Modeling, Verification and Applications of Explicit Time Models

Béatrice Bérard

LAMSADE
Université Paris-Dauphine & CNRS
berard@lamsade.dauphine.fr
ANR Project DOTS

PNTAP'08, March 3rd 2008

Verification is necessary

especially...

Verification is necessary

especially...

for critical systems

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
- reachability on $\mathcal{S} \parallel A_T$, product of $\mathcal{S}$ with testing automaton A_T
- ...

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
- reachability on $\mathcal{S} \parallel A_T$, product of $\mathcal{S}$ with testing automaton A_T
- ...

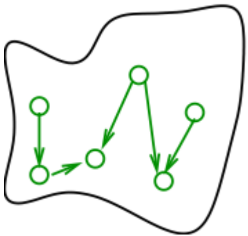
system

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
- reachability on $\mathcal{S} \parallel A_T$, product of $\mathcal{S}$ with testing automaton A_T
- ...

system

Modeling



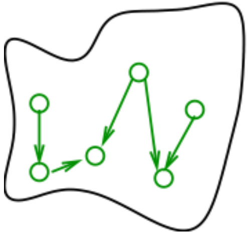
Classical verification problems

- 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
- reachability on $\mathcal{S} \parallel A_T$, product of $\mathcal{S}$ with testing automaton A_T
- ...

Does the system meet its specification ?

Modeling _ _ _ _ _ _ _ _ _

$\mathcal{S}$



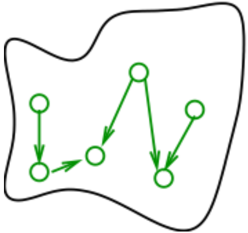
Classical verification problems

- 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
- reachability on $\mathcal{S} \parallel A_T$, product of $\mathcal{S}$ with testing automaton A_T
- ...

Does the system meet its specification ?

Modeling

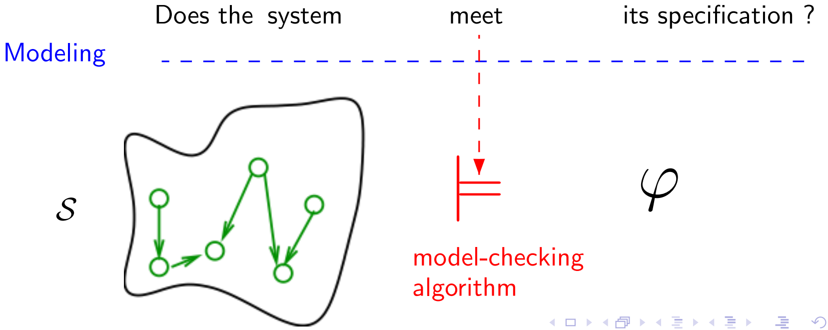
$\mathcal{S}$



φ

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
- reachability on $\mathcal{S} \parallel A_T$, product of $\mathcal{S}$ with testing automaton A_T
- ...

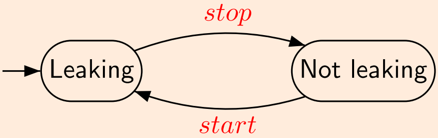


Why add time ?

The gas burner example [ACHH93]

The gas burner may leak and :

- each time a leakage is detected, it is repaired or stopped in less than 1s
- two leakages are separated by at least 30s



Is it possible that the gas burner leaks during a time greater than $\frac{1}{20}$ of the global time after the first 60s?

Timed features are needed in the model and in the properties:

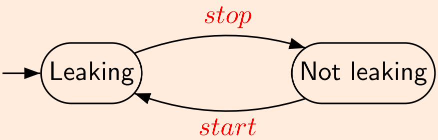
Instead of observing a sequence of events $a_1 a_2 \ldots$,
observe a sequence of pairs $(a_1, t_1)(a_2, t_2)\ldots$ where t_i is the time at which a_i occurs.

Why add time ?

The gas burner example [ACHH93]

The gas burner may leak and :

- each time a leakage is detected, it is repaired or stopped in less than 1s
- two leakages are separated by at least 30s



Is it possible that the gas burner leaks during a time greater than $\frac{1}{20}$ of the global time after the first 60s?

Timed features are needed in the model and in the properties:

Instead of observing a sequence of events $a_1 a_2 \ldots$,
observe a sequence of pairs $(a_1, t_1)(a_2, t_2)\ldots$ where t_i is the time at which a_i occurs.

Outline

Timed Models

Verification

Applications

Conclusion

Outline

Timed Models

Verification

Applications

Conclusion

Transition systems

Definition

$\color{red}{Act}$ alphabet of actions

$\mathcal{T} = (S, s_0, E)$ transition system

- S set of configurations, s_0 initial configuration,
- $E \subseteq S \times Act \times S$ contains

  action transitions: $s \xrightarrow{a} s'$, instantaneous execution of a

Example: a finite automaton

An execution:

ok $\xrightarrow{p}$ fault $\xrightarrow{r}$ ok $\xrightarrow{p}$ fault $\xrightarrow{h}$ alarm $\xrightarrow{r}$ ok $\longrightarrow \cdots$

Transition systems

Definition

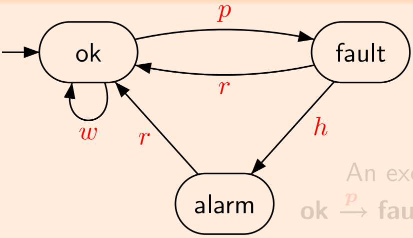
Act alphabet of actions

$\mathcal{T} = (S, s_0, E)$ transition system

- S set of configurations, s_0 initial configuration,
- $E \subseteq S \times Act \times S$ contains

  action transitions: $s \xrightarrow{a} s'$, instantaneous execution of a

Example: a finite automaton



w working, p problem

r return, h handling

An execution:

ok $\xrightarrow{p}$ fault $\xrightarrow{r}$ ok $\xrightarrow{p}$ fault $\xrightarrow{h}$ alarm $\xrightarrow{r}$ ok $\longrightarrow \cdots$

Transition systems

Definition

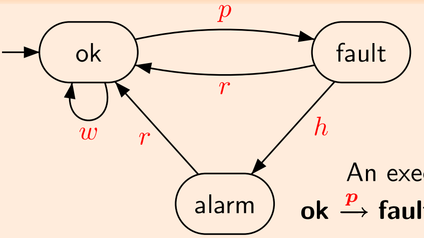
Act alphabet of actions

$\mathcal{T} = (S, s_0, E)$ transition system

- S set of configurations, s_0 initial configuration,
- $E \subseteq S \times Act \times S$ contains

  action transitions: $\boldsymbol{s \xrightarrow{a} s'}$, instantaneous execution of $\boldsymbol{a}$

Example: a finite automaton



w working, p problem

r return, h handling

An execution:

ok $\xrightarrow{p}$ **fault** $\xrightarrow{r}$ **ok** $\xrightarrow{p}$ **fault** $\xrightarrow{h}$ **alarm** $\xrightarrow{r}$ **ok** $\longrightarrow \cdots$

Timed Transition Systems

Definition

Act alphabet of actions,

$\mathcal{T} = (S, s_0, L, E)$ transition system

- S set of configurations, s_0 initial configuration,
- $E \subseteq S \times Act \times S$ contains

  action transitions: $s \xrightarrow{a} s'$, instantaneous execution of a

  delay transitions: $s \xrightarrow{d} s'$, time elapsing for d time units.

Timed Transition Systems

Definition

Act alphabet of actions, $\mathbb{T}$ time domain contained in $\mathbb{R}_{\geq 0}$,

$\mathcal{T} = (S, s_0, L, E)$ timed transition system

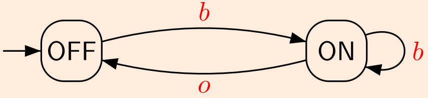
- S set of configurations, s_0 initial configuration,
- $E \subseteq S \times (Act \cup \mathbb{T}) \times S$ contains

  action transitions: $s \xrightarrow{a} s'$, instantaneous execution of a

  delay transitions: $s \xrightarrow{d} s'$, time elapsing for d time units.

Why not discretize ?

A time switch



b button pressed

o light off

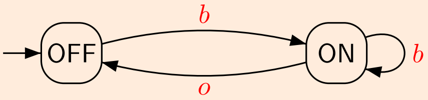
Unfolding with discrete time

when adding the constraint: the light stays on exactly 3 time units once the button is pressed.

may lead to state explosion.

Why not discretize ?

A time switch

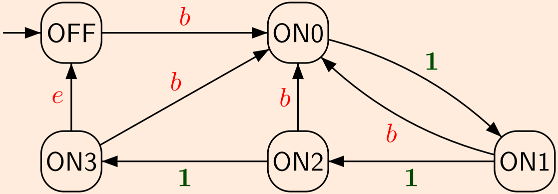


b button pressed

o light off

Unfolding with discrete time

when adding the constraint: the light stays on exactly 3 time units once the button is pressed.

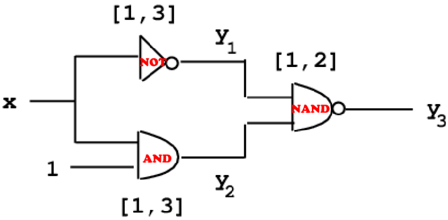


$\mathbf{1}$ wait for 1 t.u.

may lead to state explosion.

Discussion: reachable configurations

for asynchronous digital circuits [Alur 1991] [Brzozowski Seger 1991]



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

Input x changes to 1. The corresponding stable configuration is y=[011]

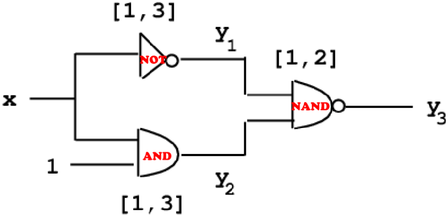
However, many possible behaviours, e.g.

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

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

Discussion: reachable configurations

for asynchronous digital circuits [Alur 1991] [Brzozowski Seger 1991]



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

Input x changes to 1. The corresponding stable configuration is y=[011]

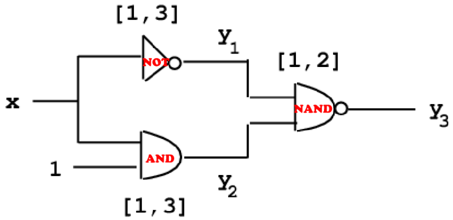
However, many possible behaviours, e.g.

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

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

Discussion: reachable configurations

for asynchronous digital circuits [Alur 1991] [Brzozowski Seger 1991]



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

Input x changes to 1. The corresponding stable configuration is y=[011]

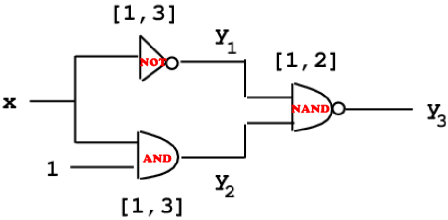
However, many possible behaviours, e.g.

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

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

Discussion: reachable configurations

for asynchronous digital circuits [Alur 1991] [Brzozowski Seger 1991]



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

Input x changes to 1. The corresponding stable configuration is y=[011]

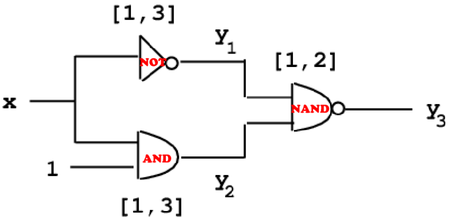
However, many possible behaviours, e.g.

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

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

Discussion: reachable configurations

for asynchronous digital circuits [Alur 1991] [Brzozowski Seger 1991]



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

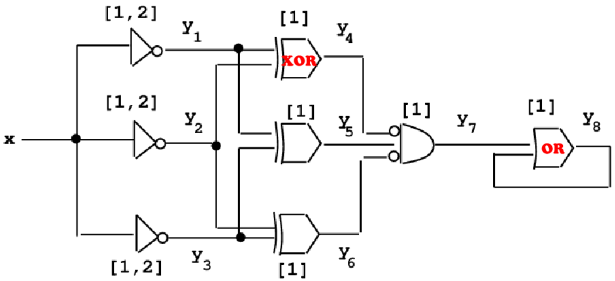
Input x changes to 1. The corresponding stable configuration is y=[011]

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: $\{[101], [111], [110], [010], [011], [001]\}$

A circuit which is not 1-discretizable



Why? initially $x = 0$ and $y = [11100000]$, then x is set to 1

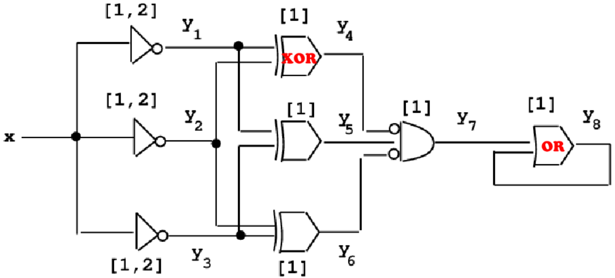
$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$

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

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

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

A circuit which is not 1-discretizable



Why? initially $x = 0$ and $y = [11100000]$, then x is set to 1

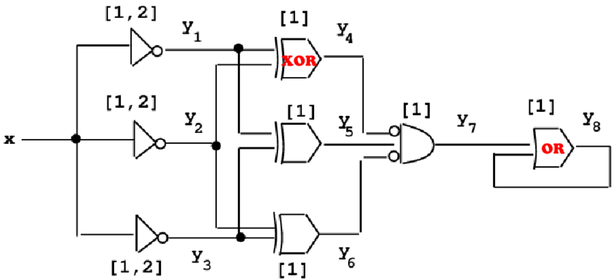
$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3,y_5} [00001000] \xrightarrow[3]{y_5,y_7} [00000010] \xrightarrow[4]{y_7,y_8} [00000001]$

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

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

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

A circuit which is not 1-discretizable



Why? initially $x = 0$ and $y = [11100000]$, then x is set to 1

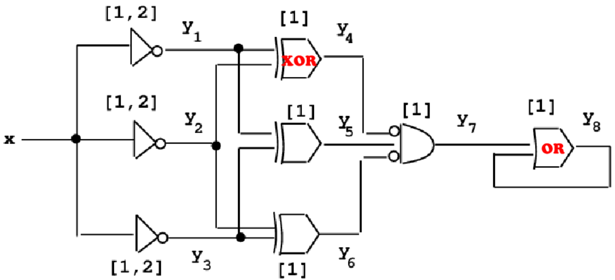
$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$

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

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

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

A circuit which is not 1-discretizable



Why?

initially x = 0 and y = [11100000], then x is set to 1

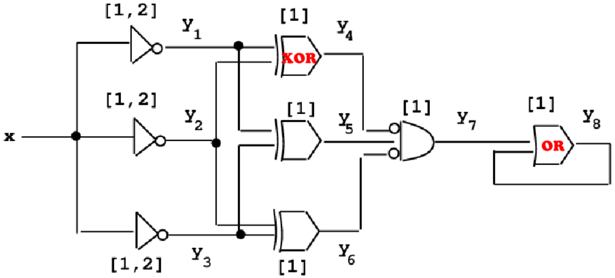
$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$

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

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

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

A circuit which is not 1-discretizable



Why? initially $x = 0$ and $y = [11100000]$, then x is set to 1

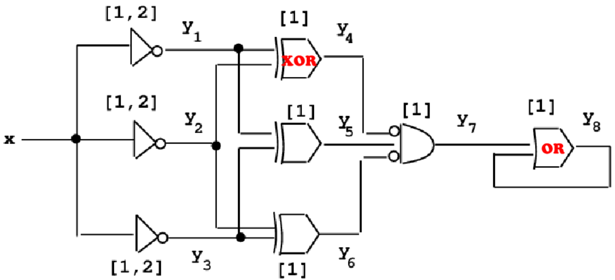
$$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$$

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

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

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

A circuit which is not 1-discretizable



Why? initially x = 0 and y = [11100000], then x is set to 1

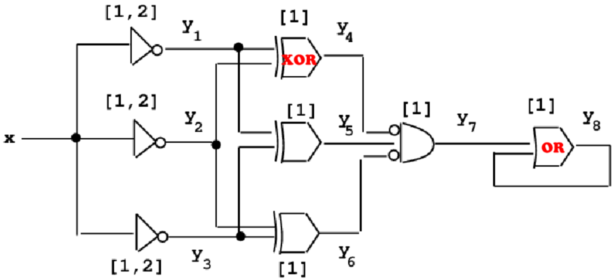
$[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3,y_5} [00001000] \xrightarrow[3]{y_5,y_7} [00000010] \xrightarrow[4]{y_7,y_8} [00000001]$

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

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

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

A circuit which is not 1-discretizable



Why?

initially x = 0 and y = [11100000], then 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?

Theorem [Brzozowski Seger 1991]

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

Consequence

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

Furthermore

there exist systems for which no discrete execution is possible, whatever the granularity choice.

(see later)

Is discretizing sufficient?

Theorem [Brzozowski Seger 1991]

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

Consequence

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

Furthermore

there exist systems for which no discrete execution is possible, whatever the granularity choice.

(see later)

Is discretizing sufficient?

Theorem [Brzozowski Seger 1991]

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

Consequence

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

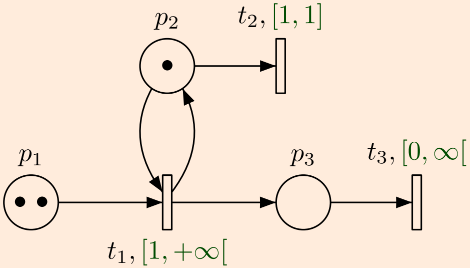
Furthermore

there exist systems for which no discrete execution is possible, whatever the granularity choice.

(see later)

Adding time intervals on transitions (1)

Example 1: Time Petri Nets [Merlin 1974]

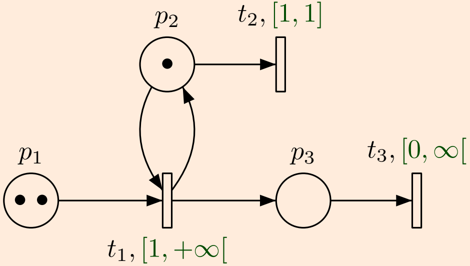


Markings: $M_0 = (2, 1, 0)$, $M_1 = (1, 1, 1)$, $M_2 = (0, 1, 2)$, $M_3 = (0, 0, 2)$
Time valuation of a transition t: time since t was last enabled, $\perp$ if t is not enabled.

An execution:
$(M_0, [0, 0, \perp]) \xrightarrow{1} (M_0, [1, 1, \perp]) \xrightarrow{t_1} (M_1, [1, 1, 0]) \xrightarrow{t_1} (M_2, [\perp, 1, 0]) \xrightarrow{t_2}$
$(M_3, [\perp, \perp, 0]) \xrightarrow{1.5} (M_3, [\perp, \perp, 1.5]) \cdots$

Adding time intervals on transitions (1)

Example 1: Time Petri Nets [Merlin 1974]

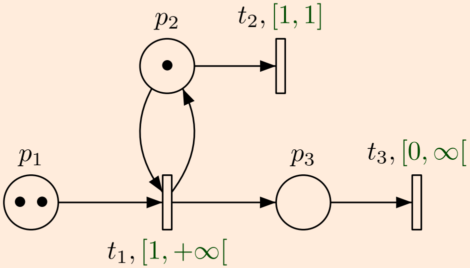


Markings: $M_0 = (2, 1, 0)$, $M_1 = (1, 1, 1)$, $M_2 = (0, 1, 2)$, $M_3 = (0, 0, 2)$
Time valuation of a transition t: time since t was last enabled, $\perp$ if t is not enabled.

An execution:
$(M_0, [0, 0, \perp]) \xrightarrow{1} (M_0, [1, 1, \perp]) \xrightarrow{t_1} (M_1, [1, 1, 0]) \xrightarrow{t_1} (M_2, [\perp, 1, 0]) \xrightarrow{t_2} (M_3, [\perp, \perp, 0]) \xrightarrow{1.5} (M_3, [\perp, \perp, 1.5]) \cdots$

Adding time intervals on transitions (1)

Example 1: Time Petri Nets [Merlin 1974]



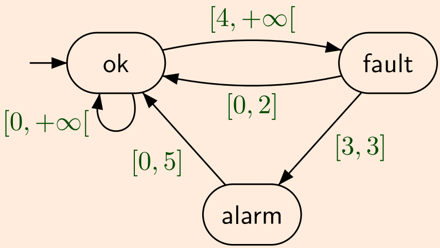
Markings: $M_0 = (2, 1, 0)$, $M_1 = (1, 1, 1)$, $M_2 = (0, 1, 2)$, $M_3 = (0, 0, 2)$
Time valuation of a transition t: time since t was last enabled, $\perp$ if t is not enabled.

An execution:

$(\boldsymbol{M_0}, [\mathbf{0}, \mathbf{0}, \perp]) \xrightarrow{\mathbf{1}} (\boldsymbol{M_0}, [\mathbf{1}, \mathbf{1}, \perp]) \xrightarrow{\boldsymbol{t_1}} (\boldsymbol{M_1}, [\mathbf{1}, \mathbf{1}, \mathbf{0}]) \xrightarrow{\boldsymbol{t_1}} (\boldsymbol{M_2}, [\perp, \mathbf{1}, \mathbf{0}]) \xrightarrow{\boldsymbol{t_2}}$
$(\boldsymbol{M_3}, [\perp, \perp, \mathbf{0}]) \xrightarrow{\mathbf{1.5}} (\boldsymbol{M_3}, [\perp, \perp, \mathbf{1.5}]) \cdots$

Adding time intervals on transitions (2)

Example 2: finite automata with delays [Emerson et al. 1992]

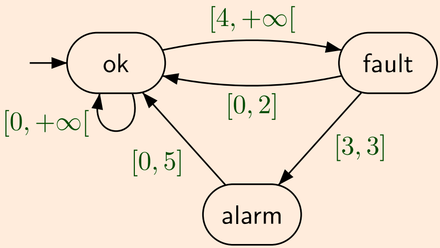


An execution: **ok** $\xrightarrow{15}$ **fault** $\xrightarrow{1.5}$ **ok** $\xrightarrow{8}$ **fault** $\xrightarrow{3}$ q_2 $\xrightarrow{2.7}$ **ok** $\cdots$

Remark: only delay transitions

Adding time intervals on transitions (2)

Example 2: finite automata with delays [Emerson et al. 1992]

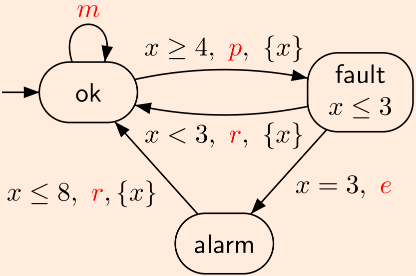


An execution: **ok** $\xrightarrow{15}$ **fault** $\xrightarrow{1.5}$ **ok** $\xrightarrow{8}$ **fault** $\xrightarrow{3}$ $\boldsymbol{q_2}$ $\xrightarrow{2.7}$ **ok** $\cdots$

Remark: only delay transitions

Adding clocks: timed automata (1)

A variation of [Alur Dill 1990]



$\boldsymbol{x}$ real valued clock

$\boldsymbol{x < 3}$, $\boldsymbol{x = 3}$, $\boldsymbol{x \geq 4}$ guards

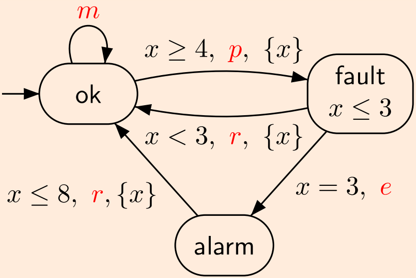
$\boldsymbol{x \leq 3}$ invariant

$\boldsymbol{\{x\}}$ reset operation for $\boldsymbol{x}$

also written $\boldsymbol{x := 0}$

Adding clocks: timed automata (1)

A variation of [Alur Dill 1990]



x real valued clock

$x < 3$, $x = 3$, $x \geq 4$ guards

$x \leq 3$ invariant

$\{x\}$ reset operation for x

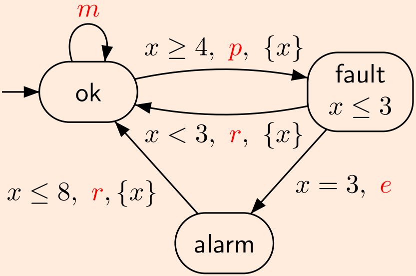
also written $x := 0$

Clock valuations and clock constraints

X a set of clocks, valuation $v : X \mapsto \mathbb{R}_{\geq 0}$,
$\mathcal{C}(X)$ set of clock constraints: conjunctions of atomic constraints of the form $x \bowtie c$, for clock x, constant c and $\bowtie$ in $\{<, \leq, =, \geq, >\}$.

Adding clocks: timed automata (1)

A variation of [Alur Dill 1990]



x real valued clock

$x < 3$, $x = 3$, $x \geq 4$ guards

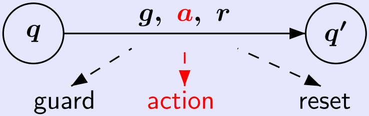
$x \leq 3$ invariant

$\{x\}$ reset operation for x

also written $x := 0$

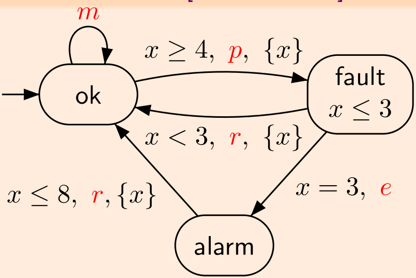
Timed automaton $\mathcal{A} = (Q, q_0, Inv, \Delta)$

- Q set of (control) states, q_0 initial state,
- Inv associates an invariant with each state
- Δ contains transitions :



Adding clocks : timed automata (2)

A variation of [Alur Dill 1990]

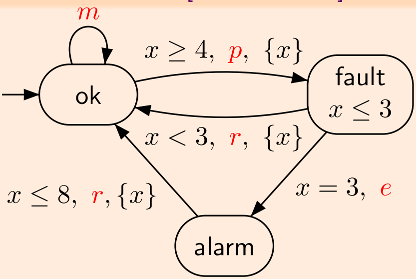


An execution: $(\mathbf{ok}, [0]) \xrightarrow{8.3} (\mathbf{ok}, [8.3]) \xrightarrow{p} (\mathbf{fault}, [0]) \xrightarrow{3} (\mathbf{fault}, [3]) \xrightarrow{e} (\mathbf{alarm}, [3]) \xrightarrow{2.1} (\mathbf{alarm}, [5.1]) \xrightarrow{r} (\mathbf{ok}, [0]) \cdots$

Timed observation: $(p, 8.3)(e, 11.3)(r, 13.4)\ldots$

Adding clocks : timed automata (2)

A variation of [Alur Dill 1990]



Configurations: (q, v)

v value of x satisfying

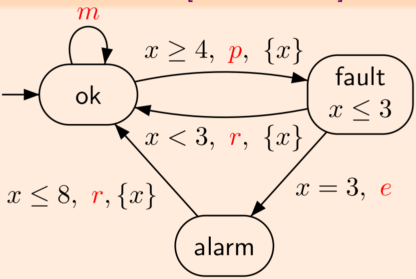
the invariant

An execution: $(\textbf{ok}, [0]) \xrightarrow{8.3} (\textbf{ok}, [8.3]) \xrightarrow{p} (\textbf{fault}, [0]) \xrightarrow{3} (\textbf{fault}, [3])$
$\xrightarrow{e} (\textbf{alarm}, [3]) \xrightarrow{2.1} (\textbf{alarm}, [5.1]) \xrightarrow{r} (\textbf{ok}, [0]) \cdots$

Timed observation: $(p, 8.3)(e, 11.3)(r, 13.4)\ldots$

Adding clocks : timed automata (2)

A variation of [Alur Dill 1990]



Configurations: (q, v)

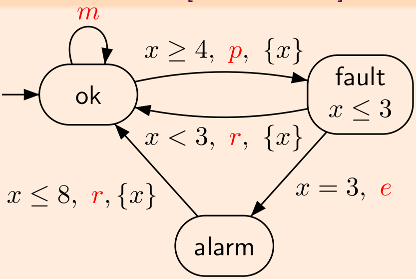
v value of x satisfying the invariant

An execution: $(\mathbf{ok}, [0]) \xrightarrow{8.3} (\mathbf{ok}, [8.3]) \xrightarrow{p} (\mathbf{fault}, [0]) \xrightarrow{3} (\mathbf{fault}, [3]) \xrightarrow{e} (\mathbf{alarm}, [3]) \xrightarrow{2.1} (\mathbf{alarm}, [5.1]) \xrightarrow{r} (\mathbf{ok}, [0]) \cdots$

Timed observation: $(p, 8.3)(e, 11.3)(r, 13.4)\ldots$

Adding clocks : timed automata (2)

A variation of [Alur Dill 1990]



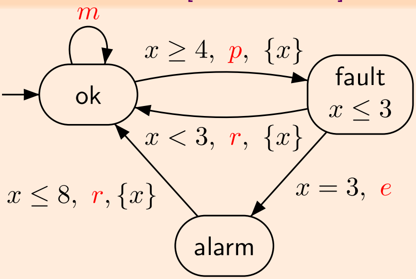
Configurations: (q, v)
v value of x satisfying the invariant

An execution: $(\textbf{ok}, [0]) \xrightarrow{8.3} (\textbf{ok}, [8.3]) \xrightarrow{p} (\textbf{fault}, [0]) \xrightarrow{3} (\textbf{fault}, [3]) \xrightarrow{e} (\textbf{alarm}, [3]) \xrightarrow{2.1} (\textbf{alarm}, [5.1]) \xrightarrow{r} (\textbf{ok}, [0]) \cdots$

Timed observation: $(p, 8.3)(e, 11.3)(r, 13.4)\ldots$

Adding clocks : timed automata (2)

A variation of [Alur Dill 1990]



Configurations: (q, v)

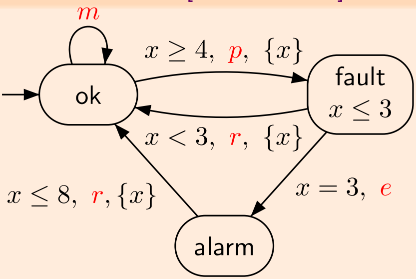
v value of x satisfying the invariant

An execution: $(\mathbf{ok}, [0]) \xrightarrow{8.3} (\mathbf{ok}, [8.3]) \xrightarrow{p} (\mathbf{fault}, [0]) \xrightarrow{3} (\mathbf{fault}, [3])$
$\xrightarrow{e} (\mathbf{alarm}, [3]) \xrightarrow{2.1} (\mathbf{alarm}, [5.1]) \xrightarrow{r} (\mathbf{ok}, [0]) \cdots$

Timed observation: $(p, 8.3)(e, 11.3)(r, 13.4)\ldots$

Adding clocks : timed automata (2)

A variation of [Alur Dill 1990]



Configurations: (q, v)

v value of x satisfying

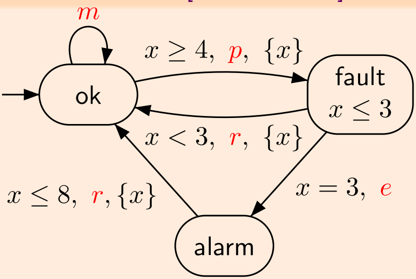
the invariant

An execution: $(\textbf{ok}, [0]) \xrightarrow{8.3} (\textbf{ok}, [8.3]) \xrightarrow{p} (\textbf{fault}, [0]) \xrightarrow{3} (\textbf{fault}, [3])$ $\xrightarrow{e} (\textbf{alarm}, [3]) \xrightarrow{2.1} (\textbf{alarm}, [5.1]) \xrightarrow{r} (\textbf{ok}, [0]) \cdots$

Timed observation: $(p, 8.3)(e, 11.3)(r, 13.4)\ldots$

Adding clocks : timed automata (2)

A variation of [Alur Dill 1990]



Configurations: (q, v)

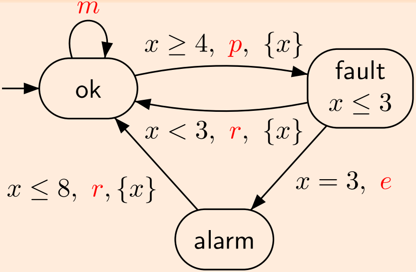
v value of x satisfying the invariant

An execution: $(\textbf{ok}, [0]) \xrightarrow{8.3} (\textbf{ok}, [8.3]) \xrightarrow{p} (\textbf{fault}, [0]) \xrightarrow{3} (\textbf{fault}, [3]) \xrightarrow{e} (\textbf{alarm}, [3]) \xrightarrow{2.1} (\textbf{alarm}, [5.1]) \xrightarrow{r} (\textbf{ok}, [0]) \cdots$

Timed observation: $(p, 8.3)(e, 11.3)(r, 13.4)\ldots$

Adding clocks : timed automata (2)

A variation of [Alur Dill 1990]



Configurations: (q, v)

v value of x satisfying the invariant

An execution: $(\textbf{ok}, [0]) \xrightarrow{8.3} (\textbf{ok}, [8.3]) \xrightarrow{p} (\textbf{fault}, [0]) \xrightarrow{3} (\textbf{fault}, [3]) \xrightarrow{e} (\textbf{alarm}, [3]) \xrightarrow{2.1} (\textbf{alarm}, [5.1]) \xrightarrow{r} (\textbf{ok}, [0]) \cdots$

Timed observation: $(p, 8.3)(e, 11.3)(r, 13.4)\ldots$

Semantics of timed automata (1)

Operations on valuations

X set of clocks. For valuation v :

- for a subset r of X, valuation $v[r \mapsto 0]$ is obtained by reset of the clocks in r, other values unchanged,
- for a duration d, valuation $v + d$ is obtained by adding d to all clock values.

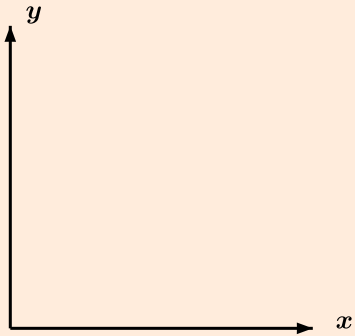
Semantics of timed automata (1)

Operations on valuations

X set of clocks. For valuation v :

- for a subset r of X, valuation $v[r \mapsto 0]$ is obtained by reset of the clocks in r, other values unchanged,
- for a duration d, valuation $v + d$ is obtained by adding d to all clock values.

Geometric view with two clocks x et y



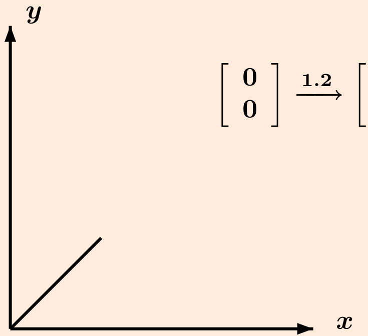
Semantics of timed automata (1)

Operations on valuations

X set of clocks. For valuation v :

- for a subset r of X, valuation $v[r \mapsto 0]$ is obtained by reset of the clocks in r, other values unchanged,
- for a duration d, valuation $v + d$ is obtained by adding d to all clock values.

Geometric view with two clocks x et y



$$\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \xrightarrow{\mathbf{1.2}} \begin{bmatrix} \mathbf{1.2} \\ \mathbf{1.2} \end{bmatrix}$$

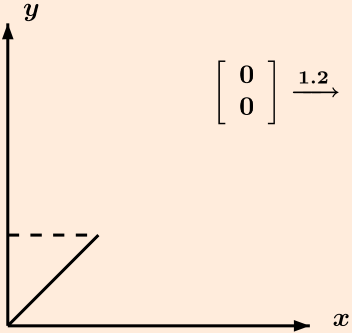
Semantics of timed automata (1)

Operations on valuations

X set of clocks. For valuation v :

- for a subset r of X, valuation $v[r \mapsto 0]$ is obtained by reset of the clocks in r, other values unchanged,
- for a duration d, valuation $v + d$ is obtained by adding d to all clock values.

Geometric view with two clocks x et y



$$\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \xrightarrow{\mathbf{1.2}} \begin{bmatrix} \mathbf{1.2} \\ \mathbf{1.2} \end{bmatrix} \xrightarrow{\boldsymbol{x:=0}} \begin{bmatrix} \mathbf{0} \\ \mathbf{1.2} \end{bmatrix}$$

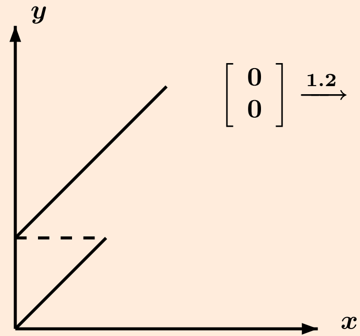
Semantics of timed automata (1)

Operations on valuations

X set of clocks. For valuation v :

- for a subset r of X, valuation $v[r \mapsto 0]$ is obtained by reset of the clocks in r, other values unchanged,
- for a duration d, valuation $v + d$ is obtained by adding d to all clock values.

Geometric view with two clocks x et y



$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \xrightarrow{1.2} \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix} \xrightarrow{x:=0} \begin{bmatrix} 0 \\ 1.2 \end{bmatrix} \xrightarrow{2} \begin{bmatrix} 2 \\ 3.2 \end{bmatrix}$$

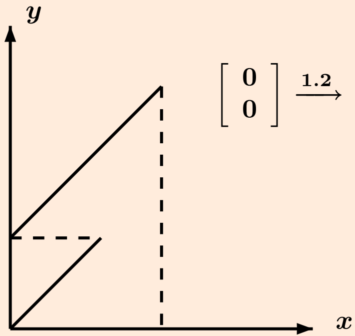
Semantics of timed automata (1)

Operations on valuations

X set of clocks. For valuation v :

- for a subset r of X, valuation $v[r \mapsto 0]$ is obtained by reset of the clocks in r, other values unchanged,
- for a duration d, valuation $v + d$ is obtained by adding d to all clock values.

Geometric view with two clocks x et y



$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \xrightarrow{1.2} \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix} \xrightarrow{x:=0} \begin{bmatrix} 0 \\ 1.2 \end{bmatrix} \xrightarrow{2} \begin{bmatrix} 2 \\ 3.2 \end{bmatrix} \xrightarrow{y:=0} \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

Semantics of timed automata (2)

Definition

For a timed automaton $\mathcal{A} = (Q, q_0, Inv, \Delta)$, the transition system is $\mathcal{T} = (S, s_0, E)$ with:

- the set of configurations $S = \{(q, v) \in Q \times \mathbb{R}_{\geq 0} \mid v \models Inv(q)\}$,
- initial configuration $s_0 = (q_0, \mathbf{0})$,
- action transitions: $\boldsymbol{(q, v) \xrightarrow{a} (q', v')}$, if there exists a transition $q \xrightarrow{g,a,r} q'$ from $\mathcal{A}$ such that $v \models g$ and $v' \models Inv(q')$, with $v' = v[r \mapsto 0]$,
- delay transitions $\boldsymbol{(q, v) \xrightarrow{d} (q, v + d)}$ if $v + d \models Inv(q)$.

Semantics of timed automata (2)

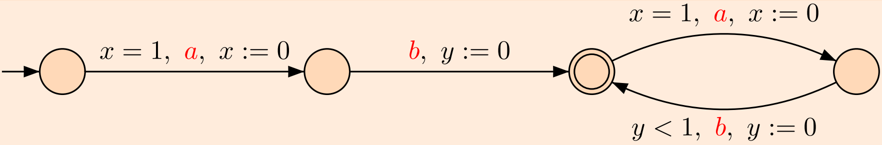
Definition

For a timed automaton $\mathcal{A} = (Q, q_0, Inv, \Delta)$, the transition system is $\mathcal{T} = (S, s_0, E)$ with:

- the set of configurations $S = \{(q, v) \in Q \times \mathbb{R}_{\geq 0} \mid v \models Inv(q)\}$,
- initial configuration $s_0 = (q_0, \mathbf{0})$,
- action transitions: $\boldsymbol{(q, v) \xrightarrow{a} (q', v')}$, if there exists a transition $q \xrightarrow{g,a,r} q'$ from $\mathcal{A}$ such that $v \models g$ and $v' \models Inv(q')$, with $v' = v[r \mapsto 0]$,
- delay transitions $\boldsymbol{(q, v) \xrightarrow{d} (q, v + d)}$ if $v + d \models Inv(q)$.

Discrete vs dense time (revisited)

[Alur Dill 1994]

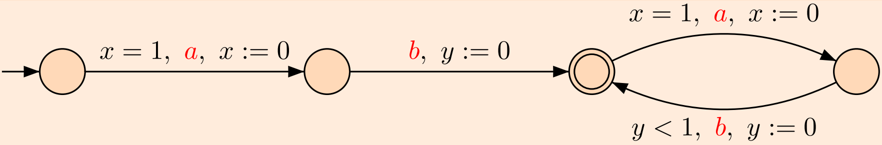


- Dense-time
  The infinite observation $(a,1)(b,2)(a,2)(b,2.9)(a,3)(3.8)(a,4)(b,4.7)\ldots$
  is in L_{dense}

- Discrete-time
  $L_{disc} = \emptyset$ no infinite observation whatever the granularity choice

Discrete vs dense time (revisited)

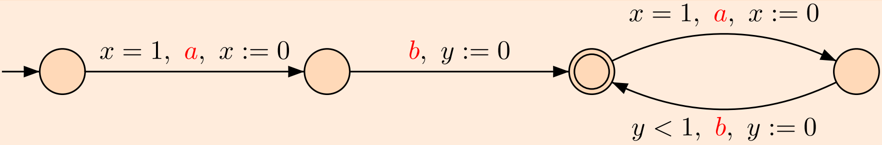
[Alur Dill 1994]



- Dense-time
  The infinite observation $(a,1)(b,2)(a,2)(b,2.9)(a,3)(3.8)(a,4)(b,4.7)\ldots$
  is in L_{dense}
- Discrete-time
  $L_{disc} = \emptyset$ no infinite observation whatever the granularity choice

Discrete vs dense time (revisited)

[Alur Dill 1994]

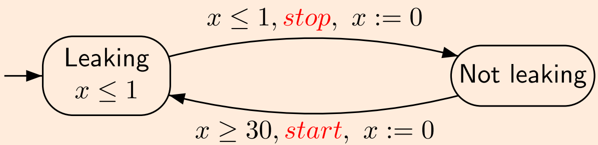


- Dense-time
  The infinite observation $(a,1)(b,2)(a,2)(b,2.9)(a,3)(3.8)(a,4)(b,4.7)\ldots$ is in L_{dense}
- Discrete-time
  $L_{disc} = \emptyset$ no infinite observation whatever the granularity choice

The gas burner (revisited)

as a timed automaton

- each time a leakage is detected, it is repaired or stopped in less than 1s
- two leakages are separated by at least 30s

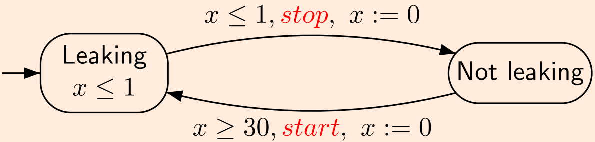


Not expressive enough for the property: Is it possible that the gas burner leaks during a time greater than $\frac{1}{20}$ of the global time after the first 60s?

The gas burner (revisited)

as a timed automaton

- each time a leakage is detected, it is repaired or stopped in less than 1s
- two leakages are separated by at least 30s



Not expressive enough for the property: Is it possible that the gas burner leaks during a time greater than $\frac{1}{20}$ of the global time after the first 60s?

Timed logics

Temporal logics

A request is always granted

in Computational Tree Logic CTL

$$\mathsf{AG}(\mathsf{request} \Rightarrow \mathsf{AF}\,\mathsf{grant})$$

How to express:

A request is always granted in less than 5 time units

CTL + time: TCTL [Alur Henzinger 1991]

$$\varphi, \psi ::= P \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathsf{E}\varphi\mathsf{U}_{\bowtie c}\psi \mid \mathsf{A}\varphi\mathsf{U}_{\bowtie c}\psi$$

P an atomic proposition, c a constant and $\bowtie$ an operator in $\{<, >, \leq, \geq, =\}$.

In TCTL

$$\mathsf{AG}(\mathsf{request} \Rightarrow \mathsf{AF}_{\leq 5}\,\mathsf{grant})$$

Timed logics

Temporal logics

A request is always granted

in Computational Tree Logic CTL

$$\mathsf{AG}(\text{request} \Rightarrow \mathsf{AF}\,\text{grant})$$

How to express:

A request is always granted in less than 5 time units

CTL + time: TCTL [Alur Henzinger 1991]

$$\varphi, \psi ::= P \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathsf{E}\varphi\mathsf{U}_{\bowtie c}\psi \mid \mathsf{A}\varphi\mathsf{U}_{\bowtie c}\psi$$

P an atomic proposition, c a constant and $\bowtie$ an operator in $\{<, >, \leq, \geq, =\}$.

In TCTL

$$\mathsf{AG}(\text{request} \Rightarrow \mathsf{AF}_{\leq 5}\,\text{grant})$$

Timed logics

Temporal logics

A request is always granted

in Computational Tree Logic CTL

$$\mathsf{AG}(\text{request} \Rightarrow \mathsf{AF}\,\text{grant})$$

How to express:

A request is always granted in less than 5 time units

CTL + time: TCTL [Alur Henzinger 1991]

$$\varphi, \psi ::= P \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathsf{E}\varphi\mathsf{U}_{\bowtie c}\psi \mid \mathsf{A}\varphi\mathsf{U}_{\bowtie c}\psi$$

P an atomic proposition, c a constant and $\bowtie$ an operator in $\{<, >, \leq, \geq, =\}$.

In TCTL

$$\mathsf{AG}(\text{request} \Rightarrow \mathsf{AF}_{\leq 5}\,\text{grant})$$

Timed logics

Temporal logics

A request is always granted

in Computational Tree Logic CTL

$$\mathsf{AG}(\text{request} \Rightarrow \mathsf{AF}\,\text{grant})$$

How to express:

A request is always granted in less than 5 time units

CTL + time: TCTL [Alur Henzinger 1991]

$$\varphi, \psi ::= P \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathsf{E}\varphi\mathsf{U}_{\bowtie c}\psi \mid \mathsf{A}\varphi\mathsf{U}_{\bowtie c}\psi$$

P an atomic proposition, c a constant and $\bowtie$ an operator in $\{<, >, \leq, \geq, =\}$.

In TCTL

$$\mathsf{AG}(\text{request} \Rightarrow \mathsf{AF}_{\leq 5}\,\text{grant})$$

Timed logics

Temporal logics

A request is always granted

in Computational Tree Logic CTL

$$\mathsf{AG}(\text{request} \Rightarrow \mathsf{AF}\,\text{grant})$$

How to express:

A request is always granted in less than 5 time units

CTL + time: TCTL [Alur Henzinger 1991]

$$\varphi, \psi ::= P \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathsf{E}\varphi\mathsf{U}_{\bowtie c}\psi \mid \mathsf{A}\varphi\mathsf{U}_{\bowtie c}\psi$$

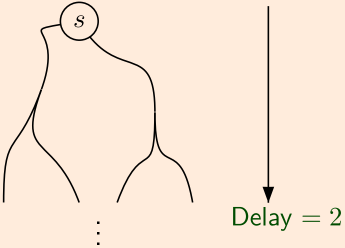
P an atomic proposition, c a constant and $\bowtie$ an operator in $\{<, >, \leq, \geq, =\}$.

In TCTL

$$\mathsf{AG}(\text{request} \Rightarrow \mathsf{AF}_{\leq 5}\,\text{grant})$$

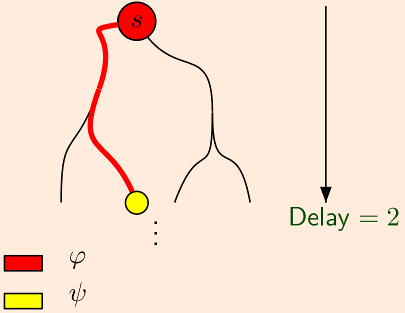
Interpretation

A formula is interpreted on a configuration of a TTS



Interpretation

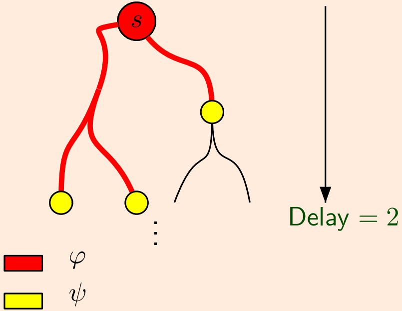
A formula is interpreted on a configuration of a TTS



$s \models \mathsf{E}\varphi\mathsf{U}_{\leq 2}\psi$

Interpretation

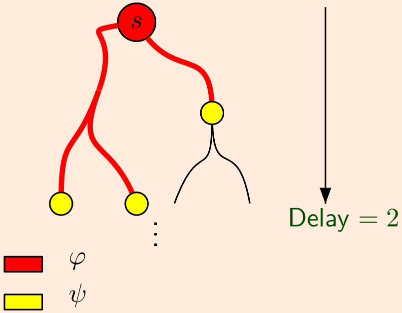
A formula is interpreted on a configuration of a TTS



$$s \models \mathsf{A}\varphi\mathsf{U}_{\leq 2}\psi$$

Interpretation

A formula is interpreted on a configuration of a TTS



$$s \models \mathsf{A}\varphi\mathsf{U}_{\leq 2}\psi$$

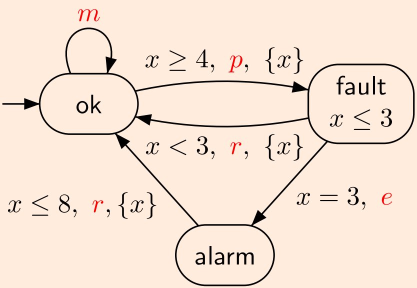
Abbreviations

$\mathsf{AF}_{\bowtie c}\psi$ means $\mathsf{A}\ true\ \mathsf{U}_{\bowtie c}\psi$

$\mathsf{EF}_{\bowtie c}\psi$ means $\mathsf{E}\ true\ \mathsf{U}_{\bowtie c}\psi$

$\mathsf{AG}_{\bowtie c}\psi$ means $\neg\mathsf{EF}_{\bowtie c}(\neg\varphi)$

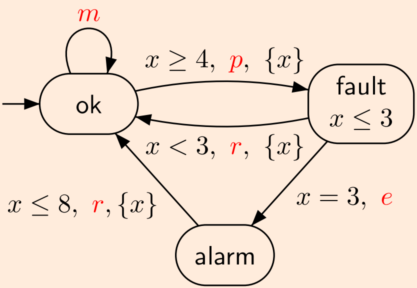
Example for a timed automaton



initial state ok satisfies:

$$\mathsf{AG}(\mathsf{fault} \Rightarrow \mathsf{AF}_{\leq 8}\, \mathsf{ok})$$

Example for a timed automaton



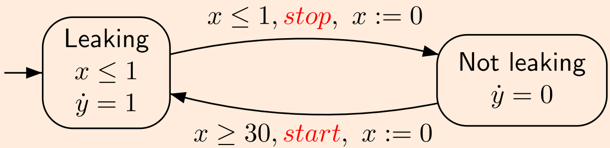
initial state ok satisfies:

$$\mathsf{AG}(\mathsf{fault} \Rightarrow \mathsf{AF}_{\leq 8}\, \mathsf{ok})$$

Other logics

Back again to the gas burner

as a linear hybrid automaton



Add a stopwatch y and a clock z which are never reset

and use these variables in a CTL formula:

$$\mathsf{AG}(z \geq 60 \Rightarrow 20y \leq z)$$

Timed logics for linear time

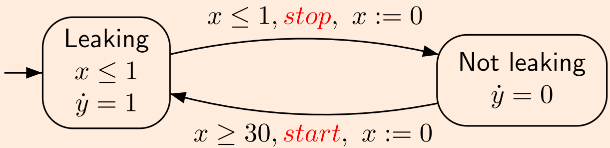
Extensions of Linear Temporal Logic LTL

- with intervals as subscript: MTL, with non singular intervals: MITL,
- with clocks in formulas...

Other logics

Back again to the gas burner

as a linear hybrid automaton



Add a stopwatch y and a clock z which are never reset

and use these variables in a CTL formula:

$$\mathsf{AG}(z \geq 60 \Rightarrow 20y \leq z)$$

Timed logics for linear time

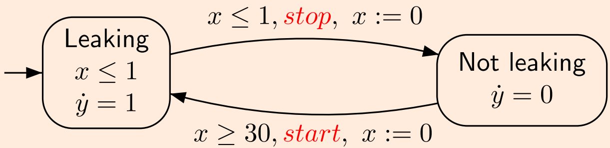
Extensions of Linear Temporal Logic LTL

- with intervals as subscript: MTL, with non singular intervals: MITL,
- with clocks in formulas...

Other logics

Back again to the gas burner

as a linear hybrid automaton



Add a stopwatch y and a clock z which are never reset

and use these variables in a CTL formula:

$$\mathsf{AG}(z \geq 60 \Rightarrow 20y \leq z)$$

Timed logics for linear time

Extensions of Linear Temporal Logic LTL

- with intervals as subscript: MTL, with non singular intervals: MITL,
- with clocks in formulas...

Outline

Timed Models

Verification

Applications

Conclusion

Reachability

Deciding reachability of a control state reduces to decide emptiness.

Theorem [Alur Dill 1990]

The emptiness problem for timed automata is PSPACE-complete.

Decision procedure

Input: a timed automaton $\mathcal{A} = (Q, q_0, Inv, \Delta)$ on a set X of real valued clocks

- Construction of a (Büchi) standard automaton $\mathcal{H}$, such that:
  no execution possible in $\mathcal{A} \Leftrightarrow$ no execution possible in $\mathcal{H}$
- Emptiness test for $\mathcal{H}$.

Reachability

Deciding reachability of a control state reduces to decide emptiness.

Theorem [Alur Dill 1990]

The emptiness problem for timed automata is PSPACE-complete.

Decision procedure

Input: a timed automaton $\mathcal{A} = (Q, q_0, Inv, \Delta)$ on a set X of real valued clocks

- Construction of a (Büchi) standard automaton $\mathcal{H}$, such that:
  no execution possible in $\mathcal{A} \Leftrightarrow$ no execution possible in $\mathcal{H}$
- Emptiness test for $\mathcal{H}$.

Reachability

Deciding reachability of a control state reduces to decide emptiness.

Theorem [Alur Dill 1990]

The emptiness problem for timed automata is PSPACE-complete.

Decision procedure

Input: a timed automaton $\mathcal{A} = (Q, q_0, Inv, \Delta)$ on a set X of real valued clocks

- Construction of a (Büchi) standard automaton $\mathcal{H}$, such that:
  no execution possible in $\mathcal{A} \Leftrightarrow$ no execution possible in $\mathcal{H}$
- Emptiness test for $\mathcal{H}$.

$\mathcal{T} = (S, s_0, E)$

transition system of $\mathcal{A}$

configurations: (q, v)

$q \in Q$, $v \in \mathbb{R}^X_{\geq 0}$

$\xrightarrow{quotient}$

$\mathcal{H}$

region automaton of $\mathcal{A}$

states: $(q, [v])$

$q \in Q$, $[v]$ equivalence class

for some relation $\sim$ on $\mathbb{R}^X_{\geq 0}$

Quotient construction (1)

with the following properties:

For two equivalent valuations $v \sim v'$

1. if an action transition $q \xrightarrow{g,a,r} q'$ is possible from v, then the same transition is possible from v' and the resulting valuations $v[r \mapsto 0]$ et $v'[r \mapsto 0]$ are equivalent,
2. if a delay transition of $\boldsymbol{d}$ is possible from v, then a delay transition of $\boldsymbol{d'}$ is possible from v' and the resulting valuations $v + d$ et $v' + d'$ are equivalent.

Remarks

- Relation $\sim$ produces a time-abstract bisimulation between configurations (q, v) of $\mathcal{T}$ and states $(q, [v])$ of $\mathcal{H}$.
- For the first condition, it is enough to consider constraints $x \bowtie k$, for clocks in X et constants $0 \leq k \leq m$, where m is the maximal constant in the constraints of $\mathcal{A}$.

Quotient construction (1)

with the following properties:

For two equivalent valuations $v \sim v'$

1. if an action transition $q \xrightarrow{g,a,r} q'$ is possible from v, then the same transition is possible from v' and the resulting valuations $v[r \mapsto 0]$ et $v'[r \mapsto 0]$ are equivalent,
2. if a delay transition of $\boldsymbol{d}$ is possible from v, then a delay transition of $\boldsymbol{d'}$ is possible from v' and the resulting valuations $v + d$ et $v' + d'$ are equivalent.

Remarks

- Relation $\sim$ produces a time-abstract bisimulation between configurations (q, v) of $\mathcal{T}$ and states $(q, [v])$ of $\mathcal{H}$.
- For the first condition, it is enough to consider constraints $x \bowtie k$, for clocks in X et constants $0 \leq k \leq m$, where m is the maximal constant in the constraints of $\mathcal{A}$.

Quotient construction (1)

with the following properties:

For two equivalent valuations $v \sim v'$

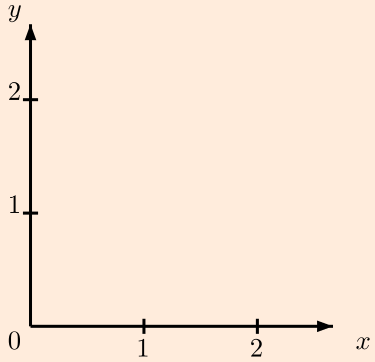
1. if an action transition $q \xrightarrow{g,a,r} q'$ is possible from v, then the same transition is possible from v' and the resulting valuations $v[r \mapsto 0]$ et $v'[r \mapsto 0]$ are equivalent,
2. if a delay transition of $\boldsymbol{d}$ is possible from v, then a delay transition of $\boldsymbol{d'}$ is possible from v' and the resulting valuations $v + d$ et $v' + d'$ are equivalent.

Remarks

- Relation $\sim$ produces a time-abstract bisimulation between configurations (q, v) of $\mathcal{T}$ and states $(q, [v])$ of $\mathcal{H}$.
- For the first condition, it is enough to consider constraints $x \bowtie k$, for clocks in X et constants $0 \leq k \leq m$, where m is the maximal constant in the constraints of $\mathcal{A}$.

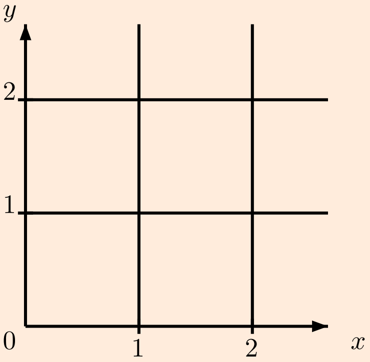
Quotient construction (2)

Geometric view with two clocks x and y, for $m = 2$



Quotient construction (2)

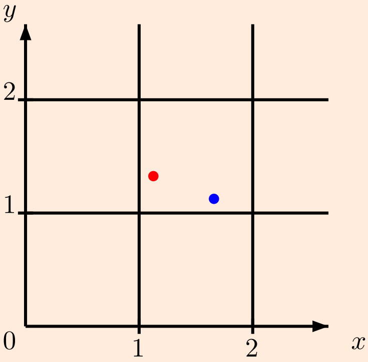
Geometric view with two clocks x and y, for $m = 2$



- Equivalent valuations satisfy the same constraints $x \bowtie k$

Quotient construction (2)

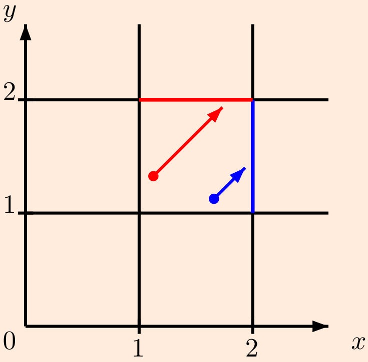
Geometric view with two clocks x and y, for $m = 2$



- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

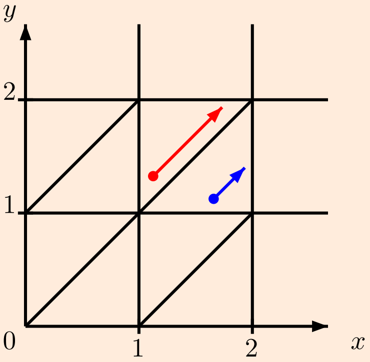
Geometric view with two clocks x and y, for $m = 2$



- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

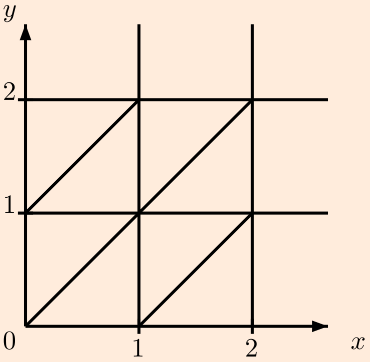
Geometric view with two clocks x and y, for $m = 2$



- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

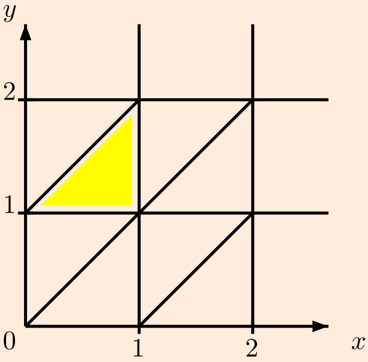
Geometric view with two clocks x and y, for $m = 2$



- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

Geometric view with two clocks x and y, for $m = 2$



region R defined by

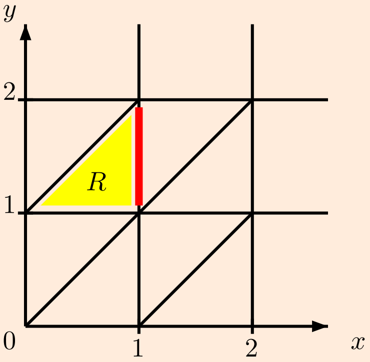
$I_x =]0;1[$, $I_y =]1;2[$

$frac(x) > frac(y)$

- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

Geometric view with two clocks x and y, for $m = 2$



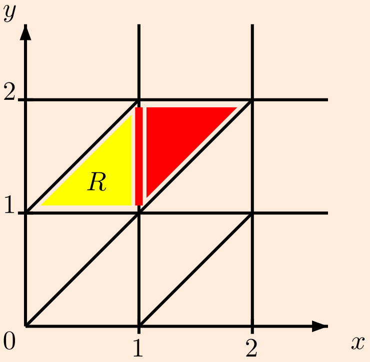
region R defined by
$I_x =]0;1[$, $I_y =]1;2[$
$frac(x) > frac(y)$

Time successor of R
$I_x = [1;1]$, $I_y =]1;2[$

- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

Geometric view with two clocks x and y, for $m = 2$



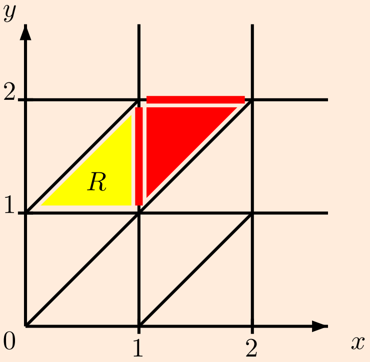
region R defined by
$I_x =]0;1[$, $I_y =]1;2[$
$frac(x) > frac(y)$

Time successor of R
$I_x = [1;1]$, $I_y =]1;2[$

- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

Geometric view with two clocks x and y, for $m = 2$



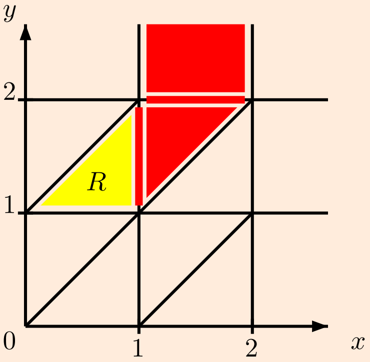
region R defined by
$I_x =]0;1[$, $I_y =]1;2[$
$frac(x) > frac(y)$

Time successor of R
$I_x = [1;1]$, $I_y =]1;2[$

- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

Geometric view with two clocks x and y, for $m = 2$



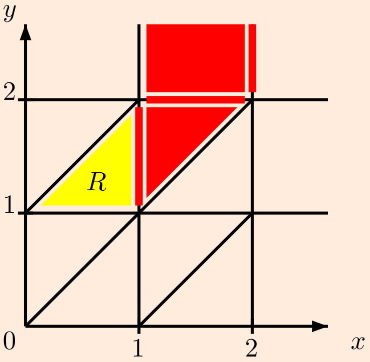
region R defined by
$I_x =]0;1[$, $I_y =]1;2[$
$frac(x) > frac(y)$

Time successor of R
$I_x = [1;1]$, $I_y =]1;2[$

- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

Geometric view with two clocks x and y, for $m = 2$



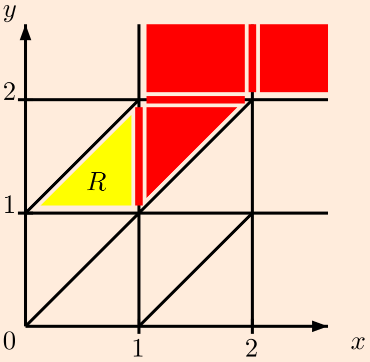
region R defined by
$I_x =]0;1[$, $I_y =]1;2[$
$frac(x) > frac(y)$

Time successor of R
$I_x = [1;1]$, $I_y =]1;2[$

- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

Geometric view with two clocks x and y, for $m = 2$



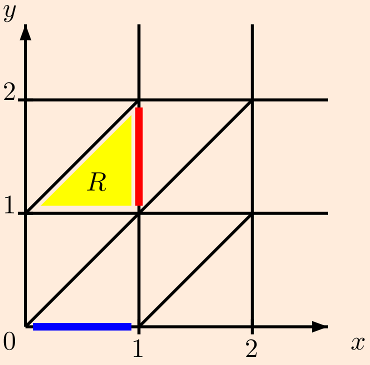
region R defined by
$I_x =]0;1[$, $I_y =]1;2[$
$frac(x) > frac(y)$

Time successor of R
$I_x = [1;1]$, $I_y =]1;2[$

- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (2)

Geometric view with two clocks x and y, for $m = 2$



region R defined by

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

$frac(x) > frac(y)$

Time successor of R

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

Action successor of R

with $y := 0$

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

- Equivalent valuations satisfy the same constraints $x \bowtie k$
- Equivalent valuations respect time elapsing

Quotient construction (3)

Region automaton $\mathcal{H}$

For timed automaton $\mathcal{A} = (Q, q_0, Inv, \Delta)$,
with set of clocks X, maximal constant m and quotient $\mathcal{R} = \mathbb{R}^X_{\geq 0}/\sim$,

- states $Q \times \mathcal{R}$
- (abstract) delay transitions: $(q, R) \xrightarrow{\leq} (q, succ(R))$
- action transitions: $(q, R) \xrightarrow{a} (q', R')$
  if there exists a transition $q \xrightarrow{g,a,r} q'$ from $\mathcal{A}$ such that $R \models g$ and $R' = R[r \mapsto 0]$

Quotient size

The size of $\mathcal{R}$ is $\mathcal{O}(|X|! \cdot m^{|X|})$, to be multiplied by $|\ Q\ |$.

Quotient construction (3)

Region automaton $\mathcal{H}$

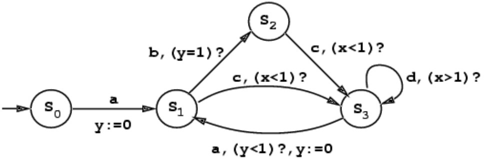
For timed automaton $\mathcal{A} = (Q, q_0, Inv, \Delta)$,
with set of clocks X, maximal constant m and quotient $\mathcal{R} = \mathbb{R}^X_{\geq 0}/\sim$,

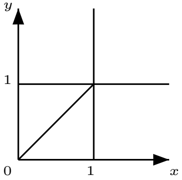
- states $Q \times \mathcal{R}$
- (abstract) delay transitions: $(q, R) \xrightarrow{\leq} (q, succ(R))$
- action transitions: $(q, R) \xrightarrow{a} (q', R')$
  if there exists a transition $q \xrightarrow{g,a,r} q'$ from $\mathcal{A}$ such that $R \models g$ and $R' = R[r \mapsto 0]$

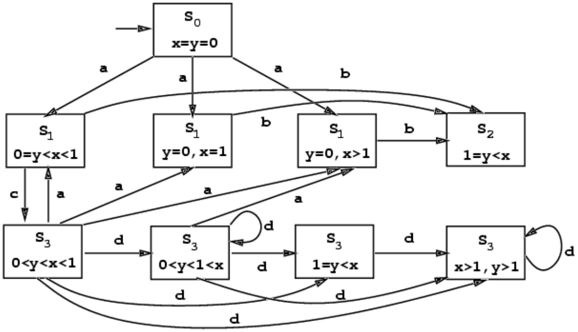
Quotient size

The size of $\mathcal{R}$ is $\mathcal{O}(|X|! \cdot m^{|X|})$, to be multiplied by $\mid Q \mid$.

Example [Alur Dill 1990]







Other results

Complexity is higher than for untimed models

- The model-checking problem for TCTL on timed automata is PSPACE-complete [Alur et al. 1993] .
- The model-checking problem for MITL on timed automata is EXPSPACE-complete [Alur et al. 1996].

and sometimes worse:

The model-checking problem for MTL on timed automata is undecidable [Henzinger 1991].

Some efficient algorithms

by restriction: for the logic $\mathsf{TCTL}_{\leq,\geq}$ (without equality)

- for automata with duration and discrete time, model-checking is in polynomial time ($|\mathcal{A}| \cdot |\varphi|$) [Laroussinie et al. 2002].
- for timed automata with a single clock, model-checking is P-complete [Laroussinie et al. 2004].

Other results

Complexity is higher than for untimed models

- The model-checking problem for TCTL on timed automata is PSPACE-complete [Alur et al. 1993] .
- The model-checking problem for MITL on timed automata is EXPSPACE-complete [Alur et al. 1996].

and sometimes worse:

The model-checking problem for MTL on timed automata is undecidable [Henzinger 1991].

Some efficient algorithms

by restriction: for the logic $\mathsf{TCTL}_{\leq,\geq}$ (without equality)

- for automata with duration and discrete time, model-checking is in polynomial time ($|\mathcal{A}| \cdot |\varphi|$) [Laroussinie et al. 2002].
- for timed automata with a single clock, model-checking is P-complete [Laroussinie et al. 2004].

Other results

Complexity is higher than for untimed models

- The model-checking problem for TCTL on timed automata is PSPACE-complete [Alur et al. 1993] .
- The model-checking problem for MITL on timed automata is EXPSPACE-complete [Alur et al. 1996].

and sometimes worse:

The model-checking problem for MTL on timed automata is undecidable [Henzinger 1991].

Some efficient algorithms

by restriction: for the logic $\mathrm{TCTL}_{\leq,\geq}$ (without equality)

- for automata with duration and discrete time, model-checking is in polynomial time $(|\mathcal{A}| \cdot |\varphi|)$ [Laroussinie et al. 2002].
- for timed automata with a single clock, model-checking is P-complete [Laroussinie et al. 2004].

Verification in practice

Several tools

have been developed and applied to case studies, in spite of the complexity:

- KRONOS and UPPAAL for timed automata
- HCMC and HYTECH for linear hybrid automata (semi-algorithms)
- TSMV for automata with duration (discrete time)
- Romeo and TINA, for time Petri nets
- ...

using specific data structures

- for the representation of regions or zones: DBM (Difference Bounded Matrices) and variations (CDD, NDD, etc.)
- for the representation of polyedras

and heuristics for the algorithms

- on the fly analysis
- compositional methods
- constraint solving

Verification in practice

Several tools

have been developed and applied to case studies, in spite of the complexity:

- KRONOS and UPPAAL for timed automata
- HCMC and HYTECH for linear hybrid automata (semi-algorithms)
- TSMV for automata with duration (discrete time)
- Romeo and TINA, for time Petri nets
- ...

using specific data structures

- for the representation of regions or zones: DBM (Difference Bounded Matrices) and variations (CDD, NDD, etc.)
- for the representation of polyedras

and heuristics for the algorithms

- on the fly analysis
- compositional methods
- constraint solving

Verification in practice

Several tools

have been developed and applied to case studies, in spite of the complexity:

- KRONOS and UPPAAL for timed automata
- HCMC and HYTECH for linear hybrid automata (semi-algorithms)
- TSMV for automata with duration (discrete time)
- Romeo and TINA, for time Petri nets
- ...

using specific data structures

- for the representation of regions or zones: DBM (Difference Bounded Matrices) and variations (CDD, NDD, etc.)
- for the representation of polyedras

and heuristics for the algorithms

- on the fly analysis
- compositional methods
- constraint solving

Outline

Timed Models

Verification

Applications

Conclusion

Many experiments

in the areas of

- communication protocols
- programmable logic controllers (PLCs)
- etc.

Example: Mecatronic Standard System (MSS) platform from Bosch Group [BBGRS05], joint work with LURPA, ENS Cachan

Many experiments

in the areas of

- communication protocols
- programmable logic controllers (PLCs)
- etc.

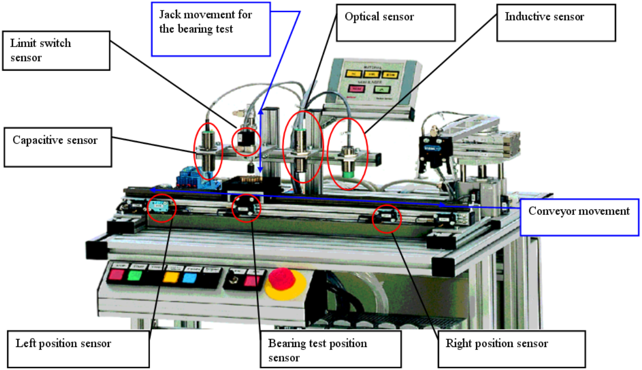
Example: Mecatronic Standard System (MSS) platform from Bosch Group [BBGRS05], joint work with LURPA, ENS Cachan

Many experiments

in the areas of

- communication protocols
- programmable logic controllers (PLCs)
- etc.

Example: Mecatronic Standard System (MSS) platform from Bosch Group [BBGRS05], joint work with LURPA, ENS Cachan



Presentation of MSS station 2

- Work-pieces are transported by a linear conveyor
- They are tested by a jack for the presence or absence of a bearing (inside)
- and by sensors to determine their material

The system is controlled by a program, in two versions: with an event-driven task, triggered when the testing position is reached, or without it.

Requirement

The conveyor arrives at the bearing test position with a high speed (200 mm/s) and it must react to the stopping order in less than 5ms.

P: the conveyor stops in less than 5 ms at the bearing test position.

Presentation of MSS station 2

- Work-pieces are transported by a linear conveyor
- They are tested by a jack for the presence or absence of a bearing (inside)
- and by sensors to determine their material

The system is controlled by a program, in two versions: with an event-driven task, triggered when the testing position is reached, or without it.

Requirement

The conveyor arrives at the bearing test position with a high speed (200 mm/s) and it must react to the stopping order in less than 5ms.

P: the conveyor stops in less than 5 ms at the bearing test position.

Modeling MSS station 2 (1)

with Uppaal

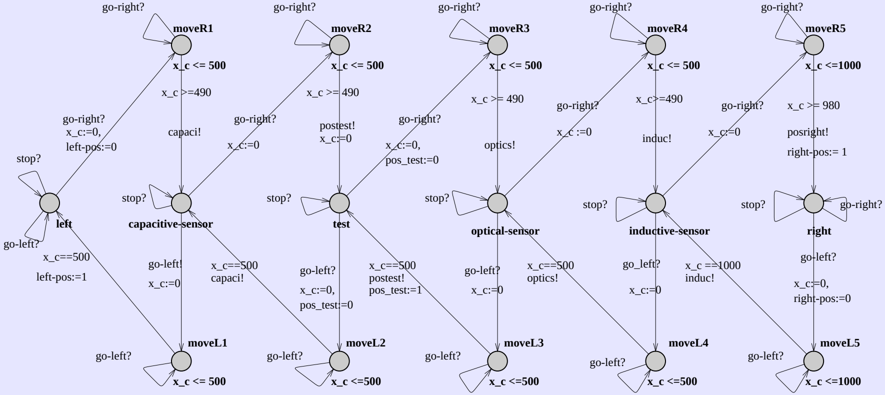
as a network of timed automata, handling clocks and discrete variables and communicating through binary and broadcast channels.
The conveyor:

Modeling MSS station 2 (1)

with Uppaal

as a network of timed automata, handling clocks and discrete variables and communicating through binary and broadcast channels.

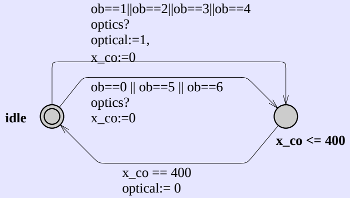
The conveyor:

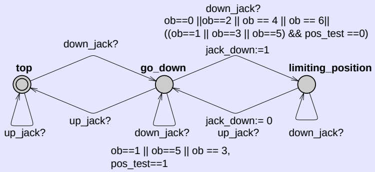


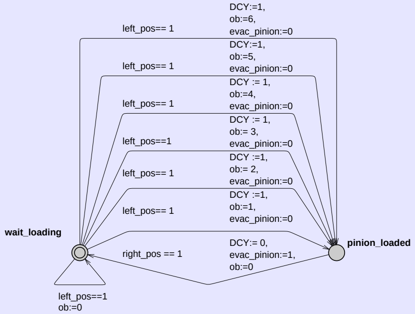
Modeling station 2 of the platform (2)

other elements

An optical sensor, the jack and the environment (abstracted):







Modeling the control program (1)

written in *Ladder Diagram* (IEC 61131-3)

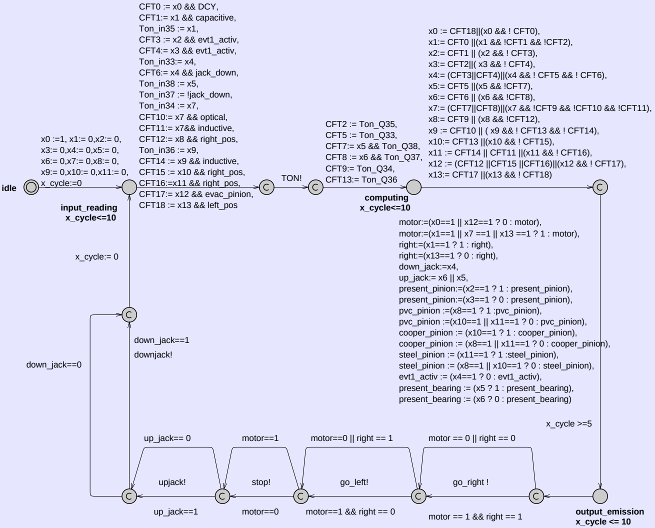
```
|     a      b      ST     |
+---| |---| |---(   )---+
|     c             MA     |
+---|/|---+-----(   )---+
|     d    |               |
+---| |---+                |
|                          |
```

means

```
ST := a and b
MA := not(c) or d
```

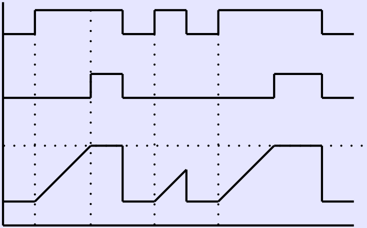
Modeling the control program (2)

in UPPAAL



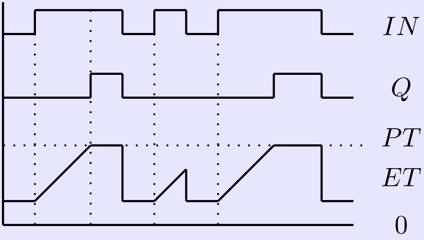
Time in PLCs

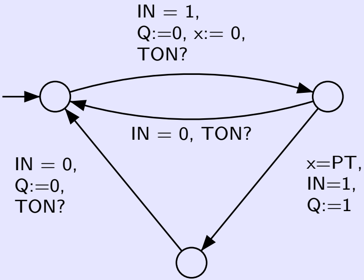
Timer On Delay (TON)



Time in PLCs

Timer On Delay (TON)





Results

Verification uses an observer automaton with clock X, reset when the signal is sent and tested when the conveyor stops.

property	result	time	memory
with the event driven task			
C1:E<> obs.stop and $X > 5$	yes	15 s	30 Mb
C2:E<> obs.stop and $X \leq 5$	yes	15 s	30 Mb
C3:E<> obs.stop and $X > 10$	no	22 s	61 Mb
without the event driven task			
C5:E<> obs.stop and $X \geq 10$	yes	16 s	30 Mb
C6:E<> obs.stop and $X > 20$	no	22 s	70 Mb
C7:E<> obs.stop and $X < 10$	no	22 s	69 Mb
with Mader-Wupper model			
C8:E<> obs.stop and $X > 5$	-	$>$29 h	-

Linux machine, pentium4 at 2.4 GHz with 3 Gb RAM

- Multitask programming reduces the reaction time from two to one cycle time.
- However, C1 proves that it is not sufficient to satisfy requirement **P**.

Performances (14 automata, 11 clocks, 30.10^6 states) are due to an atomicity hypothesis in the control program and enhanced model of the TON block.

Outline

Timed Models

Verification

Applications

Conclusion

Conclusion

Many works in this area

- for other models and other logics
- for quantitative extensions with weights, costs, probabilities, etc.
- relating control problems with game theory

Perspectives

Theoretical: refine the limits for decidability questions
Practical : deal with the combinatorial explosion problem

- specifications and models fitting particular settings, with simpler and more efficient algorithms
- data structures for the combination of discrete and continuous features
- abstraction methods

Conclusion

Many works in this area

- for other models and other logics
- for quantitative extensions with weights, costs, probabilities, etc.
- relating control problems with game theory

Perspectives

Theoretical: refine the limits for decidability questions
Practical : deal with the combinatorial explosion problem

- specifications and models fitting particular settings, with simpler and more efficient algorithms
- data structures for the combination of discrete and continuous features
- abstraction methods

Thank you

Bibliography

[ACHH93] Alur, Courcoubetis, Henzinger, Ho. **Hybrid Automata: an Algorithmic Approach to Specification and Verification of Hybrid Systems.** Hybrid Systems I (LNCS 736).

[Alur91] Alur. **Techniques for Automatic Verification of Real-Time Systems.** PhD Thesis, 1991.

[BS91] Brzozowski, Seger. **Advances in Asynchronous Circuit Theory**. BEATCS, 1991.

[Merlin74] Merlin. **A Study of the Recoverability of Computing Systems.** PhD Thesis, 1974.

[EMSS92] Emerson, Mok, Sistla, Srinivasan. **Quantitative Temporal Reasoning.** Real-Time Systems 4(4), 1992.

[AD90] **Automata for Modeling Real-Time Systems.** ICALP'90 (LNCS 443).

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

[AH91] Alur, Henzinger. **Logics and models of real time: a survey.** Real-time: Theory in practice (LNCS 600).

[ACD93] Alur, Courcoubetis, Dill. **Model-Checking in Dense Real-Time.** Information and Computation 104(1), 1993.

[AFH96] Alur, Feder, Henzinger. **The Benefits of Relaxing Punctuality.** JACM 43(1), 1996.

[Henzinger91] Henzinger. **The temporal specification and verification of real-time systems.** PhD Thesis, 1991.

[LMP06] Laroussinie, Markey, Schnoebelen. **Efficient timed model checking for discrete time systems.** TCS 353(1-3), 2006.

[LMP04] Laroussinie, Markey, Schnoebelen. **Model checking timed automata with one or two clocks.** CONCUR'04 (LNCS 3170).

[BBGRS05] Bel mokadem, Bérard, Gourcuff, Roussel, de Smet. **Verification of a timed multitask system with Uppaal.** ETFA'05, IEEE, 2005.