bisimulation



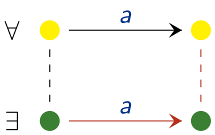
Time-abstract bisimulation

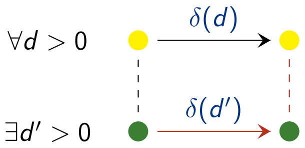


Time-abstract bisimulation

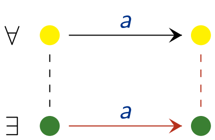


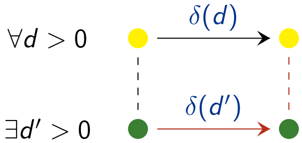
Time-abstract bisimulation





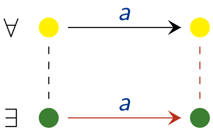
Time-abstract bisimulation

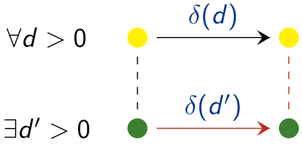




$$(\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \xrightarrow{a_2, t_2} (\ell_2, v_2) \xrightarrow{a_3, t_3} \dots$$

Time-abstract bisimulation

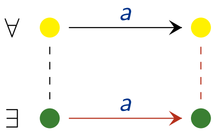


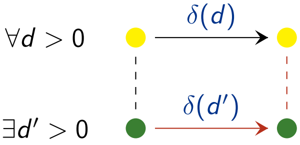


$$
\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \ldots \\
\downarrow & & \downarrow & & \downarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3}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

with $v_i \in R_i$ for all i.

Time-abstract bisimulation



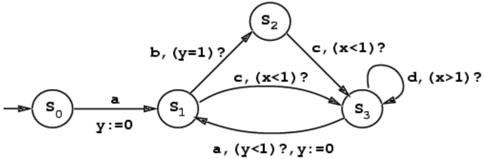


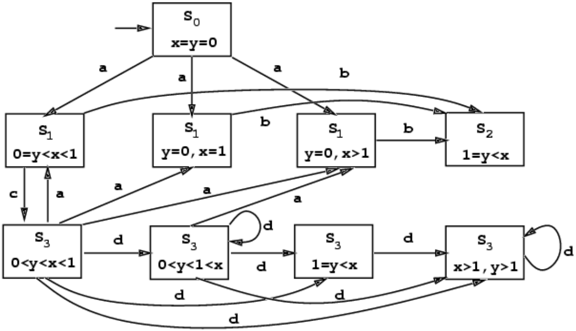
$$
\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \dots \\
\updownarrow & & \updownarrow & & \updownarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3}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

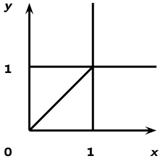
with $v_i \in R_i$ for all i.

An example

[Alur & Dill 1990's]







Consequence of region automata construction

Region automata: correct finite abstraction for checking reachability/Büchi-like properties

Consequence of region automata construction

Region automata: correct finite abstraction for checking reachability/Büchi-like properties

However, everything can not be reduced to finite automata...

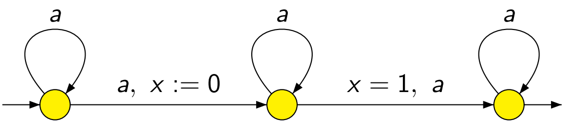
A model not far from undecidability

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

A model not far from undecidability

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

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



Partial conclusion

➜ a timed model interesting for verification purposes

Numerous works have been (and are) devoted to:

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

Partial conclusion

➜ a timed model interesting for verification purposes

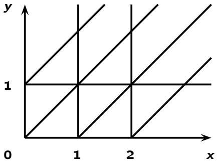
Numerous works have been (and are) devoted to:

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

Role of diagonal constraints

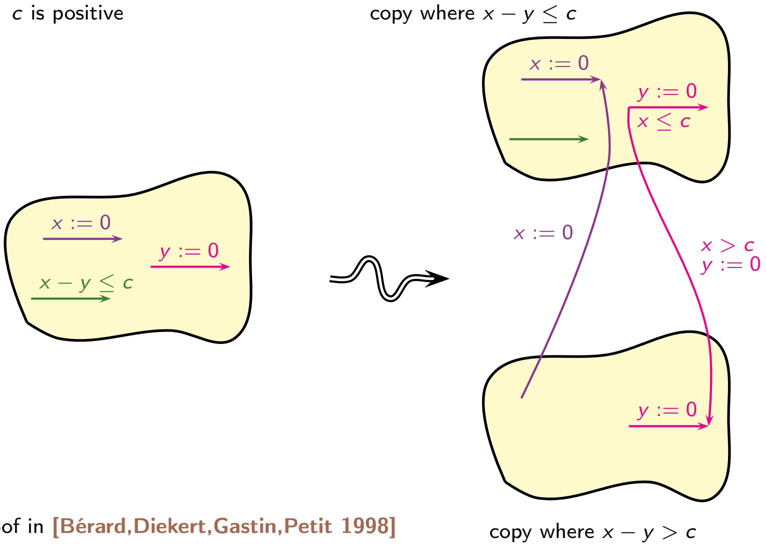
$x - y \sim c$ and $x \sim c$

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



- **Expressiveness:** no additional expressive power

Role of diagonal constraints (cont.)

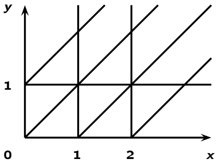


➜ proof in [Bérard,Diekert,Gastin,Petit 1998]

Role of diagonal constraints

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

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

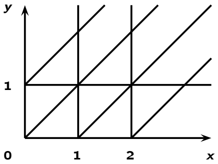


- **Expressiveness:** no additional expressive power...

Role of diagonal constraints

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

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



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

... but there is an exponential blowup which is unavoidable

[Bouyer,Chevalier 2005]

Adding silent actions

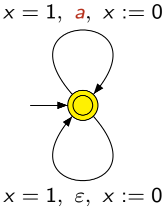
$$\xrightarrow{g,\ \varepsilon,\ C := 0}$$

[Bérard,Diekert,Gastin,Petit 1998]

- **Decidability:** yes

  (actions have no influence on region automaton construction)

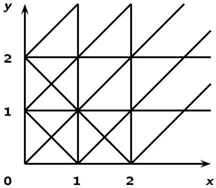
- **Expressiveness:** strictly more expressive!



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

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

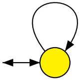
- **Decidability:** - for two clocks, decidable using the abstraction

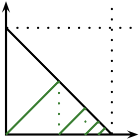


  - for four clocks (or more), undecidable!

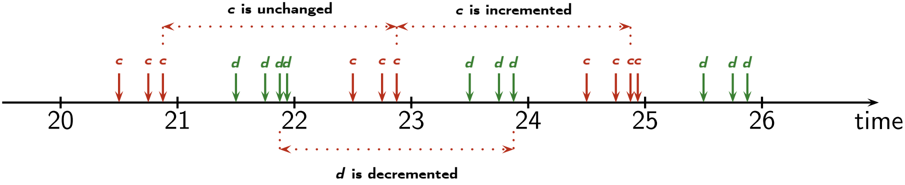
- **Expressiveness:** more expressive! (even using two clocks)

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





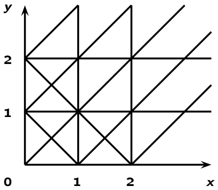
Undecidability sketch of proof



[Bérard,Dufourd 2000]

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

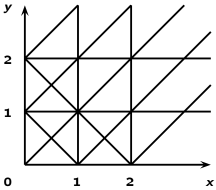
- Two clocks: decidable using the abstraction



- 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...

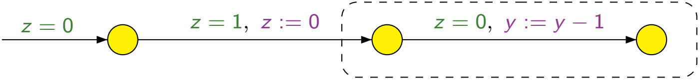
- The general model is undecidable.
  (simulation of a two-counter machine)

Adding new operations on clocks

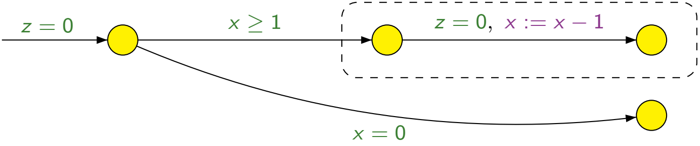
Several types of updates: $x := y + c$, $x :< c$, $x :> c$, etc...

- 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

$$\{x \sim 1, y \sim 1, x - y \sim 1\} + \{x := 0, y := 0, y := 1\}$$

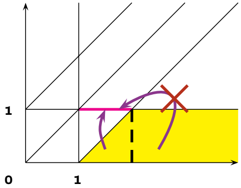


image by $y := 1$

➜ the bisimulation property is not met

The classical region automaton construction is not correct.

Decidability

$$\{x \sim 1, y \sim 1, x - y \sim 1\} + \{x := 0, y := 0, y := 1\}$$

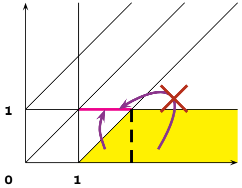
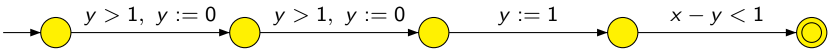


image by $y := 1$

➜ the bisimulation property is not met

The classical region automaton construction is not correct.



Decidability (cont.)

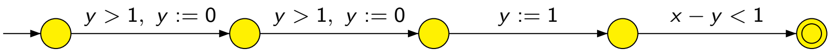
$\mathcal{A}$ $\rightsquigarrow$ Diophantine linear inequations system
$\rightsquigarrow$ is there a solution?
$\rightsquigarrow$ if yes, belongs to a decidable class

Examples:

- constraint $x \sim c$ — $c \leq \max_x$
- 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, $\max_z \geq c + \max_{z,x}$

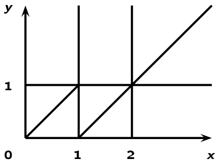
The constants $(\max_x)$ and $(\max_{x,y})$ define a set of regions.

Decidability (cont.)



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

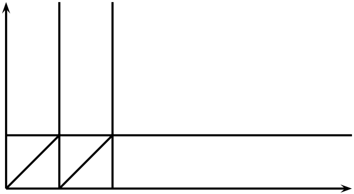
The bisimulation property is met.



What's wrong when undecidable?

Decrementation $x := x - 1$

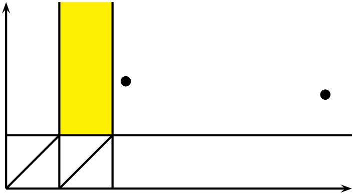
$$\max_x \leq \max_x - 1$$



What's wrong when undecidable?

Decrementation $x := x - 1$

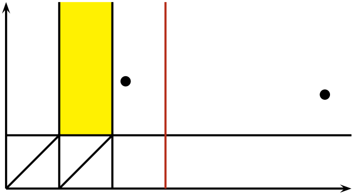
$$\max_x \leq \max_x - 1$$



What's wrong when undecidable?

Decrementation $x := x - 1$

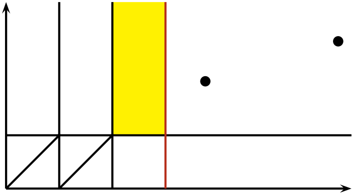
$\max_x \leq \max_x - 1$



What's wrong when undecidable?

Decrementation $x := x - 1$

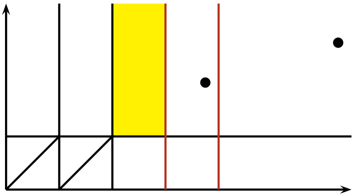
$$\max_x \leq \max_x - 1$$



What's wrong when undecidable?

Decrementation $x := x - 1$

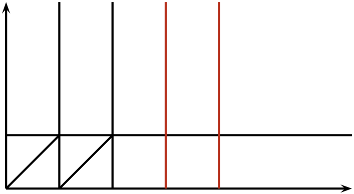
$$\mathsf{max}_x \leq \mathsf{max}_x - 1$$



What's wrong when undecidable?

Decrementation $x := x - 1$

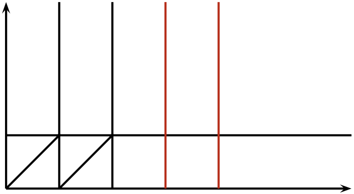
$\max_x \leq \max_x - 1$



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]

Expressiveness and conciseness of UTA

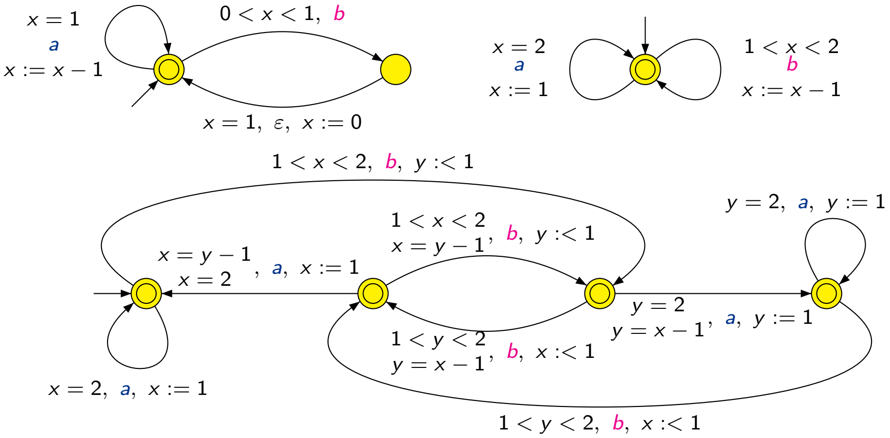
Proposition [Bouyer,Dufourd,Fleury,Petit 2000]

Decidable subclasses of UTA are not more expressive than TA_ε, but are strictly more expressive than TA.

Expressiveness and conciseness of UTA

Proposition [Bouyer,Dufourd,Fleury,Petit 2000]

Decidable subclasses of UTA are not more expressive than TA_ε, but are strictly more expressive than TA.



Expressiveness and conciseness of UTA

Proposition [Bouyer,Dufourd,Fleury,Petit 2000]

Decidable subclasses of UTA are not more expressive than TA_ε, but are strictly more expressive than TA.

Proposition [Bouyer,Chevalier 2005]

Decidable subclasses of UTA are exponentially more concise than TA (resp. TA_ε).

We can implement addition up to 2^n using $\sim n$ clocks:

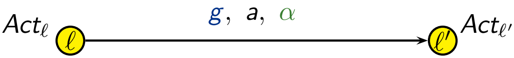
- setting bit b_i to 1: use update $x_i := 1$
- setting bit b_i to 0: use reset $x_i := 0$

and thus express in a concise way languages like

$$L_n = \{(a^{2^n}, \tau) \mid 0 < \tau_1 < \ldots < \tau_{2^n} < 1\}$$

Linear hybrid automata [Henzinger 1996]

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



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

What about decidability?

➜ almost everything is undecidable
[Henzinger,Kopke,Puri,Varaiya 98]

Theorem

The class of LHA with clocks and only one variable having possibly two slopes $k_1 \neq k_2$ is undecidable.

Theorem

The class of *stopwatch* automata is undecidable.

One of the "largest" classes of LHA which are decidable is the class of **initialized rectangular automata** .

Adding alternance...

Alternating timed automata $\equiv$ ATA

[Lasota,Walukiewicz 2005] [Ouaknine,Worrell 2005]

Example

"No two *a*'s are separated by 1 unit of time"

$$\begin{cases} \ell_0, a, true & \mapsto & \ell_0 \wedge (x := 0, \ell_1) \\ \ell_1, a, x \neq 1 & \mapsto & \ell_1 \\ \ell_1, a, x = 1 & \mapsto & \ell_2 \\ \ell_2, a, true & \mapsto & \ell_2 \end{cases}$$

$$\begin{cases} \ell_0 \text{ initial state} \\ \ell_0, \ell_1 \text{ final states} \\ \ell_2 \text{ losing state} \end{cases}$$

Adding alternance...

Alternating timed automata $\equiv$ ATA

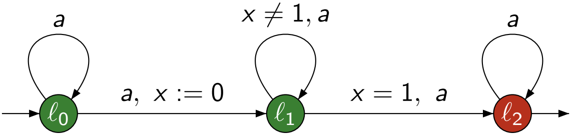
[Lasota,Walukiewicz 2005] [Ouaknine,Worrell 2005]

Example

"No two a's are separated by 1 unit of time"

$$\begin{cases} \ell_0, a, \mathit{true} & \mapsto & \ell_0 \wedge (x := 0, \ell_1) \\ \ell_1, a, x \neq 1 & \mapsto & \ell_1 \\ \ell_1, a, x = 1 & \mapsto & \ell_2 \\ \ell_2, a, \mathit{true} & \mapsto & \ell_2 \end{cases}$$

$$\begin{cases} \ell_0 \text{ initial state} \\ \ell_0, \ell_1 \text{ final states} \\ \ell_2 \text{ losing state} \end{cases}$$



[Lasota,Walukiewicz 2005]

- nice closure properties

[Lasota,Walukiewicz 2005]

- nice closure properties

➜ universality is as difficult as reachability

[Lasota,Walukiewicz 2005]

- nice closure properties

  ➜ universality is as difficult as reachability

- more expressive than timed automata

[Lasota,Walukiewicz 2005]

- nice closure properties

➜ universality is as difficult as reachability

- more expressive than timed automata

Theorem

- ATA are undecidable.
- One clock ATA are decidable, with a non-primitive recursive lower bound.
- One clock ATA with ε-transitions are undecidable.

[Lasota,Walukiewicz 2005]

- nice closure properties

➜ universality is as difficult as reachability

- more expressive than timed automata

Theorem

- ATA are undecidable.
- One clock ATA are decidable, with a non-primitive recursive lower bound.
- One clock ATA with ε-transitions are undecidable.

Lower bound: simulation of a lossy channel system... [Schnoebelen 2002]

Why is that difficult?
A configuration: a finite set of pairs (ℓ, x)

$$(\ell, 0) \quad (\ell, 0.3) \quad (\ell, 1.2) \quad (\ell, 2.3) \quad (\ell', 0.4) \quad (\ell', 1) \quad (\ell', 0.8)$$

Why is that difficult?

A configuration: a finite set of pairs (ℓ, x)

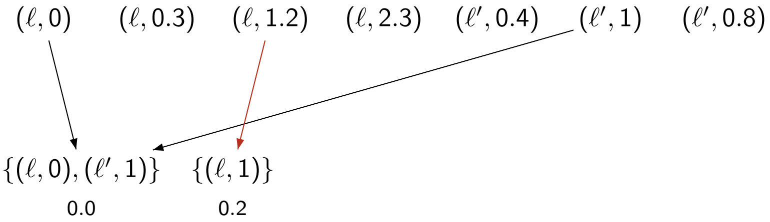
$(\ell, 0)$ $(\ell, 0.3)$ $(\ell, 1.2)$ $(\ell, 2.3)$ $(\ell', 0.4)$ $(\ell', 1)$ $(\ell', 0.8)$

$\{(\ell, 0), (\ell', 1)\}$

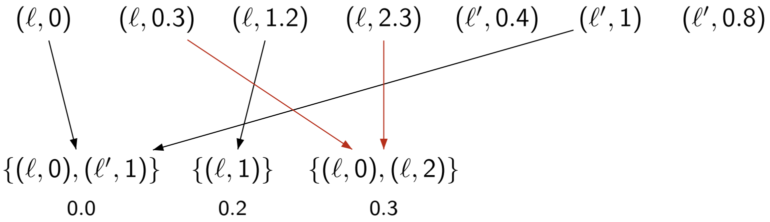
0.0

Why is that difficult?

A configuration: a finite set of pairs (ℓ, x)

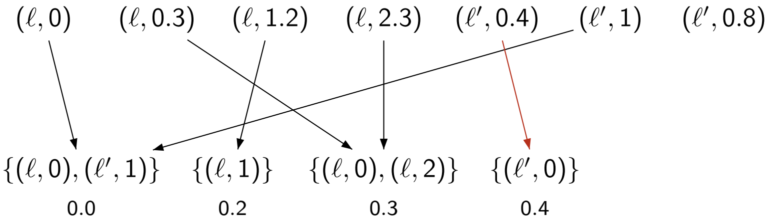


Why is that difficult?
A configuration: a finite set of pairs (ℓ, x)



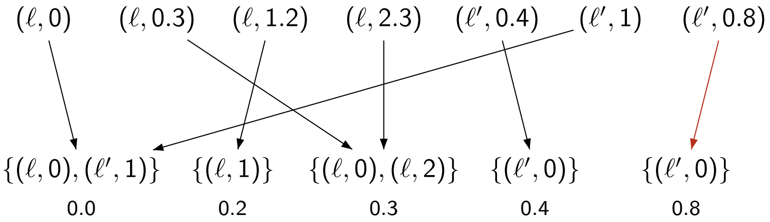
Why is that difficult?

A configuration: a finite set of pairs (ℓ, x)



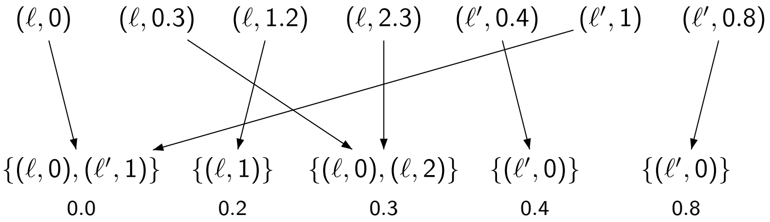
Why is that difficult?

A configuration: a finite set of pairs (ℓ, x)



Why is that difficult?

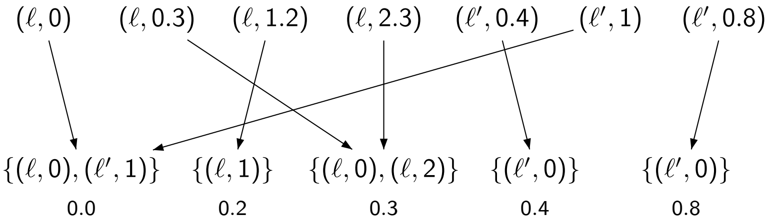
A configuration: a finite set of pairs (ℓ, x)



☹ possibly infinitely many such abstractions

Why is that difficult?

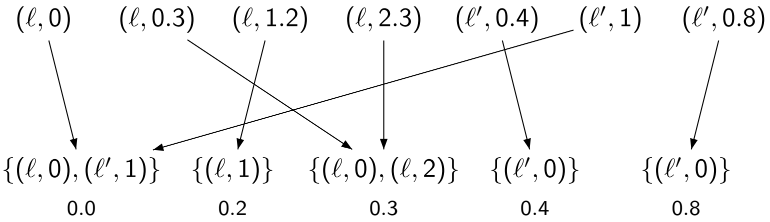
A configuration: a finite set of pairs (ℓ, x)



- ☹ possibly infinitely many such abstractions
- ☺ there is a well-quasi ordering on these abstractions!

Why is that difficult?

A configuration: a finite set of pairs (ℓ, x)



- ☹ possibly infinitely many such abstractions
- ☺ there is a well-quasi ordering on these abstractions!

Reachability is decidable!

Conclusion

- A decidable model not far from undecidability
- Recently, much works on one-clock timed automata
  - universality (over finite words is decidable) **[Ouaknine,Worrell 2004]**
  - reachability is NLOGSPACE-complete **[Laroussinie,Markey,Schnoebelen 2004]**
  - reachability of one-clock alternating timed automata is decidable **[Lasota,Walukiewicz 2005]**
- Some current research directions:
  - controller synthesis
  - implementability issues (program synthesis)
  - optimal computations
  - ...

Bibliography I

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

[AM04] Alur, Madhusudan. **Decision Problems for Timed Automata.** SFM-04:RT (LNCS 3142).

[Asa04] Asarin. **Challenges in Timed Languages: From Applied Theory to Basic Theory.** BEATCS 83, 2004.

[BC05] Bouyer, Chevalier. **On the Conciseness of Extensions of Timed Automata.** Submitted, 2005.

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

[BDFP04] Bouyer, Dufourd, Fleury, Petit. **Updatable Timed Automata.** TCS 321(2–3), 2004.

[BDGP98] Bérard, Diekert, Gastin, Petit. **Characterization of the Expressive Power of Silent Transitions in Timed Automata.** Fund. Inf. 36(2–3), 1998.

Bibliography II

[HKPV98] Henzinger, Kopke, Puri, Varaiya. **What's Decidable about Hybrid Automata?** J. Comp. and Sys. Sci 57, 1998.

[LW05] Lasota, Walukiewicz. **Alternating Timed Automata.** FOSSACS'05. To appear.

[LMS04] Laroussinie, Markey, Schnoebelen. **Model Checking Timed Automata with One or Two Clocks.** CONCUR'04 (LNCS 3170).

[OW05] Ouaknine, Worrell. **On the Decidability of Metric Temporal Logic.** Submitted, 2005

[OW04] Ouaknine, Worrell. **On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap.** LICS'04.

[Sch02] Schnoebelen. **Verifying Lossy Channel Systems has Nonprimitive Recursive Complexity.** IPL 83(5), 2002.