An Introduction to Timed Systems

Patricia Bouyer

LSV, ENS Cachan & CNRS, France

Outline

Time!

Context: verification of embedded critical systems

Time

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

➜ Need of models and specification languages integrating timing aspects

Adding timing informations

- **Untimed case:** sequence of observable events
  a: send message b: receive message

$$a\ b\ a\ b\ a\ b\ a\ b\ a\ b\ \cdots = (a\ b)^{\omega}$$

Adding timing informations

- **Untimed case:** sequence of observable events
  a: send message b: receive message

$$a\ b\ a\ b\ a\ b\ a\ b\ a\ b\ \cdots = (a\ b)^\omega$$

- **Timed case:** sequence of **dated** observable events

$$(a, d_1)\ (b, d_2)\ (a, d_3)\ (b, d_4)\ (a, d_5)\ (b, d_6)\ \cdots$$

d_1: date at which the first a occurs
d_2: date at which the first b occurs, ...

Adding timing informations

- **Untimed case:** sequence of observable events
  a: send message b: receive message

$$a\ b\ a\ b\ a\ b\ a\ b\ a\ b\ \cdots = (a\ b)^\omega$$

- **Timed case:** sequence of **dated** observable events

$$(a, d_1)\ (b, d_2)\ (a, d_3)\ (b, d_4)\ (a, d_5)\ (b, d_6)\ \cdots$$

  d_1: date at which the first a occurs
  d_2: date at which the first b occurs, ...
  - **Discrete-time semantics:** dates are *e.g.* taken in $\mathbb{N}$
    Ex: $(a, 1)(b, 3)(c, 4)(a, 6)$

Adding timing informations

- **Untimed case:** sequence of observable events
  a: send message b: receive message

$$a\ b\ a\ b\ a\ b\ a\ b\ a\ b\ \cdots = (a\ b)^\omega$$

- **Timed case:** sequence of **dated** observable events

$$(a, d_1)\ (b, d_2)\ (a, d_3)\ (b, d_4)\ (a, d_5)\ (b, d_6)\ \cdots$$

  d_1: date at which the first a occurs
  d_2: date at which the first b occurs, ...
  - **Discrete-time semantics:** dates are *e.g.* taken in $\mathbb{N}$
    Ex: $(a, 1)(b, 3)(c, 4)(a, 6)$
  - **Dense-time semantics:** dates are *e.g.* taken in $\mathbb{Q}_+$, or in $\mathbb{R}_+$
    Ex: $(a, 1.28).(b, 3.1).(c, 3.98)(a, 6.13)$

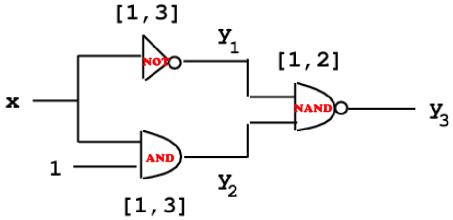
A case for dense-time

Time domain: discrete (*e.g.* $\mathbb{N}$) or dense (*e.g.* $\mathbb{Q}_+$ or $\mathbb{R}_+$)

- A compositionality problem with discrete time
- Dense-time is a more general model than discrete time
- But, can we not always discretize?

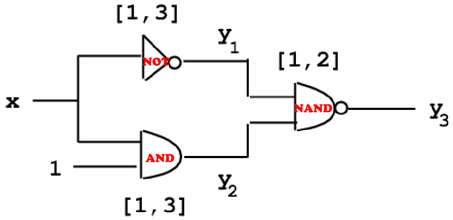
A digital circuit [Alur 91]

Discussion in the context of reachability problems for asynchronous digital circuits [Brzozowski, Seger 1991]



A digital circuit [Alur 91]

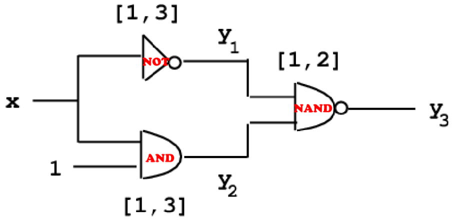
Discussion in the context of reachability problems for asynchronous digital circuits [Brzozowski, Seger 1991]



Start with x=0 and y=[101] (stable configuration)

A digital circuit [Alur 91]

Discussion in the context of reachability problems for asynchronous digital circuits [Brzozowski, Seger 1991]

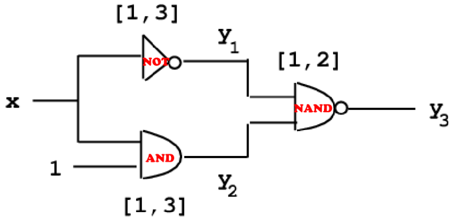


Start with x=0 and y=[101] (stable configuration)

The input x changes to 1. The corresponding stable state is y=[011]

A digital circuit [Alur 91]

Discussion in the context of reachability problems for asynchronous digital circuits [Brzozowski, Seger 1991]



Start with x=0 and y=[101] (stable configuration)

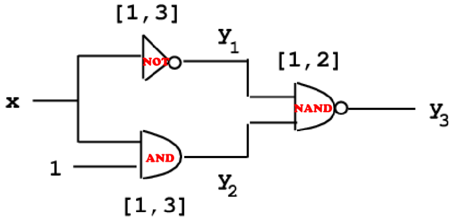
The input x changes to 1. The corresponding stable state is y=[011]

However, many possible behaviours, e.g.

$$[101] \xrightarrow[1.2]{y_2} [111] \xrightarrow[2.5]{y_3} [110] \xrightarrow[2.8]{y_1} [010] \xrightarrow[4.5]{y_3} [011]$$

A digital circuit [Alur 91]

Discussion in the context of reachability problems for asynchronous digital circuits [Brzozowski, Seger 1991]



Start with x=0 and y=[101] (stable configuration)

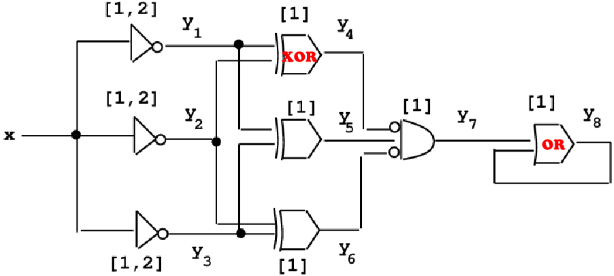
The input x changes to 1. The corresponding stable state is y=[011]

However, many possible behaviours, e.g.

$$[101] \xrightarrow[1.2]{y_2} [111] \xrightarrow[2.5]{y_3} [110] \xrightarrow[2.8]{y_1} [010] \xrightarrow[4.5]{y_3} [011]$$

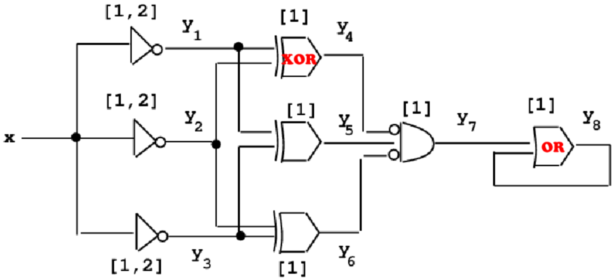
Reachable configurations: {[101], [111], [110], [010], [011], [001]}

Is discretizing sufficient? An example [Alur 91]



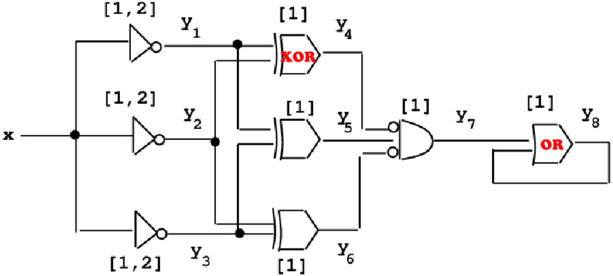
- This digital circuit **is not** 1-discretizable.

Is discretizing sufficient? An example **[Alur 91]**



- This digital circuit **is not** 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, x is set to 1)

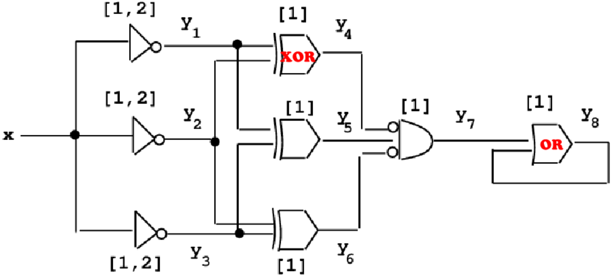
Is discretizing sufficient? An example **[Alur 91]**



- This digital circuit **is not** 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, x is set to 1)

$$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$$

Is discretizing sufficient? An example [Alur 91]

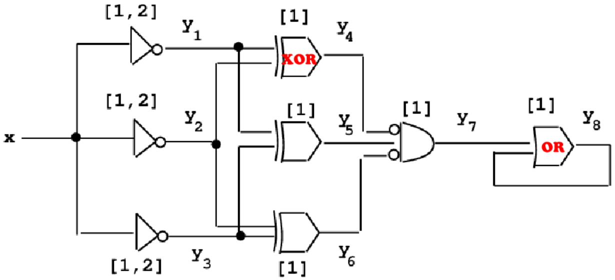


- This digital circuit **is not** 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, x is set to 1)

$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$

$[11100000] \xrightarrow[1]{y_1, y_2, y_3} [00000000]$

Is discretizing sufficient? An example **[Alur 91]**



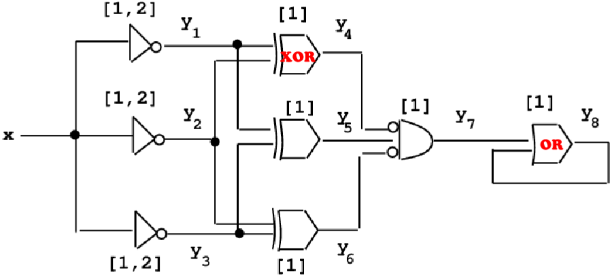
- This digital circuit **is not** 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, x is set to 1)

$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$

$[11100000] \xrightarrow[1]{y_1, y_2, y_3} [00000000]$

$[11100000] \xrightarrow[1]{y_1} [01111000] \xrightarrow[2]{y_2, y_3, y_4, y_5} [00000000]$

Is discretizing sufficient? An example [Alur 91]



- This digital circuit **is not** 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, x is set to 1)

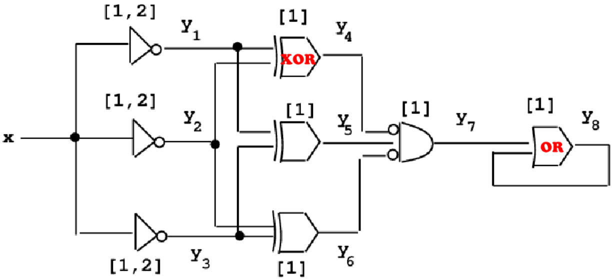
$$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$$

$$[11100000] \xrightarrow[1]{y_1, y_2, y_3} [00000000]$$

$$[11100000] \xrightarrow[1]{y_1} [01111000] \xrightarrow[2]{y_2, y_3, y_4, y_5} [00000000]$$

$$[11100000] \xrightarrow[1]{y_1, y_2} [00100000] \xrightarrow[2]{y_3, y_5, y_6} [00001100] \xrightarrow[3]{y_5, y_6} [00000000]$$

Is discretizing sufficient? An example **[Alur 91]**



- This digital circuit **is not** 1-discretizable.
- Why that? (initially $x = 0$ and $y = [11100000]$, x is set to 1)

$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} \boxed{[00000001]}$

$[11100000] \xrightarrow[1]{y_1, y_2, y_3} \boxed{[00000000]}$

$[11100000] \xrightarrow[1]{y_1} [01111000] \xrightarrow[2]{y_2, y_3, y_4, y_5} \boxed{[00000000]}$

$[11100000] \xrightarrow[1]{y_1, y_2} [00100000] \xrightarrow[2]{y_3, y_5, y_6} [00001100] \xrightarrow[3]{y_5, y_6} \boxed{[00000000]}$

Is discretizing sufficient?

Theorem [Brzozowski Seger 1991]

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

Is discretizing sufficient?

Theorem [Brzozowski Seger 1991]

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

Claim

Finding a correct granularity is as difficult as computing the set of reachable states in dense-time.

Is discretizing sufficient?

Theorem [Brzozowski Seger 1991]

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

Claim

Finding a correct granularity is as difficult as computing the set of reachable states in dense-time.

Further counter-example

There exist systems for which no granularity exists. (see later)

Is discretizing sufficient?

Theorem [Brzozowski Seger 1991]

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

Claim

Finding a correct granularity is as difficult as computing the set of reachable states in dense-time.

Further counter-example

There exist systems for which no granularity exists. (see later)

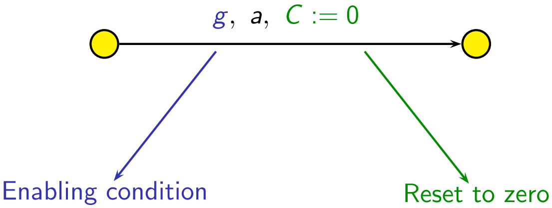
Hence, we better consider a dense-time domain!

Outline

Timed automata

[Alur,Dill 1990]

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



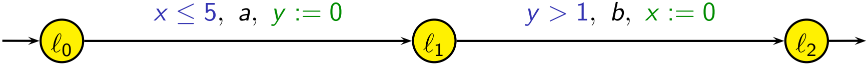
- An enabling condition (or guard) is:

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

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

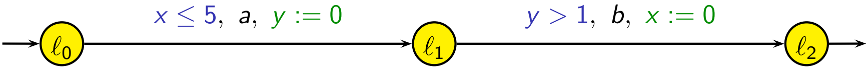
Timed automata (example)

x, y : clocks



Timed automata (example)

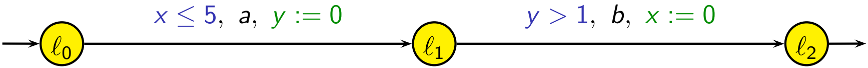
x, y : clocks



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

Timed automata (example)

x, y : clocks

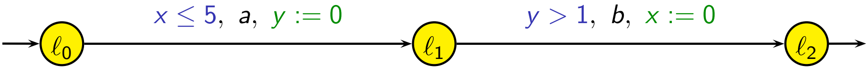


$$
\begin{array}{cccccccccc}
 & \ell_0 & \xrightarrow{\delta(4.1)} & \ell_0 & \xrightarrow{a} & \ell_1 & \xrightarrow{\delta(1.4)} & \ell_1 & \xrightarrow{b} & \ell_2 \\
x & 0 & & 4.1 & & 4.1 & & 5.5 & & 0 \\
y & 0 & & 4.1 & & 0 & & 1.4 & & 1.4
\end{array}
$$

(clock) valuation

Timed automata (example)

x, y : clocks



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

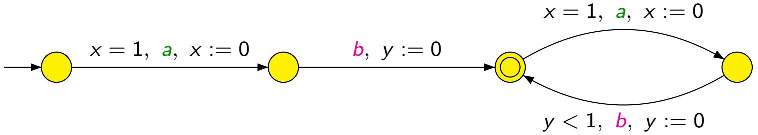
(clock) valuation

➜ timed word $(a, 4.1)(b, 5.5)$

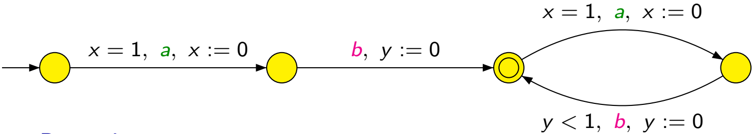
Timed automata semantics

- $\mathcal{A} = (\Sigma, L, X, \longrightarrow)$ is a TA
- **Configurations:** $(\ell, v) \in L \times T^X$ where T is the time domain
- **Timed Transition System:**
  - **action transition**: $(\ell, v) \xrightarrow{a} (\ell', v')$ if $\exists \ell \xrightarrow{g,a,r} \ell' \in \mathcal{A}$ s.t. $\begin{cases} v \models g \\ v' = v[r \leftarrow 0] \end{cases}$
  - **delay transition**: $(\ell, v) \xrightarrow{\delta(d)} (\ell, v + d)$ if $d \in T$

Discrete *vs* dense-time semantics

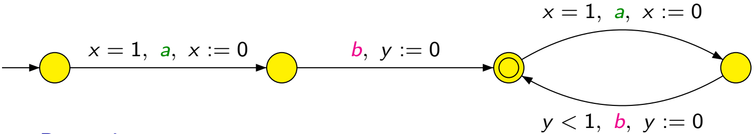


Discrete *vs* dense-time semantics



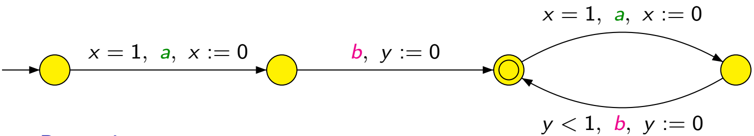
- Dense-time:
  $L_{dense} = \{((ab)^\omega, \tau) \mid \forall i,\ \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1}\}$

Discrete *vs* dense-time semantics

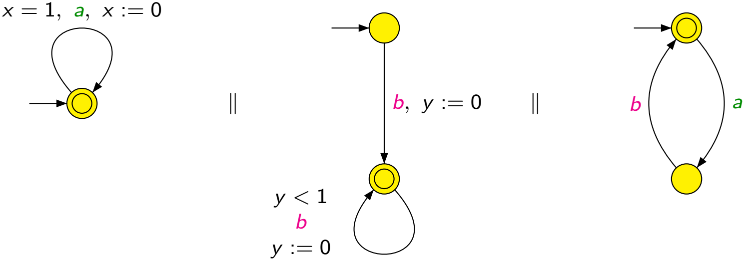


- Dense-time:
  $L_{dense} = \{((ab)^\omega, \tau) \mid \forall i,\ \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1}\}$
- Discrete-time: $L_{discrete} = \emptyset$

Discrete *vs* dense-time semantics



- 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: $L_{discrete} = \emptyset$

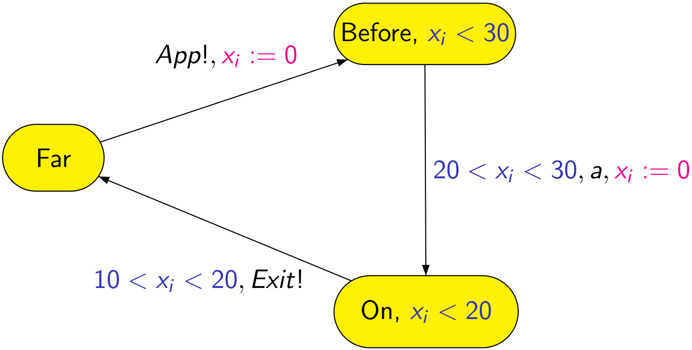


Classical verification problems

- reachability of a control state
- $\mathcal{S} \sim \mathcal{S}'$: bisimulation, etc...
- $L(\mathcal{S}) \subseteq L(\mathcal{S}')$: language inclusion
- $\mathcal{S} \models \varphi$ for some formula φ: model-checking
- $\mathcal{S} \parallel A_T$ + reachability: testing automata
- ...

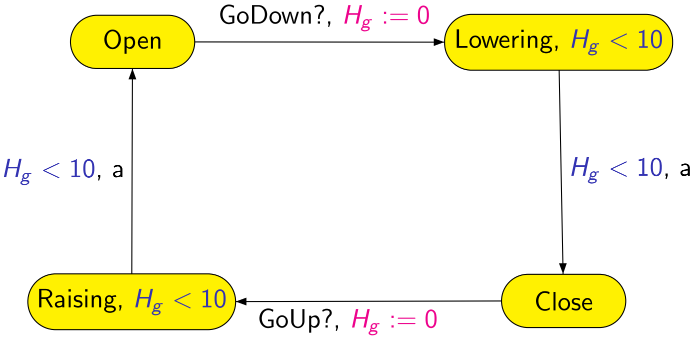
The train crossing example (1)

Train$_i$ **with** $i = 1, 2, \ldots$



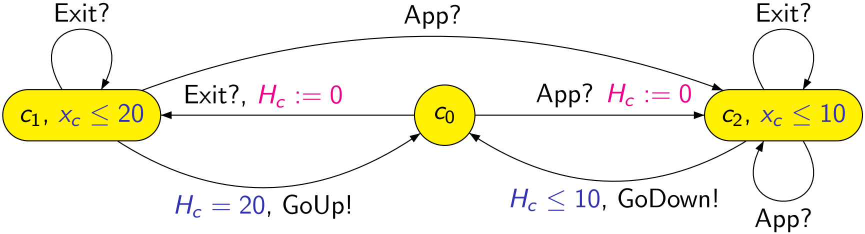
The train crossing example (2)

The gate:



The train crossing example (3)

The controller:



The train crossing example (4)

We use the synchronization function f:

Train_1	Train_2	Gate	Controller	
$App!$	.	.	$App?$	App
.	$App!$	.	$App?$	App
$Exit!$	.	.	$Exit?$	$Exit$
.	$Exit!$	.	$Exit?$	$Exit$
a	.	.	.	a
.	a	.	.	a
.	.	a	.	a
.	.	$GoUp?$	$GoUp!$	$GoUp$
.	.	$GoDown?$	$GoDown!$	$GoDown$

to define the parallel composition $(\mathsf{Train}_1 \parallel \mathsf{Train}_2 \parallel \mathsf{Gate} \parallel \mathsf{Controller})$

NB: the parallel composition does not add expressive power!

The train crossing example (5)

Some properties one could check:

- Is the gate closed when a train crosses the road?

The train crossing example (5)

Some properties one could check:

- Is the gate closed when a train crosses the road?

$$\mathbf{AG}(\text{train.On} \Rightarrow \text{gate.Close})$$

The train crossing example (5)

Some properties one could check:

- Is the gate closed when a train crosses the road?

$$\mathbf{AG}(\text{train.On} \Rightarrow \text{gate.Close})$$

- Is the gate always closed for less than 5 minutes?

The train crossing example (5)

Some properties one could check:

- Is the gate closed when a train crosses the road?

$$\mathbf{AG}(\text{train.On} \Rightarrow \text{gate.Close})$$

- Is the gate always closed for less than 5 minutes?

$$\neg\mathbf{EF}(\text{gate.Close} \wedge \mathbf{E}(\text{gate.Close}\ \mathbf{U}_{>5\ \text{min}}\ \neg\text{gate.Close}))$$

Outline

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 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
  ➜ 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
  ➜ classical methods for finite-state systems cannot be applied

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

Theorem [Alur,Dill 1990]

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

Verification

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

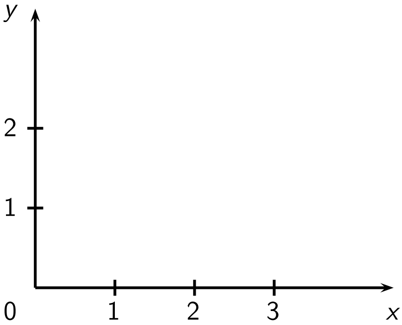
- **Problem:** the set of configurations is infinite
  ➜ classical methods for finite-state systems cannot be applied
- **Positive key point:** variables (clocks) increase at the same speed

Theorem [Alur,Dill 1990]

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

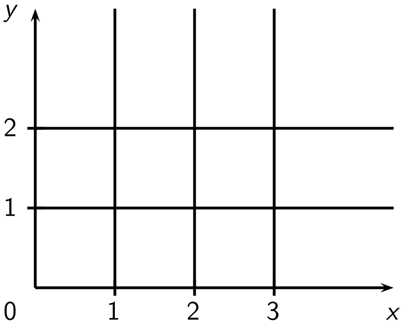
Method: construct a finite abstraction

The region abstraction



Equivalence of finite index

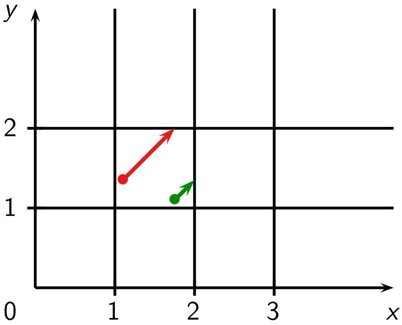
The region abstraction



Equivalence of finite index

- "compatibility" between regions and constraints

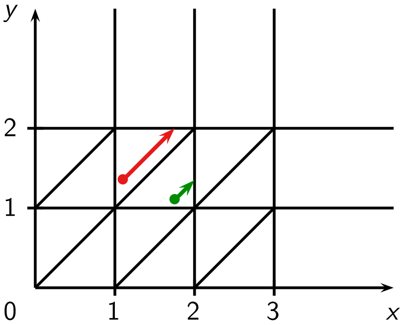
The region abstraction



Equivalence of finite index

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

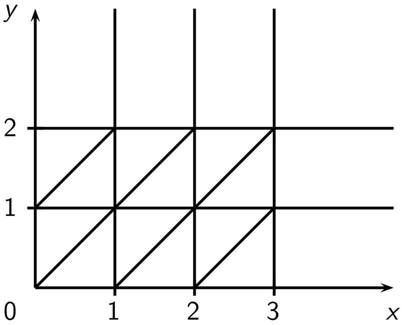
The region abstraction



Equivalence of finite index

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

The region abstraction

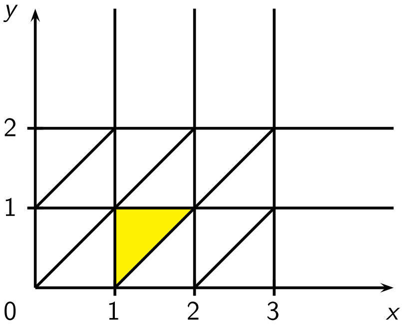


Equivalence of finite index

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

➜ a time-abstract bisimulation property

The region abstraction



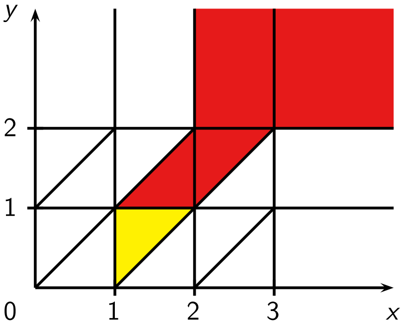
Equivalence of finite index

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 time-abstract bisimulation property

The region abstraction



Equivalence of finite index

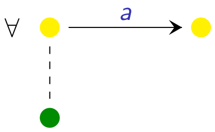
region defined by
$I_x =]1;2[$, $I_y =]0;1[$
$\{x\} < \{y\}$

successor regions

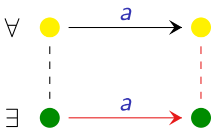
- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

➜ a time-abstract bisimulation property

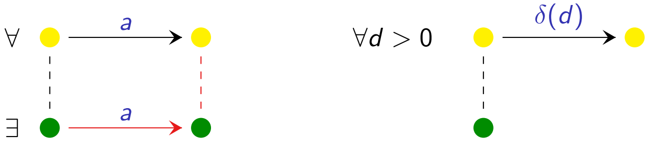
Time-abstract bisimulation



Time-abstract bisimulation



Time-abstract bisimulation

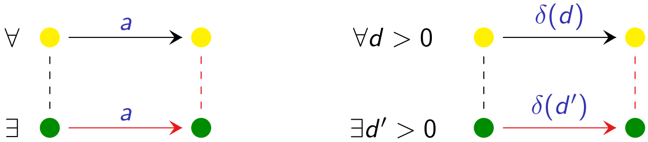


Time-abstract bisimulation

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

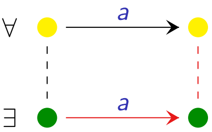
$$\begin{tikzcd} \forall d > 0 & \bullet \arrow[r, "\delta(d)"] \arrow[d, dash, dashed] & \bullet \arrow[d, dash, dashed] \\ \exists d' > 0 & \bullet \arrow[r, "\delta(d')"] & \bullet \end{tikzcd}$$

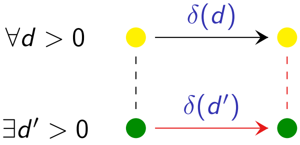
Time-abstract bisimulation



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

Time-abstract bisimulation

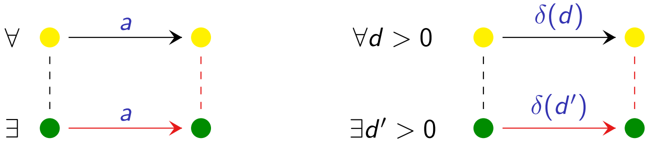




$$\begin{tikzcd}
(\ell_0, v_0) \arrow[r, "a_1, t_1"] \arrow[d] & (\ell_1, v_1) \arrow[r, "a_2, t_2"] \arrow[d] & (\ell_2, v_2) \arrow[r, "a_3, t_3"] \arrow[d] & \ldots \\
(\ell_0, R_0) \arrow[r, "a_1"] & (\ell_1, R_1) \arrow[r, "a_2"] & (\ell_2, R_2) \arrow[r, "a_3"] & \ldots
\end{tikzcd}$$

with $v_i \in R_i$ for all i.

Time-abstract bisimulation



$$
\begin{tikzcd}
(\ell_0, v_0) \arrow[r, "a_1, t_1"] \arrow[d, leftrightarrow] & (\ell_1, v_1) \arrow[r, "a_2, t_2"] \arrow[d, leftrightarrow] & (\ell_2, v_2) \arrow[r, "a_3, t_3"] \arrow[d, leftrightarrow] & \dots \\
(\ell_0, R_0) \arrow[r, "a_1"] & (\ell_1, R_1) \arrow[r, "a_2"] & (\ell_2, R_2) \arrow[r, "a_3"] & \dots
\end{tikzcd}
$$

with $v_i \in R_i$ for all i.

Region automaton $\equiv$ finite bisimulation quotient

timed automaton $\bigotimes$ region abstraction

Region automaton $\equiv$ finite bisimulation quotient

timed automaton $\bigotimes$ region abstraction

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

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

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

Region automaton $\equiv$ finite bisimulation quotient

timed automaton $\bigotimes$ region abstraction

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

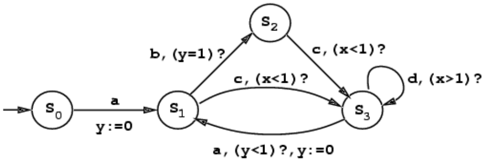
$(\ell, R) \xrightarrow{a} (\ell', R')$ if there exists $R'' \in \mathrm{Succ}_t^*(R)$ s.t.

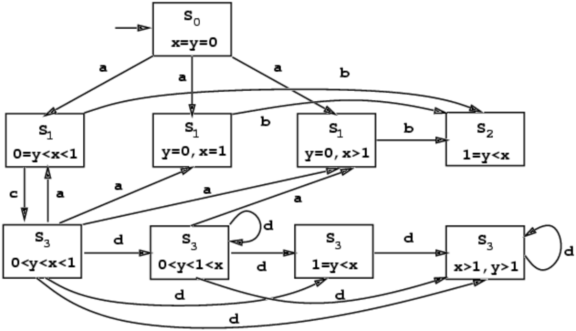
- $R'' \subseteq g$
- $[C \leftarrow 0]R'' \subseteq R'$

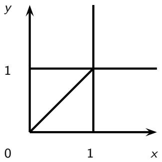
$\mathcal{L}$**(reg. aut.) = UNTIME(**$\mathcal{L}$**(timed aut.))**

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

An example [AD 90's]







Consequence of region automata construction

Region automata:

correct finite (and exponential) abstraction for checking reachability/Büchi-like properties.

Consequence of region automata construction

Region automata:

correct finite (and exponential) abstraction for checking reachability/Büchi-like properties.

However...

everything can not be reduced to finite automata...

A model not far from undecidability

Some bad news...

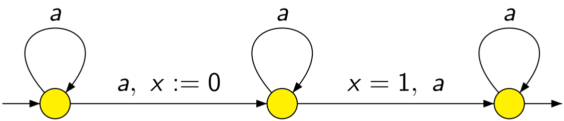
- Language universality is undecidable **[Alur,Dill 1990]**
- Language inclusion is undecidable **[Alur,Dill 1990]**
- Complementability is undecidable **[Tripakis 2003, Finkel 2006]**
- ...

A model not far from undecidability

Some bad news...

- Language universality is undecidable [Alur,Dill 1990]
- Language inclusion is undecidable [Alur,Dill 1990]
- Complementability is undecidable [Tripakis 2003, Finkel 2006]
- ...

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

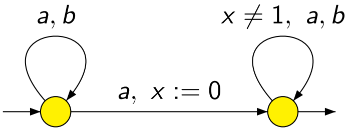


A model not far from undecidability

Some bad news...

- Language universality is undecidable [Alur,Dill 1990]
- Language inclusion is undecidable [Alur,Dill 1990]
- Complementability is undecidable [Tripakis 2003, Finkel 2006]
- ...

An example of non-determinizable/non-complementable timed aut.: [Alur,Madhusudan 2004]



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!)**

Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many "timed" or "hybrid" systems:
  - various extensions of timed automata
    [Bérard,Diekert,Gastin,Petit 1998] [Choffrut,Goldwurm 2000] [Bouyer,Dufourd,Fleury,Petit 2004] $\cdots$

Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many "timed" or "hybrid" systems:
  - various extensions of timed automata
    [Bérard,Diekert,Gastin,Petit 1998] [Choffrut,Goldwurm 2000] [Bouyer,Dufourd,Fleury,Petit 2004] $\cdots$
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    [Alur,Courcoubetis,Dill 1993] [Laroussinie,Larsen,Weise 1995]

Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many "timed" or "hybrid" systems:
  - various extensions of timed automata
    [Bérard,Diekert,Gastin,Petit 1998] [Choffrut,Goldwurm 2000] [Bouyer,Dufourd,Fleury,Petit 2004] $\cdots$
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    [Alur,Courcoubetis,Dill 1993] [Laroussinie,Larsen,Weise 1995]
  - weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
    [Bouyer,Larsen,Markey,Rasmussen 2006] [Bouyer,Larsen,Markey 2007]

Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many "timed" or "hybrid" systems:
  - various extensions of timed automata
    [Bérard,Diekert,Gastin,Petit 1998] [Choffrut,Goldwurm 2000] [Bouyer,Dufourd,Fleury,Petit 2004] $\cdots$
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    [Alur,Courcoubetis,Dill 1993] [Laroussinie,Larsen,Weise 1995]
  - weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
    [Bouyer,Larsen,Markey,Rasmussen 2006] [Bouyer,Larsen,Markey 2007]
  - o-minimal hybrid systems
    [Lafferriere,Pappas,Sastry 2000] [Brihaye 2005]

Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many "timed" or "hybrid" systems:
  - various extensions of timed automata
    [Bérard,Diekert,Gastin,Petit 1998] [Choffrut,Goldwurm 2000] [Bouyer,Dufourd,Fleury,Petit 2004] ⋯
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    [Alur,Courcoubetis,Dill 1993] [Laroussinie,Larsen,Weise 1995]
  - weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
    [Bouyer,Larsen,Markey,Rasmussen 2006] [Bouyer,Larsen,Markey 2007]
  - o-minimal hybrid systems
    [Lafferriere,Pappas,Sastry 2000] [Brihaye 2005]
  - ⋯

Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many "timed" or "hybrid" systems:
  - various extensions of timed automata
    [Bérard,Diekert,Gastin,Petit 1998] [Choffrut,Goldwurm 2000] [Bouyer,Dufourd,Fleury,Petit 2004] $\cdots$
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    [Alur,Courcoubetis,Dill 1993] [Laroussinie,Larsen,Weise 1995]
  - weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
    [Bouyer,Larsen,Markey,Rasmussen 2006] [Bouyer,Larsen,Markey 2007]
  - o-minimal hybrid systems
    [Lafferriere,Pappas,Sastry 2000] [Brihaye 2005]
  - $\cdots$
- Note however that it might be hard to prove there is a finite bisimulation quotient!

Partial conclusion

- This idea of a finite bisimulation quotient has been applied to many "timed" or "hybrid" systems:
  - various extensions of timed automata
    [Bérard,Diekert,Gastin,Petit 1998] [Choffrut,Goldwurm 2000] [Bouyer,Dufourd,Fleury,Petit 2004] · · ·
  - model-checking of branching-time properties (TCTL, timed μ-calculus)
    [Alur,Courcoubetis,Dill 1993] [Laroussinie,Larsen,Weise 1995]
  - weighted/priced timed automata (e.g. WCTL model-checking, optimal games)
    [Bouyer,Larsen,Markey,Rasmussen 2006] [Bouyer,Larsen,Markey 2007]
  - o-minimal hybrid systems
    [Lafferriere,Pappas,Sastry 2000] [Brihaye 2005]
  - · · ·
- Note however that it might be hard to prove there is a finite bisimulation quotient!
- Note that in practice, the region automaton is not constructed, and symbolic technics based on *zones* are used

Outline

The example of alternating timed automata

Alternating timed automata $\equiv$ ATA

[Lasota,Walukiewicz 2005,2007] [Ouaknine,Worrell 2005,2007]

The example of alternating timed automata

Alternating timed automata $\equiv$ ATA

[Lasota,Walukiewicz 2005,2007] [Ouaknine,Worrell 2005,2007]

Example

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

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

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

The example of alternating timed automata

Alternating timed automata $\equiv$ ATA

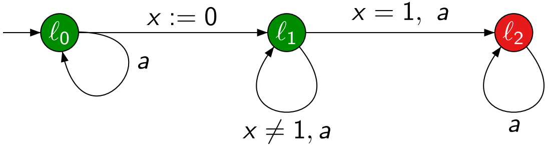
[Lasota,Walukiewicz 2005,2007] [Ouaknine,Worrell 2005,2007]

Example

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

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

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



[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

- Emptiness of ATA is undecidable.
- Emptiness of one-clock ATA is decidable, but non-primitive recursive.
- Emptiness for Büchi properties of one-clock ATA is undecidable.
- Emptiness of one-clock ATA with ε-transitions is undecidable.

[Lasota,Walukiewicz 2005]

- nice closure properties

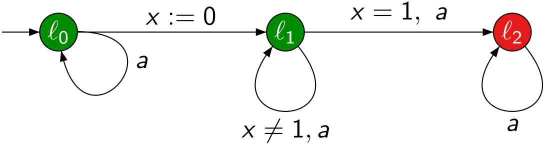
  ➜ universality is as difficult as reachability
- more expressive than timed automata

Theorem

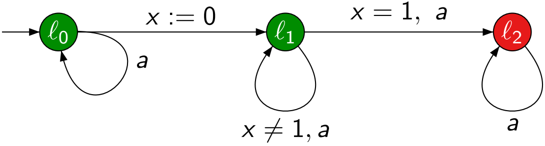
- Emptiness of ATA is undecidable.
- Emptiness of one-clock ATA is decidable, but non-primitive recursive.
- Emptiness for Büchi properties of one-clock ATA is undecidable.
- Emptiness of one-clock ATA with ε-transitions is undecidable.

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

Example

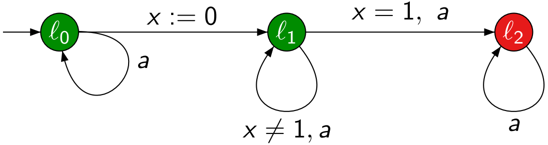


Example



Execution over timed word $(a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)$

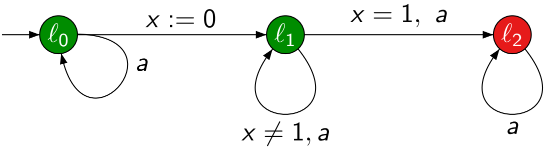
Example



Execution over timed word $(a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)$

$$\{\ (\ell_0, 0)\ \}$$

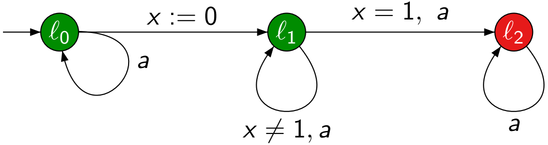
Example



Execution over timed word $(a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)$

$$\{\boxed{(\ell_0, 0)}\}$$
$$\downarrow$$
$$\{\boxed{(\ell_0, .3)}, \boxed{(\ell_1, 0)}\}$$

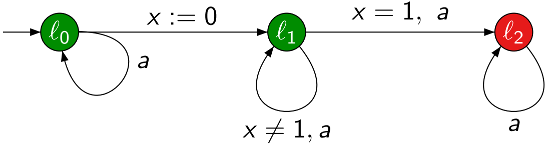
Example



Execution over timed word $(a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)$

$$\{\boxed{(\ell_0, 0)}\}$$
$$\downarrow$$
$$\{\boxed{(\ell_0, .3)}, \boxed{(\ell_1, 0)}\}$$
$$\downarrow$$
$$\{\boxed{(\ell_0, .8)}, \boxed{(\ell_1, 0)}, \boxed{(\ell_1, .5)}\}$$

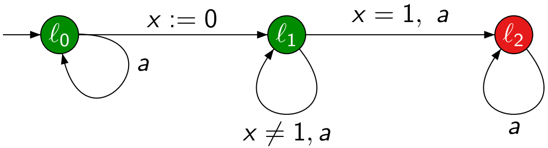
Example



Execution over timed word $(a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)$

$$\{\ (\ell_0, 0)\ \}$$
$$\downarrow$$
$$\{\ (\ell_0, .3),\ (\ell_1, 0)\ \}$$
$$\downarrow$$
$$\{\ (\ell_0, .8),\ (\ell_1, 0),\ (\ell_1, .5)\ \}$$
$$\downarrow$$
$$\{\ (\ell_0, 1.4),\ (\ell_1, 0),\ (\ell_1, .6),\ (\ell_1, 1.1)\ \}$$

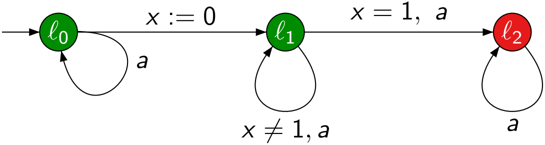
Example



Execution over timed word $(a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)$

$$\{ (\ell_0, 0) \}$$
$$\downarrow$$
$$\{ (\ell_0, .3), (\ell_1, 0) \}$$
$$\downarrow$$
$$\{ (\ell_0, .8), (\ell_1, 0), (\ell_1, .5) \}$$
$$\downarrow$$
$$\{ (\ell_0, 1.4), (\ell_1, 0), (\ell_1, .6), (\ell_1, 1.1) \}$$
$$\downarrow$$
$$\{ (\ell_0, 1.8), (\ell_1, 0), (\ell_1, .4), (\ell_2, 1), (\ell_1, 1.5) \}$$

Example



Execution over timed word $(a, .3)(a, .8)(a, 1.4)(a, 1.8)(a, 2)$

$$\{(\ell_0, 0)\}$$
$$\downarrow$$
$$\{(\ell_0, .3), (\ell_1, 0)\}$$
$$\downarrow$$
$$\{(\ell_0, .8), (\ell_1, 0), (\ell_1, .5)\}$$
$$\downarrow$$
$$\{(\ell_0, 1.4), (\ell_1, 0), (\ell_1, .6), (\ell_1, 1.1)\}$$
$$\downarrow$$
$$\{(\ell_0, 1.8), (\ell_1, 0), (\ell_1, .4), (\ell_2, 1), (\ell_1, 1.5)\}$$
$$\downarrow$$
$$\{(\ell_0, 2), (\ell_1, 0), (\ell_1, .2), (\ell_1, .6), (\ell_2, 1.2), (\ell_1, 1.7)\}$$

An abstraction

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

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

An abstraction

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

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

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

0.0

An abstraction

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

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

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

0.0 0.2

An abstraction

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

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

$\{(\ell, 0), (\ell', 1)\}$ $\{(\ell, 1)\}$ $\{(\ell, 0), (\ell, 2)\}$

0.0 0.2 0.3

An abstraction

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

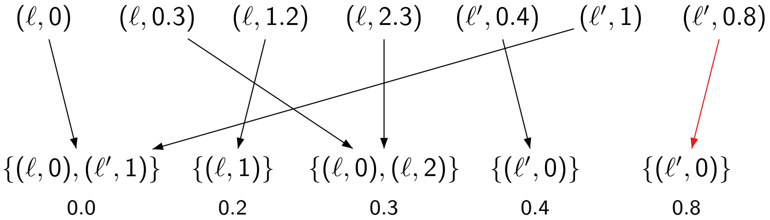
$(\ell, 0) \quad (\ell, 0.3) \quad (\ell, 1.2) \quad (\ell, 2.3) \quad (\ell', 0.4) \quad (\ell', 1) \quad (\ell', 0.8)$

$\{(\ell, 0), (\ell', 1)\} \quad \{(\ell, 1)\} \quad \{(\ell, 0), (\ell, 2)\} \quad \{(\ell', 0)\}$

0.0 0.2 0.3 0.4

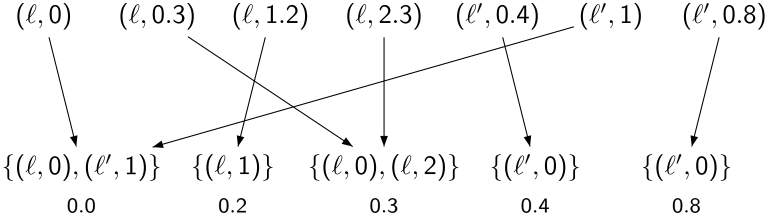
An abstraction

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



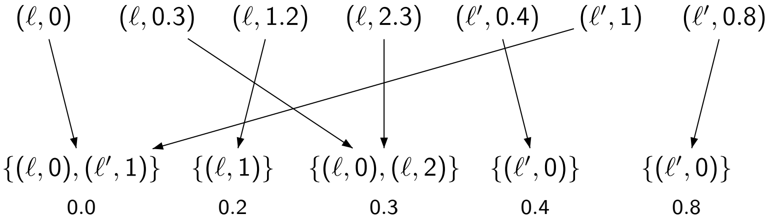
An abstraction

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



An abstraction

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



Abstracted into: $\{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \cdot \{(\ell', 0)\}$

Abstract transition system

$\boxed{\{(\ell, 0), (\ell', 1)\}} \cdot \boxed{\{(\ell, 1)\}} \cdot \boxed{\{(\ell, 0), (\ell, 2)\}} \cdot \boxed{\{(\ell', 0)\}} \cdot \boxed{\{(\ell', 0)\}}$

Abstract transition system

$\boxed{\{(\ell,0),(\ell',1)\}} \cdot \boxed{\{(\ell,1)\}} \cdot \boxed{\{(\ell,0),(\ell,2)\}} \cdot \boxed{\{(\ell',0)\}} \cdot \boxed{\{(\ell',0)\}}$

Time successors:

Abstract transition system

$\{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\} \cdot \{(\ell', 0)\}$

Time successors:

$\{(\ell', 1)\} \cdot \{(\ell, 0), (\ell', 1)\} \cdot \{(\ell, 1)\} \cdot \{(\ell, 0), (\ell, 2)\} \cdot \{(\ell', 0)\}$

Abstract transition system

$\{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\} \cdot \{(\ell',0)\}$

Time successors:

$\{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\}$

$\{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\}$

Abstract transition system

$\{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\} \cdot \{(\ell',0)\}$

Time successors:

$\{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\}$

$\{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\}$

$\{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\}$

Abstract transition system

$\{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\} \cdot \{(\ell',0)\}$

Time successors:

$\{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\}$

$\{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\}$

$\{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\}$

$\{(\ell,2)\} \cdot \{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\}$

Abstract transition system

$\{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\} \cdot \{(\ell',0)\}$

Time successors:

$\{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\}$

$\{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\}$

$\{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\}$

$\{(\ell,2)\} \cdot \{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\}$

$\{(\ell,1),(\ell',2)\} \cdot \{(\ell,2)\} \cdot \{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\}$

Abstract transition system

$$\{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\} \cdot \{(\ell',0)\}$$

Time successors:

$$\{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\}$$

$$\{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\}$$

$$\{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\}$$

$$\{(\ell,2)\} \cdot \{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\}$$

$$\{(\ell,1),(\ell',2)\} \cdot \{(\ell,2)\} \cdot \{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\}$$

Transition $\ell \xrightarrow{x>2, x:=0} \ell''$:

Abstract transition system

$\{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\} \cdot \{(\ell',0)\}$

Time successors:

$\{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\} \cdot \{(\ell',0)\}$

$\{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\} \cdot \{(\ell,0),(\ell,2)\}$

$\{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\} \cdot \{(\ell,1)\}$

$\{(\ell,2)\} \cdot \{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\} \cdot \{(\ell,0),(\ell',1)\}$

$\{(\ell,1),(\ell',2)\} \cdot \{(\ell,2)\} \cdot \{(\ell,1),(\ell,3)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\}$

Transition $\ell \xrightarrow{x>2,x:=0} \ell''$:

$\{(\ell'',0)\} \cdot \{(\ell,1),(\ell',2)\} \cdot \{(\ell,1)\} \cdot \{(\ell',1)\} \cdot \{(\ell',1)\}$

What can we do with that abstract transition system?

Correctness?

What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

- ☹ possibly infinitely many abstract configurations

What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

- ☹ possibly infinitely many abstract configurations
- ☺ there is a well-quasi ordering on the set of abstract configurations! (subword relation $\sqsubseteq$)

What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

- ☹ possibly infinitely many abstract configurations
- ☺ there is a well-quasi ordering on the set of abstract configurations! (subword relation $\sqsubseteq$)
  $+$ downward compatibility:
  $$(\gamma_1 \sqsubseteq \gamma_1' \text{ and } \gamma_1' \rightsquigarrow \gamma_2') \Rightarrow (\gamma_1 \rightsquigarrow^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma_2')$$

What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

- ☹ possibly infinitely many abstract configurations
- ☺ there is a well-quasi ordering on the set of abstract configurations! (subword relation $\sqsubseteq$)
  $+$ downward compatibility:
  $$(\gamma_1 \sqsubseteq \gamma_1' \text{ and } \gamma_1' \rightsquigarrow \gamma_2') \Rightarrow (\gamma_1 \rightsquigarrow^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma_2')$$
  $+$ downward-closed objective (all states are accepting)

What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

- ☹ possibly infinitely many abstract configurations
- ☺ there is a well-quasi ordering on the set of abstract configurations! (subword relation $\sqsubseteq$)
  $+$ downward compatibility:
  $$(\gamma_1 \sqsubseteq \gamma_1' \text{ and } \gamma_1' \rightsquigarrow \gamma_2') \Rightarrow (\gamma_1 \rightsquigarrow^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma_2')$$
  $+$ downward-closed objective (all states are accepting)

  Recipe learned on Monday:

What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

- ☹ possibly infinitely many abstract configurations
- ☺ there is a well-quasi ordering on the set of abstract configurations! (subword relation $\sqsubseteq$)
  + downward compatibility:
  $$(\gamma_1 \sqsubseteq \gamma_1' \text{ and } \gamma_1' \rightsquigarrow \gamma_2') \Rightarrow (\gamma_1 \rightsquigarrow^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma_2')$$
  + downward-closed objective (all states are accepting)

  Recipe learned on Monday:

  $$(\text{Higman's lemma} + \text{Koenig's lemma}) \Rightarrow \text{termination}$$

What can we do with that abstract transition system?

Correctness?

The previous abstraction is (almost) a time-abstract bisimulation.

Termination?

- ☹ possibly infinitely many abstract configurations
- ☺ there is a well-quasi ordering on the set of abstract configurations! (subword relation $\sqsubseteq$)
  $+$ downward compatibility:
  $$(\gamma_1 \sqsubseteq \gamma_1' \text{ and } \gamma_1' \rightsquigarrow \gamma_2') \Rightarrow (\gamma_1 \rightsquigarrow^* \gamma_2 \text{ and } \gamma_2 \sqsubseteq \gamma_2')$$
  $+$ downward-closed objective (all states are accepting)

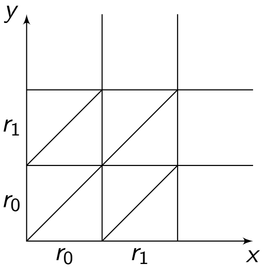
  Recipe learned on Monday:

  $$(\text{Higman's lemma} + \text{Koenig's lemma}) \Rightarrow \text{termination}$$

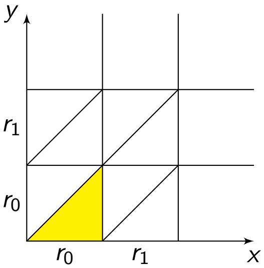
Alternative

The abstract transition system can be simulated by a kind of FIFO channel machine.

A digression on timed automata



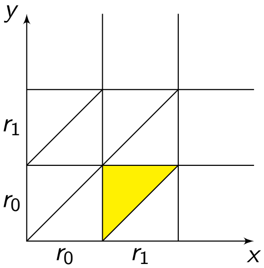
A digression on timed automata



$x, y \in r_0$, $\{y\} < \{x\}$

$\boxed{(y, r_0)} \cdot \boxed{(x, r_0)}$

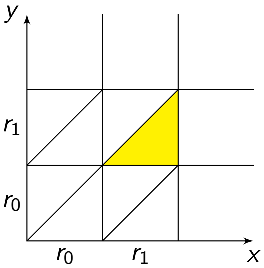
A digression on timed automata



$x \in r_1$, $y \in r_0$, $\{x\} < \{y\}$

$(x, r_1) \cdot (y, r_0)$

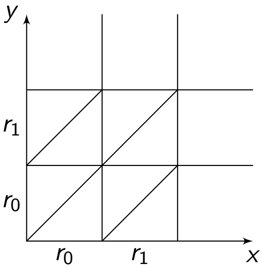
A digression on timed automata



$x, y \in r_1, \{y\} < \{x\}$

$\boxed{(y, r_1)} \cdot \boxed{(x, r_1)}$

A digression on timed automata



The classical region automaton can be simulated by a channel machine (with a single bounded channel).

Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata **[Abdulla,Jonsson 1998]**

Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata **[Abdulla,Jonsson 1998]**
- timed Petri nets **[Abdulla,Nylén 2001]**

Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata **[Abdulla,Jonsson 1998]**
- timed Petri nets **[Abdulla,Nylén 2001]**
- MTL model checking **[Ouaknine,Worrell 2005,2007]**

Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata **[Abdulla,Jonsson 1998]**
- timed Petri nets **[Abdulla,Nylén 2001]**
- MTL model checking **[Ouaknine,Worrell 2005,2007]**
- coFlatMTL model checking **[Bouyer,Markey,Ouaknine,Worrell 2007]**
  (using channel machines with a bounded number of cycles)

Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata **[Abdulla,Jonsson 1998]**
- timed Petri nets **[Abdulla,Nylén 2001]**
- MTL model checking **[Ouaknine,Worrell 2005,2007]**
- coFlatMTL model checking **[Bouyer,Markey,Ouaknine,Worrell 2007]**
  (using channel machines with a bounded number of cycles)
- single-clock automata inclusion checking **[Ouaknine,Worrel 2004]**

Partial conclusion

Similar technics apply to:

- networks of single-clock timed automata **[Abdulla,Jonsson 1998]**
- timed Petri nets **[Abdulla,Nylén 2001]**
- MTL model checking **[Ouaknine,Worrell 2005,2007]**
- coFlatMTL model checking **[Bouyer,Markey,Ouaknine,Worrell 2007]**
  (using channel machines with a bounded number of cycles)
- single-clock automata inclusion checking **[Ouaknine,Worrel 2004]**
- ...

Outline

Conclusion

- Justification of the dense-time semantics

Conclusion

- Justification of the dense-time semantics
- Two main technics for proving decidability of real-time systems
  - Are missing: initialized rectangular automata

Conclusion

- Justification of the dense-time semantics
- Two main technics for proving decidability of real-time systems
  - Are missing: initialized rectangular automata

Some current streams of research in timed systems:

- quantitative model-checking
- real-time logics
- robustness, implementability issues
- timed games
- ...