Timed Automata

Nathalie Bertrand
nathalie.bertrand@inria.fr

VerTeCs, INRIA Rennes

October 4th & 11th 2009

Outline

Formal methods for verification

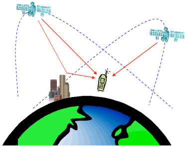
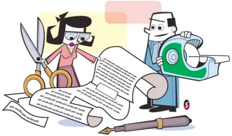
Model-based testing : automatically generate a set of testing scenarios, given mathematical representations for system under test and specification.

Static analysis : analyze properties of source code in a static manner, *i.e.* without unfolding all possible behaviours.

Automated proof : (partially automatically) prove correctness of a program through a logical reasoning using deduction rules.

Model checking : automatically prove that mathematical representation for the system satisfies model for the specification.

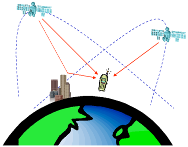
Principles of model checking

Does  satisfy  ?

system specification

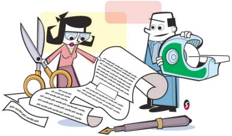
Principles of model checking

Does



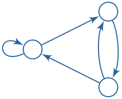
system

satisfy



specification

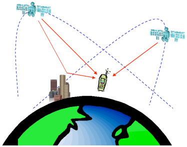
?



model

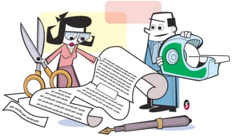
Principles of model checking

Does



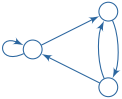
system

satisfy



specification

?

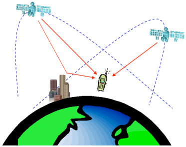
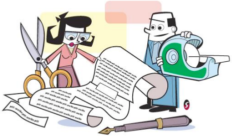


model

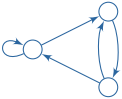
φ

formula

Principles of model checking

Does  satisfy  ?

system specification

 $\models$ φ ?

model-checker

model formula

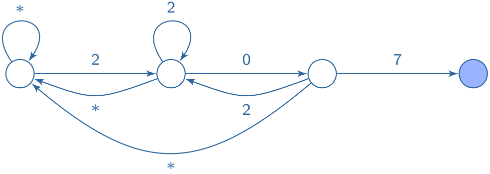
Models for systems

Systems under analysis are represented by transition systems.

- finite automata
- pushdown automata
- counter automata
- timed automata
- hybrid automata
- Petri nets
- channel systems
- message sequence charts
- ...

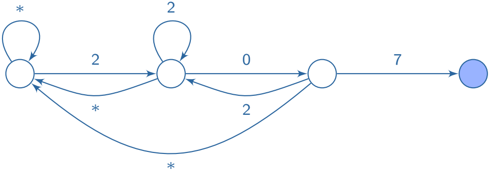
Examples of models

- A numerical code door lock

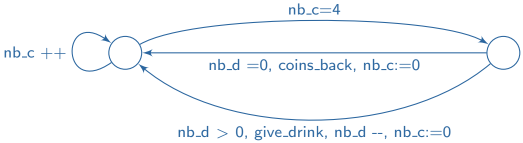


Examples of models

- A numerical code door lock

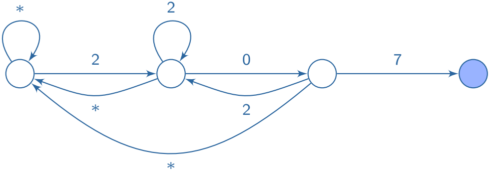


- A vending machine

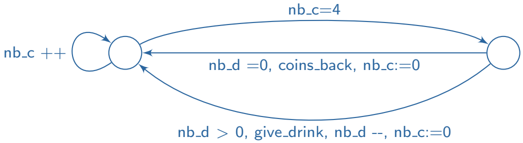


Examples of models

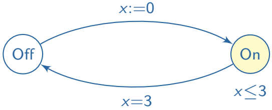
- A numerical code door lock



- A vending machine



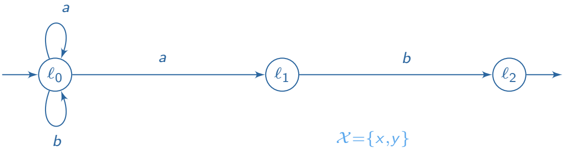
- A time-switch



Outline

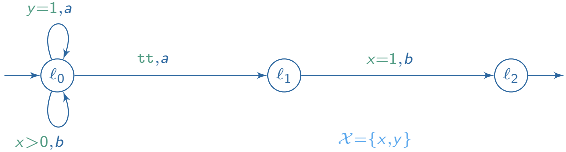
The model, informally

Timed automaton: Finite automaton enriched with clocks.



The model, informally

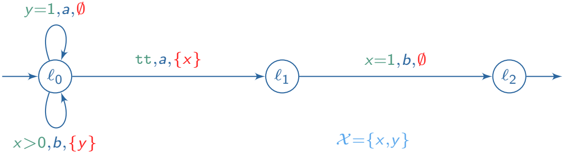
Timed automaton: Finite automaton enriched with clocks.



Transitions are equipped with guards

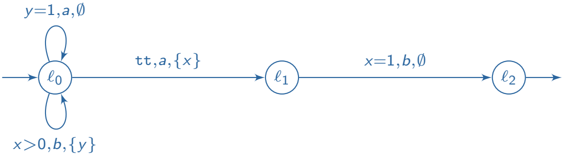
The model, informally

Timed automaton: Finite automaton enriched with clocks.



Transitions are equipped with guards and sets of reset clocks.

Syntax



Timed automata

A timed automaton is a tuple $\mathcal{A} = (L, L_0, L_{acc}, \Sigma, \mathcal{X}, E)$ with

- L finite set of locations $\qquad L = \{\ell_0, \ell_1, \ell_2\}$
- $L_0 \subseteq L$ initial locations $\qquad L_0 = \{\ell_0\}$
- $L_{acc} \subseteq L$ set of accepting locations $\qquad L_{acc} = \{\ell_2\}$
- Σ finite alphabet $\qquad \Sigma = \{a, b\}$
- $\mathcal{X}$ finite set of clocks $\qquad \mathcal{X} = \{x, y\}$
- $E \subseteq L \times \mathcal{G} \times \Sigma \times 2^{\mathcal{X}} \times L$ set of edges $\qquad \ell_0 \xrightarrow{x>0, a, \{y\}} \ell_0$
  where $\mathcal{G} = \{\bigwedge x \bowtie c \mid x \in \mathcal{X}, c \in \mathbb{N}\}$ is the set of guards.
  (with $\bowtie \in \{<, \leq, =, \geq, >\}$)

Semantics

Valuation: $v \in \mathbb{R}_+^{\mathcal{X}}$ assigns to each clock a clock-value

State: $(\ell, v) \in L \times \mathbb{R}_+^{\mathcal{X}}$ composed of a location and a valuation.

Transitions between states of $\mathcal{A}$:

- Delay transitions: $(\ell, v) \xrightarrow{\tau} (\ell, v + \tau)$
- Discrete transitions: $(\ell, v) \xrightarrow{a} (\ell', v')$

$$\text{if } \exists(\ell, g, a, Y, \ell') \in E \text{ with } v \models g \text{ and } \begin{cases} v'(x) = 0 & \text{if } x \in Y, \\ v'(x) = v(x) & \text{otherwise.} \end{cases}$$

Run of $\mathcal{A}$:

$(\ell_0, v_0) \xrightarrow{\tau_1} (\ell_0, v_0 + \tau_1) \xrightarrow{a_1} (\ell_1, v_1) \xrightarrow{\tau_2} (\ell_1, v_1 + \tau_2) \xrightarrow{a_2} \cdots \xrightarrow{a_k} (\ell_k, v_k)$

or simply: $(\ell_0, v_0) \xrightarrow{\tau_1, a_1} (\ell_1, v_1) \xrightarrow{\tau_2, a_2} \cdots \xrightarrow{\tau_k, a_k} (\ell_k, v_k)$

Semantics (cont.)

Time sequence: $\mathbf{t} = (t_i)_{1 \leq i \leq k}$ finite non-decreasing sequence over $\mathbb{R}_+$.

Timed word: $w = (\sigma, \mathbf{t}) = (a_i, t_i)_{1 \leq i \leq k}$ where $a_i \in \Sigma$ and $\mathbf{t}$ time sequence.

Accepted timed word

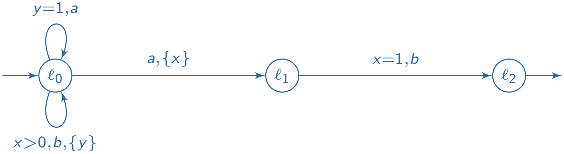
A timed word $w = (a_0, t_0)(a_1, t_1) \dots (a_k, t_k)$ is accepted in $\mathcal{A}$,
if there is a run $\rho = (\ell_0, v_0) \xrightarrow{\tau_0, a_0} (\ell_1, v_1) \xrightarrow{\tau_1, a_1} \dots (\ell_{k+1}, v_{k+1})$
with $\ell_0 \in L_0$, $\ell_{k+1} \in L_{acc}$, and $t_i = \sum_{j<i} \tau_j$.

Accepted timed language: $\mathcal{L}(\mathcal{A}) = \{w \mid w \text{ accepted by } \mathcal{A}\}$.

Back to the example

NB: In the examples, we omit

- the guard when it is equivalent to `tt`, and
- the reset set when it is empty.

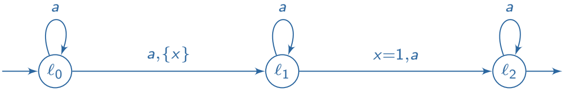


$w = (b,0.1)(b,0.3)(a,1.3)(b,1.5)(a,1.5)(b,2.5)$ is an accepted timed word

An accepting run for w is

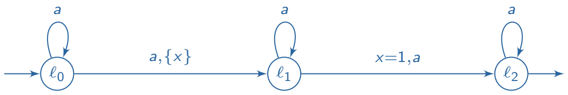
$$(\ell_0,0,0) \xrightarrow{0.1,b} (\ell_0,0.1,0) \xrightarrow{0.2,b} (\ell_0,0.3,0) \xrightarrow{1,a} (\ell_0,1.3,1)$$
$$\xrightarrow{0.2,b} (\ell_0,1.5,0) \xrightarrow{0,a} (\ell_1,0,0) \xrightarrow{1,b} (\ell_2,1,1)$$

More examples

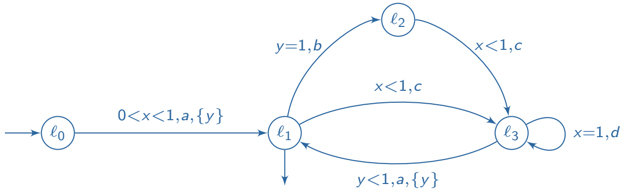


$$\mathcal{L}(\mathcal{A}) = \{(a, t_1) \cdots (a, t_k) | \exists i < j,\ t_j - t_i = 1\}$$

More examples



$$\mathcal{L}(\mathcal{A}) = \{(a,t_1)\cdots(a,t_k) | \exists i < j,\ t_j - t_i = 1\}$$



Does there exist an accepted timed word containing action b?

Variants of timed automata

Many variants in the litterature:

- Diagonal constraints: Guards are conjunctions of constraints of the form $x \bowtie c$ and $x - y \bowtie c$.
- Additive clock constraints: Constraints of the form $x \bowtie c$ and $x + y \bowtie c$.
- Epsilon transitions: Actions from the alphabet $\Sigma \cup \{\varepsilon\}$.
- Updatable TA: Clocks updates of the form: $x :\bowtie c$ and $x :\bowtie y + c$.

Outline

Region partitioning

Let $\mathcal{A}$ be a timed automaton with set of clocks $\mathcal{X}$ and set of constraints $\mathcal{C}$.
Let $\mathcal{R}$ be a finite partition of $\mathbb{R}_+^{\mathcal{X}}$, the set of valuations.

Set of regions

$\mathcal{R}$ is a set of regions (for $\mathcal{C}$) if

1. for every $g \in \mathcal{C}$ and for every $R \in \mathcal{R}$, $R \subseteq [\![g]\!]$ or $[\![g]\!] \cap R = \emptyset$,
2. for all $R, R' \in \mathcal{R}$, if there exists $v \in R$ and $t \in \mathbb{R}$ with $v + t \in R'$ then for every $v' \in R$ there exists $t' \in \mathbb{R}$ with $v' + t' \in R'$, and
3. for all $R, R' \in \mathcal{R}$, for every $Y \subseteq \mathcal{X}$ if $R_{[Y \leftarrow 0]} \cap R' \neq \emptyset$, then $R_{[Y \leftarrow 0]} \subseteq R'$.

Region partitioning

Let $\mathcal{A}$ be a timed automaton with set of clocks $\mathcal{X}$ and set of constraints $\mathcal{C}$.
Let $\mathcal{R}$ be a finite partition of $\mathbb{R}_+^{\mathcal{X}}$, the set of valuations.

Set of regions

$\mathcal{R}$ is a set of regions (for $\mathcal{C}$) if

1. for every $g \in \mathcal{C}$ and for every $R \in \mathcal{R}$, $R \subseteq [\![g]\!]$ or $[\![g]\!] \cap R = \emptyset$,
2. for all $R, R' \in \mathcal{R}$, if there exists $v \in R$ and $t \in \mathbb{R}$ with $v + t \in R'$ then for every $v' \in R$ there exists $t' \in \mathbb{R}$ with $v' + t' \in R'$, and
3. for all $R, R' \in \mathcal{R}$, for every $Y \subseteq \mathcal{X}$ if $R_{[Y \leftarrow 0]} \cap R' \neq \emptyset$, then $R_{[Y \leftarrow 0]} \subseteq R'$.

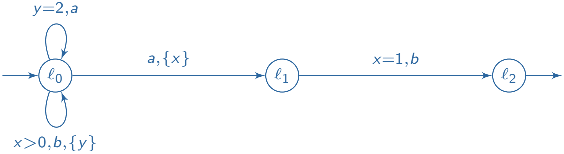
Let M be the maximal constant in $\mathcal{A}$.
The following equivalence relation yields the set of standard regions:

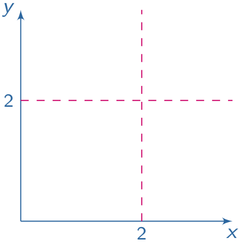
$$v \equiv^M v' \text{ if for every } x, y \in \mathcal{X}$$

- $v(x) > M \Leftrightarrow v'(x) > M$
- $v(x) \leq M \Rightarrow \Big((\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor) \text{ and } (\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0) \Big)$
- $(v(x) \leq M \text{ and } v(y) \leq M) \Rightarrow (\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\})$

Regions with 2 clocks

Standard regions for 2 clocks can be represented in 2 dimensions.



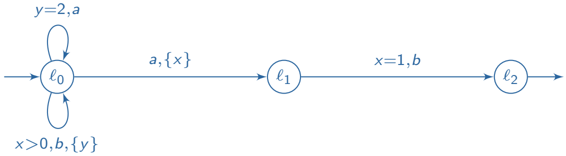


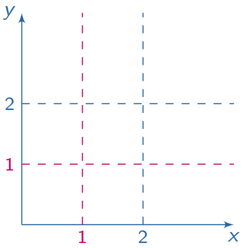
$v \equiv^M v'$ if for every $x, y \in \mathcal{X}$

- $v(x) > M \Leftrightarrow v'(x) > M$
- $v(x) \leq M \Rightarrow \Big((\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor)$ and $(\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0) \Big)$
- $(v(x) \leq M \text{ and } v(y) \leq M)$ $\Rightarrow (\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\})$

Regions with 2 clocks

Standard regions for 2 clocks can be represented in 2 dimensions.



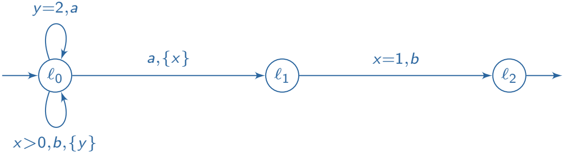


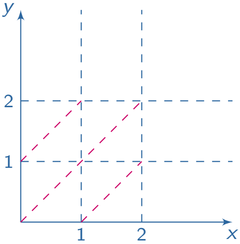
$v \equiv^M v'$ if for every $x, y \in \mathcal{X}$

- $v(x) > M \Leftrightarrow v'(x) > M$
- $v(x) \leq M \Rightarrow \Big(\big(\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor\big)$ and $\big(\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0\big) \Big)$
- $\big(v(x) \leq M \text{ and } v(y) \leq M\big)$ $\Rightarrow \big(\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\}\big)$

Regions with 2 clocks

Standard regions for 2 clocks can be represented in 2 dimensions.



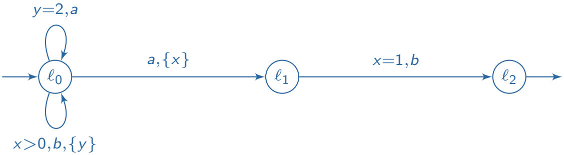


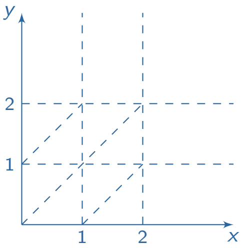
$v \equiv^M v'$ if for every $x, y \in \mathcal{X}$

- $v(x) > M \Leftrightarrow v'(x) > M$
- $v(x) \leq M \Rightarrow \Big((\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor)$ and $(\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0) \Big)$
- $(v(x) \leq M \text{ and } v(y) \leq M)$ $\Rightarrow (\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\})$

Regions with 2 clocks

Standard regions for 2 clocks can be represented in 2 dimensions.





$v \equiv^M v'$ if for every $x, y \in \mathcal{X}$

- $v(x) > M \Leftrightarrow v'(x) > M$
- $v(x) \leq M \Rightarrow \Big((\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor)$ and $(\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0) \Big)$
- $(v(x) \leq M \text{ and } v(y) \leq M)$ $\Rightarrow (\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\})$

The partition is compatible with constraints, time elapsing and resets.

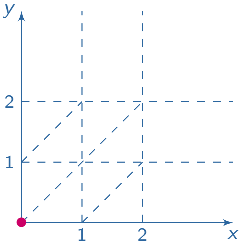
Operations on region

For two clocks, the (bounded) regions have the following shapes:

Operations on region

For two clocks, the (bounded) regions have the following shapes:

$R_{[Y\leftarrow 0]}$ denotes the region obtained from R by resetting clocks in $Y \subseteq \mathcal{X}$.
R' is a time-successor of R if there exists $v' \in R'$, $v \in R$, $t \in \mathbb{R}_+$ with $v' = v + t$.

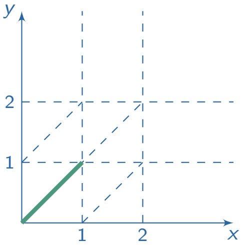


$(x{=}0, y{=}0)$

Operations on region

For two clocks, the (bounded) regions have the following shapes:

$R_{[Y \leftarrow 0]}$ denotes the region obtained from R by resetting clocks in $Y \subseteq \mathcal{X}$.
R' is a time-successor of R if there exists $v' \in R'$, $v \in R$, $t \in \mathbb{R}_+$ with $v' = v + t$.

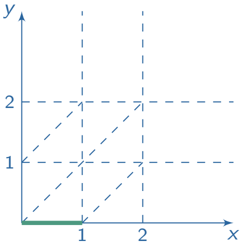


$(x{=}0, y{=}0) \xrightarrow{\text{delay}} (0{<}x{=}y{<}1)$

Operations on region

For two clocks, the (bounded) regions have the following shapes:

$R_{[Y \leftarrow 0]}$ denotes the region obtained from R by resetting clocks in $Y \subseteq \mathcal{X}$.
R' is a time-successor of R if there exists $v' \in R'$, $v \in R$, $t \in \mathbb{R}_+$ with $v' = v + t$.

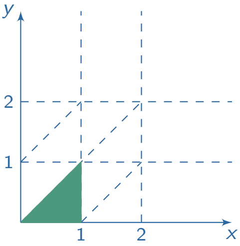


$$(x{=}0, y{=}0) \xrightarrow{\text{delay}} (0{<}x{=}y{<}1) \xrightarrow{y:=0} (0{<}x{<}1, y{=}0)$$

Operations on region

For two clocks, the (bounded) regions have the following shapes:

$R_{[Y \leftarrow 0]}$ denotes the region obtained from R by resetting clocks in $Y \subseteq \mathcal{X}$.
R' is a time-successor of R if there exists $v' \in R'$, $v \in R$, $t \in \mathbb{R}_+$ with $v' = v + t$.

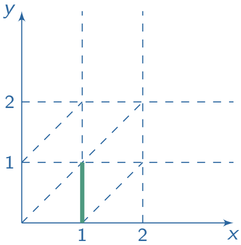


$(x{=}0, y{=}0) \xrightarrow{\text{delay}} (0{<}x{=}y{<}1) \xrightarrow{y:=0} (0{<}x{<}1, y{=}0) \xrightarrow{\text{delay}} (0{<}y{<}x{<}1)$

Operations on region

For two clocks, the (bounded) regions have the following shapes:

$R_{[Y \leftarrow 0]}$ denotes the region obtained from R by resetting clocks in $Y \subseteq \mathcal{X}$.
R' is a time-successor of R if there exists $v' \in R'$, $v \in R$, $t \in \mathbb{R}_+$ with $v' = v + t$.

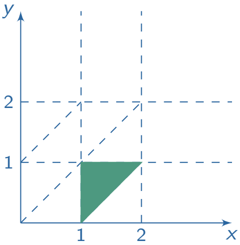


$(x{=}0, y{=}0) \xrightarrow{\text{delay}} (0{<}x{=}y{<}1) \xrightarrow{y:=0} (0{<}x{<}1, y{=}0)$
$\xrightarrow{\text{delay}} (0{<}y{<}x{<}1) \xrightarrow{\text{delay}} (0{<}y{<}1{=}x)$

Operations on region

For two clocks, the (bounded) regions have the following shapes:

$R_{[Y\leftarrow 0]}$ denotes the region obtained from R by resetting clocks in $Y \subseteq \mathcal{X}$.
R' is a time-successor of R if there exists $v' \in R'$, $v \in R$, $t \in \mathbb{R}_+$ with $v' = v + t$.

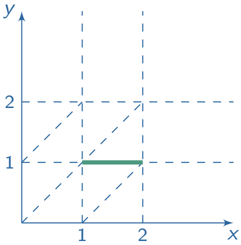


$(x{=}0,y{=}0) \xrightarrow{\text{delay}} (0{<}x{=}y{<}1) \xrightarrow{y:=0} (0{<}x{<}1,y{=}0) \xrightarrow{\text{delay}} (0{<}y{<}x{<}1) \xrightarrow{\text{delay}} (0{<}y{<}1{=}x) \xrightarrow{\text{delay}} (1{<}x{<}2,0{<}y{<}1,\{x\}{<}\{y\})$

Operations on region

For two clocks, the (bounded) regions have the following shapes:

$R_{[Y \leftarrow 0]}$ denotes the region obtained from R by resetting clocks in $Y \subseteq \mathcal{X}$.
R' is a time-successor of R if there exists $v' \in R'$, $v \in R$, $t \in \mathbb{R}_+$ with $v' = v + t$.

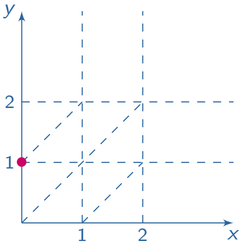


$(x{=}0,y{=}0) \xrightarrow{\text{delay}} (0{<}x{=}y{<}1) \xrightarrow{y:=0} (0{<}x{<}1,y{=}0) \xrightarrow{\text{delay}} (0{<}y{<}x{<}1) \xrightarrow{\text{delay}} (0{<}y{<}1{=}x) \xrightarrow{\text{delay}} (1{<}x{<}2,0{<}y{<}1,\{x\}{<}\{y\}) \xrightarrow{\text{delay}} (y{=}1{<}x{<}2)$

Operations on region

For two clocks, the (bounded) regions have the following shapes:

$R_{[Y \leftarrow 0]}$ denotes the region obtained from R by resetting clocks in $Y \subseteq \mathcal{X}$.
R' is a time-successor of R if there exists $v' \in R'$, $v \in R$, $t \in \mathbb{R}_+$ with $v' = v + t$.



$$(x{=}0,y{=}0) \xrightarrow{\text{delay}} (0{<}x{=}y{<}1) \xrightarrow{y:=0} (0{<}x{<}1,y{=}0) \xrightarrow{\text{delay}} (0{<}y{<}x{<}1) \xrightarrow{\text{delay}} (0{<}y{<}1{=}x) \xrightarrow{\text{delay}} (1{<}x{<}2,0{<}y{<}1,\{x\}{<}\{y\}) \xrightarrow{\text{delay}} (y{=}1{<}x{<}2) \xrightarrow{x:=0} (x{=}0,y{=}1)$$

Region automaton: construction

From a timed automaton $\mathcal{A}$ we build a finite automaton $\alpha(\mathcal{A})$ as follows:

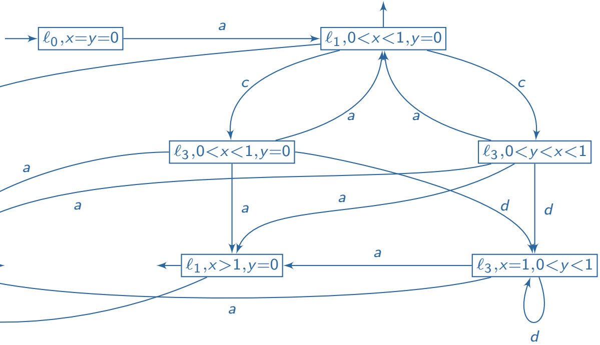
- States: $L \times \mathcal{R}$ Initial: $L_0 \times \mathcal{R}$ Final: $L_{acc} \times \mathcal{R}$
- Transitions:
  - $(\ell, R) \xrightarrow{a} (\ell', R')$ if there exists $\ell \xrightarrow{g,a,Y} \ell'$ in $\mathcal{A}$, there exists R'' time-successor of R with $R'' \subseteq [\![g]\!]$ and $R' = R''_{[Y \leftarrow 0]}$.

Region automaton: construction

From a timed automaton $\mathcal{A}$ we build a finite automaton $\alpha(\mathcal{A})$ as follows:

- States: $L \times \mathcal{R}$ Initial: $L_0 \times \mathcal{R}$ Final: $L_{acc} \times \mathcal{R}$
- Transitions:
  - $(\ell, R) \xrightarrow{a} (\ell', R')$ if there exists $\ell \xrightarrow{g,a,Y} \ell'$ in $\mathcal{A}$, there exists R'' time-successor of R with $R'' \subseteq [\![g]\!]$ and $R' = R''_{[Y \leftarrow 0]}$.

Example Region automaton for the second timed automaton of Slide 13.



Region automaton: properties

The number of states in $\alpha(\mathcal{A})$ is bounded by

$$|L| \cdot 2^{|\mathcal{X}|} \cdot |\mathcal{X}|! \cdot (2M+2)^{|\mathcal{X}|}$$

Region automaton: properties

The number of states in $\alpha(\mathcal{A})$ is bounded by

$$|L| \cdot 2^{|\mathcal{X}|} \cdot |\mathcal{X}|! \cdot (2M+2)^{|\mathcal{X}|}$$

$\text{Untime}(\mathcal{L}(\mathcal{A})) = \{\sigma | (\sigma, \mathbf{t}) \in \mathcal{L}(\mathcal{A})\} \subseteq \Sigma^*$ is the untimed language of $\mathcal{A}$.

Property

$\text{Untime}(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\alpha(\mathcal{A}))$

Consequence: the untimed language of $\mathcal{A}$ is regular.

Justification of the region automaton

Time-abstract bisimulation

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be timed automata.
$\equiv \subseteq (L_1 \times \mathbb{R}_+^{\mathcal{X}_1}) \times (L_2 \times \mathbb{R}_+^{\mathcal{X}_2})$ is a time-abstract bisimulation between $\mathcal{A}_1$ and $\mathcal{A}_2$ if

- if $(\ell_1, v_1) \equiv (\ell_2, v_2)$ and $(\ell_1, v_1) \xrightarrow{\tau_1} (\ell_1, v_1 + \tau_1)$ for some $\tau_1 \in \mathbb{R}_+$, then there exists $\tau_2 \in \mathbb{R}_+$ with $(\ell_2, v_2) \xrightarrow{\tau_2} (\ell_2, v_2 + \tau_2)$ and $(\ell_1, v_1 + \tau_1) \equiv (\ell_2, v_2 + \tau_2)$
- if $(\ell_1, v_1) \equiv (\ell_2, v_2)$ and $(\ell_1, v_1) \xrightarrow{a} (\ell_1', v_1')$ for some $a \in \Sigma$, then there exists (ℓ_2', v_2') with $(\ell_2, v_2) \xrightarrow{a} (\ell_2', v_2')$ and $(\ell_1', v_1') \equiv (\ell_2', v_2')$
- and vice versa.

Justification of the region automaton

Time-abstract bisimulation

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be timed automata.
$\equiv \subseteq (L_1 \times \mathbb{R}_+^{\mathcal{X}_1}) \times (L_2 \times \mathbb{R}_+^{\mathcal{X}_2})$ is a time-abstract bisimulation between $\mathcal{A}_1$ and $\mathcal{A}_2$ if

- if $(\ell_1, v_1) \equiv (\ell_2, v_2)$ and $(\ell_1, v_1) \xrightarrow{\tau_1} (\ell_1, v_1 + \tau_1)$ for some $\tau_1 \in \mathbb{R}_+$, then there exists $\tau_2 \in \mathbb{R}_+$ with $(\ell_2, v_2) \xrightarrow{\tau_2} (\ell_2, v_2 + \tau_2)$ and $(\ell_1, v_1 + \tau_1) \equiv (\ell_2, v_2 + \tau_2)$
- if $(\ell_1, v_1) \equiv (\ell_2, v_2)$ and $(\ell_1, v_1) \xrightarrow{a} (\ell_1', v_1')$ for some $a \in \Sigma$, then there exists (ℓ_2', v_2') with $(\ell_2, v_2) \xrightarrow{a} (\ell_2', v_2')$ and $(\ell_1', v_1') \equiv (\ell_2', v_2')$
- and vice versa.

Let $\mathcal{A}$ be a timed automaton with maximal constant M.

Regions and time-abstract bisimulation

The relation $\equiv_M$ is a time-abstract bisimulation with finite index.

Reachability problem

Input: $\mathcal{A}$ timed automaton, ℓ location of $\mathcal{A}$
Question: is location ℓ reachable in $\mathcal{A}$?

Reachability problem

Reachability is decidable for timed automata. It is a PSPACE-complete problem.

Proof

- PSPACE-membership:
  - ℓ is reachable in $\mathcal{A}$ if and only if (ℓ, R) is reachable in $\alpha(\mathcal{A})$ for some R.
  - reachability is in NLOGSPACE for finite automata
  - $\alpha(\mathcal{A})$ has exponentially more states than $\mathcal{A}$
- PSPACE-hardness: reduction of the termination problem for a Turing machine with linearly bounded work space. See black board.

Outline

Universality and language inclusion

Universality

Input: $\mathcal{A}$ timed automaton

Question: does $\mathcal{A}$ accept all timed words?

Undecidability result

Universality is undecidable for timed automata.

Universality and language inclusion

Universality

Input: $\mathcal{A}$ timed automaton

Question: does $\mathcal{A}$ accept all timed words?

Undecidability result

Universality is undecidable for timed automata.

Language inclusion

Input: $\mathcal{A}_1$, $\mathcal{A}_2$ timed automata

Question: $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$?

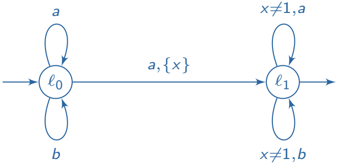
Corollary: Language inclusion in undecidable for timed automata.

Complementation

Non-closure

Timed automata are not closed under complement.

Proof hint The automaton below accepts a timed language whose complement cannot be recognized by a timed automaton.



Determinization

Deterministic TA

$\mathcal{A}$ is deterministic if $|L_0| = 1$ and for each $\ell \in L$, for every $a \in \Sigma$, $\ell \xrightarrow{g_1,a,Y_1} \ell_1$ and $\ell \xrightarrow{g_2,a,Y_2} \ell_2$ implies $[\![g_1]\!] \cap [\![g_2]\!] = \emptyset$.

If $\mathcal{A}$ is deterministic, there is at most one run on each timed word.

Closure

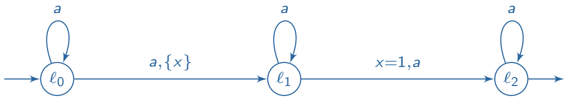
Deterministic timed automata are closed under complementation.

Expressivity

Timed automata are strictly more expressive than deterministic ones.

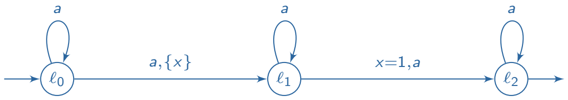
Determinizability

Example The automaton below accepts a timed language which cannot be recognized by a deterministic timed automaton. See black board.



Determinizability

Example The automaton below accepts a timed language which cannot be recognized by a deterministic timed automaton. See black board.



Determinizability

Telling whether a timed automaton can be determinized is undecidable.

Proof See black board.

Outline

Diagonal constraints

Guards may contain atomic constraints of the form $x - y \bowtie c$ for $x, y \in \mathcal{X}$.

Expressivity

Timed automata with diagonal constraints are equally expressive as classical timed automata.

Proof hint For every diagonal constraint $x - y \leq c$, duplicate the timed automaton: In the first copy $x - y \leq c$ holds and in the other copy $x - y > c$ holds.

Efficiency

Timed automata with diagonal constraints can be exponentially more succint than classical timed automata.

Silent transitions

Actions are taken from the alphabet $\Sigma \cup \{\varepsilon\}$.

Expressivity

Timed automata with silent transitions are stricly more expressive than classical timed automata.

Example

$$\mathcal{L}_\varepsilon = \{(a, t_1) \cdots (a, t_n) \mid \forall k,\ t_k \mod 2 = 0\}$$

is recognizable by a timed automaton with ε-transitions, but cannot be recognized by a classical timed automaton.

Reachability

Reachability is decidable for timed automata with silent transitions.

Additive clock constraints

Guards may contain atomic constraints of the form $x + y \bowtie c$ for $x, y \in \mathcal{X}$.

Two clocks

The reachability problem for timed automata with two clocks and additive clock constraints is decidable.

Additive clock constraints

Guards may contain atomic constraints of the form $x + y \bowtie c$ for $x, y \in \mathcal{X}$.

Two clocks

The reachability problem for timed automata with two clocks and additive clock constraints is decidable.

Four or more clocks

The reachability problem for timed automata with four (or more) clocks and additive clock constraints is undecidable.

Proof Reduction of the halting problem for a two counter machine. See black board.

Outline

Symbolic model checking

Two general methods to solve the reachability problem.

Forward analysis

iterative computation
of successors of Init

Symbolic model checking

Two general methods to solve the reachability problem.

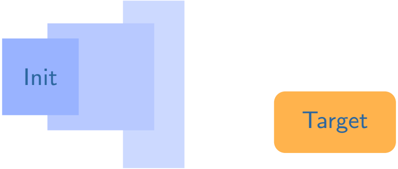
Forward analysis

iterative computation
of successors of Init

Symbolic model checking

Two general methods to solve the reachability problem.

Forward analysis

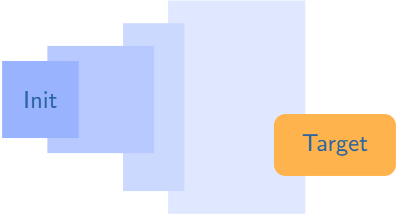


iterative computation
of successors of Init

Symbolic model checking

Two general methods to solve the reachability problem.

Forward analysis



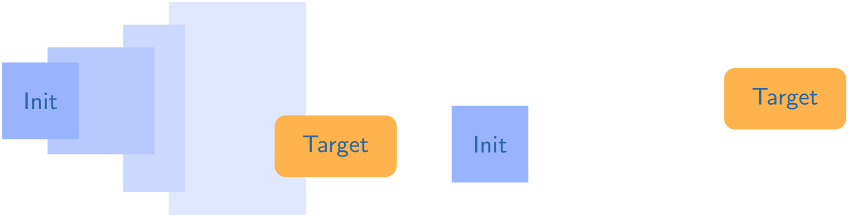
iterative computation
of successors of Init

Symbolic model checking

Two general methods to solve the reachability problem.

Forward analysis

Backward analysis



iterative computation of successors of Init

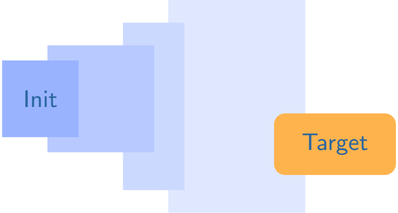
iterative computation of predecessors of Target

Symbolic model checking

Two general methods to solve the reachability problem.

Forward analysis

Backward analysis



iterative computation
of successors of Init

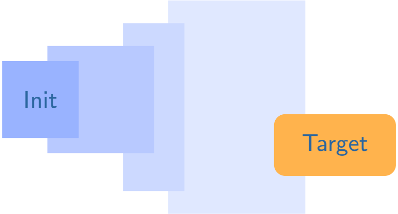
iterative computation
of predecessors of Target

Symbolic model checking

Two general methods to solve the reachability problem.

Forward analysis

Backward analysis



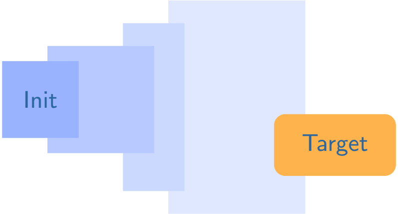
iterative computation
of successors of Init

iterative computation
of predecessors of Target

Symbolic model checking

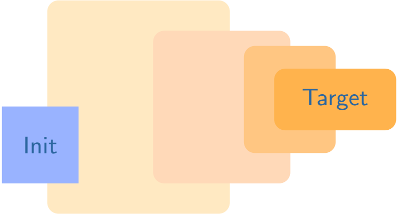
Two general methods to solve the reachability problem.

Forward analysis



iterative computation
of successors of Init

Backward analysis

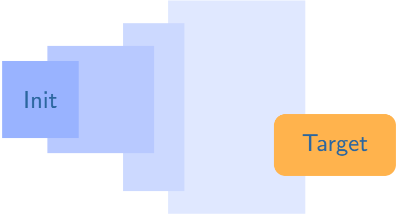


iterative computation
of predecessors of Target

Symbolic model checking

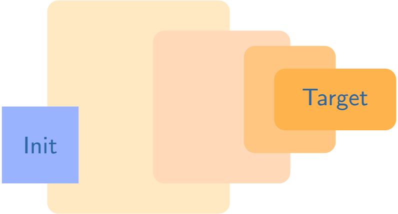
Two general methods to solve the reachability problem.

Forward analysis



iterative computation
of successors of Init

Backward analysis



iterative computation
of predecessors of Target

Issues: Representation of the sets of states + Termination of the computation.

Zones

Zones are symbolic representations of sets of valuations.
A clock constraint g defines a zone $[\![g]\!] = \{v \in \mathbb{R}_+^{\mathcal{X}} | v \models g\}$.

For verification purposes, the following operations on zones Z, Z' are needed.

- forward analysis:
  - Future of Z: $\overrightarrow{Z} = \{v + t | v \in Z,\ t \in \mathbb{R}_+\}$
  - Reset in Z of clocks in $Y \subseteq \mathcal{X}$: $Z_{[Y \leftarrow 0]} = \{v_{[Y \leftarrow 0]} | v \in Z\}$
  - Intersection of Z and Z': $Z \cap Z' = \{v | v \in Z \text{ and } v \in Z'\}$
  - Emptiness test: decide if Z is empty.
- backward analysis:
  - Past of Z: $\overleftarrow{Z} = \{v - t | v \in Z,\ t \in \mathbb{R}_+\}$
  - Inverse reset: $Z_{[Y \leftarrow 0]^{-1}}$ the largest Z' with $Z'_{[Y \leftarrow 0]} = Z$
  - Intersection
  - Emptiness test

Data structure

Zones are represented by Difference Bounded Matrices (DBM).

Difference Bounded Matrix

A DBM over the set of n clocks $\mathcal{X}$ is an $(n+1)$-square matrix of pairs

$$(m, \prec) \text{ with } \prec \in \{<, \leq\} \text{ and } m \in \mathbb{Z} \cup \{\infty\}$$

$(m_{i,j}, \prec_{i,j})$ encodes the constraint $x_i - x_j \prec_{i,j} m_{i,j}$ (with convention $x_0 = 0$)

Data structure

Zones are represented by Difference Bounded Matrices (DBM).

Difference Bounded Matrix

A DBM over the set of n clocks $\mathcal{X}$ is an $(n+1)$-square matrix of pairs

$$(m, \prec) \text{ with } \prec \in \{<, \leq\} \text{ and } m \in \mathbb{Z} \cup \{\infty\}$$

$(m_{i,j}, \prec_{i,j})$ encodes the constraint $x_i - x_j \prec_{i,j} m_{i,j}$ (with convention $x_0 = 0$)

Example A DBM and the zone it represents.

$$\begin{array}{cc} & \begin{array}{ccc} 0 & x & y \end{array} \\ \begin{array}{c} 0 \\ x \\ y \end{array} & \begin{pmatrix} (\infty, <) & (-3, \leq) & (\infty, <) \\ (\infty, <) & (\infty, <) & (4, <) \\ (5, \leq) & (\infty, <) & (\infty, <) \end{pmatrix} \end{array}$$

$$x \geq 3 \wedge y \leq 5 \wedge x - y < 4$$

Data structure

Zones are represented by Difference Bounded Matrices (DBM).

Difference Bounded Matrix

A DBM over the set of n clocks $\mathcal{X}$ is an $(n+1)$-square matrix of pairs

$$(m, \prec) \text{ with } \prec \in \{<, \leq\} \text{ and } m \in \mathbb{Z} \cup \{\infty\}$$

$(m_{i,j}, \prec_{i,j})$ encodes the constraint $x_i - x_j \prec_{i,j} m_{i,j}$ (with convention $x_0 = 0$)

Example A DBM and the zone it represents.

$$\begin{array}{c c} & \begin{array}{ccc} 0 & x & y \end{array} \\ \begin{array}{c} 0 \\ x \\ y \end{array} & \begin{pmatrix} (\infty, <) & (-3, \leq) & (\infty, <) \\ (\infty, <) & (\infty, <) & (4, <) \\ (5, \leq) & (\infty, <) & (\infty, <) \end{pmatrix} \end{array}$$

$$x \geq 3 \wedge y \leq 5 \wedge x - y < 4$$

Normal form (via Floyd algorithm)

$$\begin{array}{c c} & \begin{array}{ccc} 0 & x & y \end{array} \\ \begin{array}{c} 0 \\ x \\ y \end{array} & \begin{pmatrix} (0, \leq) & (-3, \leq) & (0, \leq) \\ (9, <) & (0, \leq) & (4, <) \\ (5, \leq) & (2, \leq) & (0, \leq) \end{pmatrix} \end{array}$$

Comparison

Backward analysis

The backward analysis terminates and is correct.

Proof Termination is based on the fact that finite union of regions are stable under the following operations: past $\overleftarrow{Z}$, inverse reset $Z_{[Y\leftarrow 0]^{-1}}$, and intersection $g \cap Z$.

Comparison

Backward analysis

The backward analysis terminates and is correct.

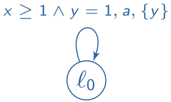
Proof Termination is based on the fact that finite union of regions are stable under the following operations: past $\overleftarrow{Z}$, inverse reset $Z_{[Y\leftarrow 0]^{-1}}$, and intersection $g \cap Z$.

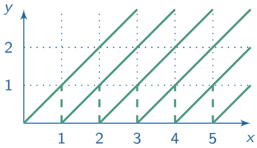
Forward analysis

The forward analysis is correct when it terminates.

Note that it may not terminate.

Example





Uppaal in a nutshell

UPPAAL

- developed at Uppsala and Aalborg universities
- performs forward analysis (with extrapolation) for timed automata

http://www.uppaal.com/

See demo.

Outline

Bibliography

AD94 R. Alur and D. Dill. *A Theory of Timed Automata*. Theoretical Computer Science 126(2): 183-235. 1994.

Fin06 O. Finkel. *Undecidable Problems about Timed Automata*. Proceedings of FORMATS'06, pages 187-199. 2006.

AL02 L. Aceto and F. Laroussinie. *Is your model-checker on time? On the complexity of model checking for timed modal logics*. Journal Logic Algebraic Programming 52-53: 7-51. 2002.

AM04 R. Alur and P. Madhusudan. *Decision Problems for Timed Automata: A Survey*. Proceedings of SFM-04, pages 1-24. 2004.

The end!