

Modeling and Analysis of Hybrid Systems

Model checking timed automata

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen University

SS 2012

Basic method: Abstraction

- **Given:** a **concrete** system
(here: a timed automaton)
- **Goal:** **reduce the size** of the system
(here: reduce the infinite state space to a finite one)
- **Result:** **abstract** system
(here: region transition system)

Basic method: Abstraction

- **Given:** a **concrete** system
(here: a timed automaton)
- **Goal:** **reduce the size** of the system
(here: reduce the infinite state space to a finite one)
- **Result:** **abstract** system
(here: region transition system)
- **Conservative (safe) abstraction:** If we see both the concrete and the abstract system as black boxes and make experiments with them, we **cannot distinguish between their observable behavior**.
- Two systems P and P' have the same observable behaviour iff for each context C we have that $\llbracket C[P] \rrbracket = \llbracket C[P'] \rrbracket$.
($C[P]$: the composition of C and P , $\llbracket \cdot \rrbracket$: (global) semantics)
- E.g., for programs it could mean the same input-output behaviour.
For model checking we require that they satisfy the same formulas of the underlying logic.
(here: TCTL)

Input: timed automaton \mathcal{T} , TCTL formula ψ

Output: the answer to the question if $\mathcal{T} \models \psi$

- 1 Eliminate the timing parameters from ψ , resulting in $\hat{\psi}$;
- 2 Make a finite abstraction of the state space
- 3 Construct abstract transition system RTS with
 $\mathcal{T} \models \psi$ iff $RTS \models \hat{\psi}$.
- 4 Apply CTL model checking to check whether $RTS \models \hat{\psi}$;
- 5 Return the model checking result.

Input: timed automaton \mathcal{T} , TCTL formula ψ

Output: the answer to the question if $\mathcal{T} \models \psi$

- 1 Eliminate the timing parameters from ψ , resulting in $\hat{\psi}$;
- 2 Make a finite abstraction of the state space
- 3 Construct abstract transition system RTS with
 $\mathcal{T} \models \psi$ iff $RTS \models \hat{\psi}$.
- 4 Apply CTL model checking to check whether $RTS \models \hat{\psi}$;
- 5 Return the model checking result.

1. Eliminating timing parameters

Let \mathcal{T} be a timed automaton with clock set \mathcal{C} and atomic propositions AP .
Let $\mathcal{T}' = \mathcal{T} \oplus z$ result from \mathcal{T} by adding a fresh clock which never gets reset.

1. Eliminating timing parameters

Let \mathcal{T} be a timed automaton with clock set \mathcal{C} and atomic propositions AP .
Let $\mathcal{T}' = \mathcal{T} \oplus z$ result from \mathcal{T} by adding a fresh clock which never gets reset.

For any state σ of \mathcal{T} it holds that

1. Eliminating timing parameters

Let \mathcal{T} be a timed automaton with clock set \mathcal{C} and atomic propositions AP .
Let $\mathcal{T}' = \mathcal{T} \oplus z$ result from \mathcal{T} by adding a fresh clock which never gets reset.

For any state σ of \mathcal{T} it holds that

$$\begin{array}{l} \mathbf{1} \quad \sigma \quad \models_{TCTL} \mathbf{E}(\psi_1 \quad \mathcal{U}^J \quad \psi_2) \text{ iff} \\ \quad \text{reset}(z) \text{ in } \sigma \quad \models_{TCTL} \mathbf{E}((\psi_1 \vee \psi_2) \quad \mathcal{U} \quad ((z \in J) \wedge \psi_2)). \end{array}$$

1. Eliminating timing parameters

Let \mathcal{T} be a timed automaton with clock set \mathcal{C} and atomic propositions AP .
Let $\mathcal{T}' = \mathcal{T} \oplus z$ result from \mathcal{T} by adding a fresh clock which never gets reset.

For any state σ of \mathcal{T} it holds that

$$\begin{array}{l} \mathbf{1} \quad \sigma \quad \models_{TCTL} \mathbf{E}(\psi_1 \quad \mathcal{U}^J \quad \psi_2) \text{ iff} \\ \text{reset}(z) \text{ in } \sigma \quad \models_{TCTL} \mathbf{E}((\psi_1 \vee \psi_2) \quad \mathcal{U} \quad ((z \in J) \wedge \psi_2)). \end{array}$$

$$\begin{array}{l} \mathbf{2} \quad \sigma \quad \models_{TCTL} \mathbf{A}(\psi_1 \quad \mathcal{U}^J \quad \psi_2) \text{ iff} \\ \text{reset}(z) \text{ in } \sigma \quad \models_{TCTL} \mathbf{A}((\psi_1 \vee \psi_2) \quad \mathcal{U} \quad ((z \in J) \wedge \psi_2)). \end{array}$$

1. Eliminating timing parameters

Let \mathcal{T} be a timed automaton with clock set \mathcal{C} and atomic propositions AP .
Let $\mathcal{T}' = \mathcal{T} \oplus z$ result from \mathcal{T} by adding a fresh clock which never gets reset.

For any state σ of \mathcal{T} it holds that

- 1 $\sigma \models_{TCTL} \mathbf{E}(\psi_1 \quad \mathcal{U}^J \quad \psi_2) \text{ iff}$
 $reset(z) \text{ in } \sigma \models_{TCTL} \mathbf{E}((\psi_1 \vee \psi_2) \quad \mathcal{U} \quad ((z \in J) \wedge \psi_2)).$
- 2 $\sigma \models_{TCTL} \mathbf{A}(\psi_1 \quad \mathcal{U}^J \quad \psi_2) \text{ iff}$
 $reset(z) \text{ in } \sigma \models_{TCTL} \mathbf{A}((\psi_1 \vee \psi_2) \quad \mathcal{U} \quad ((z \in J) \wedge \psi_2)).$
- 3 $\sigma \models_{TCTL} \mathbf{EF}^{\leq 2} \psi_1 \quad \text{iff} \quad reset(z) \text{ in } \sigma \models_{TCTL} \mathbf{EF}((z \leq 2) \wedge \psi_1)$

1. Eliminating timing parameters

Let \mathcal{T} be a timed automaton with clock set \mathcal{C} and atomic propositions AP . Let $\mathcal{T}' = \mathcal{T} \oplus z$ result from \mathcal{T} by adding a fresh clock which never gets reset.

For any state σ of \mathcal{T} it holds that

$$\begin{array}{l} \text{1} \quad \sigma \models_{TCTL} \mathbf{E}(\psi_1 \quad \mathcal{U}^J \quad \psi_2) \text{ iff} \\ \text{reset}(z) \text{ in } \sigma \models_{TCTL} \mathbf{E}((\psi_1 \vee \psi_2) \quad \mathcal{U} \quad ((z \in J) \wedge \psi_2)). \end{array}$$

$$\begin{array}{l} \text{2} \quad \sigma \models_{TCTL} \mathbf{A}(\psi_1 \quad \mathcal{U}^J \quad \psi_2) \text{ iff} \\ \text{reset}(z) \text{ in } \sigma \models_{TCTL} \mathbf{A}((\psi_1 \vee \psi_2) \quad \mathcal{U} \quad ((z \in J) \wedge \psi_2)). \end{array}$$

$$\text{3} \quad \sigma \models_{TCTL} \mathbf{EF}^{\leq 2} \psi_1 \quad \text{iff} \quad \text{reset}(z) \text{ in } \sigma \models_{TCTL} \mathbf{EF}((z \leq 2) \wedge \psi_1)$$

$$\text{4} \quad \sigma \models_{TCTL} \mathbf{EG}^{\leq 2} \psi_1 \quad \text{iff} \quad \text{reset}(z) \text{ in } \sigma \models_{TCTL} \mathbf{EG}((z \leq 2) \rightarrow \psi_1)$$

Input: timed automaton \mathcal{T} , TCTL formula ψ

Output: the answer to the question if $\mathcal{T} \models \psi$

- 1 Eliminate the timing parameters from ψ , resulting in $\hat{\psi}$;
- 2 Make a finite abstraction of the state space
- 3 Construct abstract transition system RTS with
 $\mathcal{T} \models \psi$ iff $RTS \models \hat{\psi}$.
- 4 Apply CTL model checking to check whether $RTS \models \hat{\psi}$;
- 5 Return the model checking result.

Keywords:

Finite abstraction

Equivalence relation, equivalence classes

Bisimulation

And what does it mean in our context?

2. Finite state space abstraction

We search for an **equivalence relation** \cong on states, such that equivalent states satisfy the same (sub)formulae ψ' occurring in the timed automaton \mathcal{T} or in the specification ψ :

$$\sigma \cong \sigma' \Rightarrow (\sigma \models \psi' \text{ iff } \sigma' \models \psi').$$

Since the set of such (sub)formulae is finite, we strive for a **finite** number of equivalence classes.

Definition

Let $LSTS_1 = (\Sigma_1, Lab_1, Edge_1, Init_1)$, $LSTS_2 = (\Sigma_2, Lab_2, Edge_2, Init_2)$ be two state transition systems, AP a set of atomic propositions, and $L_1 : \Sigma_1 \rightarrow 2^{AP}$ and $L_2 : \Sigma_2 \rightarrow 2^{AP}$ labeling functions over AP .

A **bisimulation** for $(LSTS_1, LSTS_2)$ is an equivalence relation $\approx \subseteq \Sigma_1 \times \Sigma_2$ such that for all $\sigma_1 \approx \sigma_2$

- 1 $L(\sigma_1) = L(\sigma_2)$
- 2 for all $\sigma'_1 \in \Sigma_1$ with $\sigma_1 \xrightarrow{a} \sigma'_1$ there exists $\sigma'_2 \in \Sigma_2$ such that $\sigma_2 \xrightarrow{a} \sigma'_2$ and $\sigma'_1 \approx \sigma'_2$.

Definition

Let $LSTS = (\Sigma, Lab, Edge, Init)$ be a state transition system, AP a set of atomic propositions, and $L : \Sigma \rightarrow 2^{AP}$ a labeling function over AP .

A **bisimulation** for $LSTS$ is an equivalence relation $\approx \subseteq \Sigma \times \Sigma$ such that for all $\sigma_1 \approx \sigma_2$

- 1 $L(\sigma_1) = L(\sigma_2)$
- 2 for all $\sigma'_1 \in \Sigma$ with $\sigma_1 \xrightarrow{a} \sigma'_1$ there exists $\sigma'_2 \in \Sigma$ such that $\sigma_2 \xrightarrow{a} \sigma'_2$ and $\sigma'_1 \approx \sigma'_2$.

Definition

Let $\mathcal{T} = (Loc, \mathcal{C}, Lab, Edge, Inv, Init)$ be a timed automaton, AP a set of atomic propositions, and $L : \Sigma \rightarrow 2^{AP}$.

A **time abstract bisimulation** on \mathcal{T} is an equivalence relation $\approx \subseteq \Sigma \times \Sigma$ such that for all $\sigma_1, \sigma_2 \in \Sigma$ satisfying $\sigma_1 \approx \sigma_2$

- $L(\sigma_1) = L(\sigma_2)$
- for all $\sigma'_1 \in \Sigma$ with $\sigma_1 \xrightarrow{a} \sigma'_1$ there is a $\sigma'_2 \in \Sigma$ such that $\sigma_2 \xrightarrow{a} \sigma'_2$ and $\sigma'_1 \approx \sigma'_2$
- for all $\sigma'_1 \in \Sigma$ with $\sigma_1 \xrightarrow{t_1} \sigma'_1$ there is a $\sigma'_2 \in \Sigma$ such that $\sigma_2 \xrightarrow{t_2} \sigma'_2$ and $\sigma'_1 \approx \sigma'_2$.

Lemma

Assume a timed automaton \mathcal{T} with state space Σ , and a bisimulation $\approx \subseteq \Sigma \times \Sigma$ on \mathcal{T} .

Then for all $\sigma, \sigma' \in \Sigma$ with $\sigma \approx \sigma'$ we have that for each path

$$\pi : \sigma \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \xrightarrow{\alpha_3} \dots$$

of \mathcal{T} there exists a path

$$\pi' : \sigma' \xrightarrow{\alpha'_1} \sigma'_1 \xrightarrow{\alpha'_2} \sigma'_2 \xrightarrow{\alpha'_3} \dots$$

of \mathcal{T} such that for all i

- $\sigma_i \approx \sigma'_i$,
- $\alpha_i = \alpha'_i$ if $\alpha_i \in Lab$ and
- $\alpha_i, \alpha'_i \in \mathbb{R}_{\geq 0}$ otherwise.

2. Finite state space abstraction

Now, back to timed automata. How could such a bisimulation look like?

Since, in general,

- the atomic propositions assigned to and
- the paths starting at

different locations in \mathcal{T} are different, **only states (l, ν) and (l', ν') satisfying $l = l'$ should be equivalent.**

2. Finite state space abstraction

Equivalent states should satisfy the same **atomic clock constraints**.

Notation:

- Integral part of $r \in \mathbb{R}$: $\lfloor r \rfloor = \max \{c \in \mathbb{N} \mid c \leq r\}$
- Fractional part of $r \in \mathbb{R}$: $\text{frac}(r) = r - \lfloor r \rfloor$

For clock constraints $x < c$ with $c \in \mathbb{N}$ we have:

$$\nu \models x < c \Leftrightarrow \nu(x) < c \Leftrightarrow \lfloor \nu(x) \rfloor < c.$$

For clock constraints $x \leq c$ with $c \in \mathbb{N}$ we have:

$$\nu \models x \leq c \Leftrightarrow \nu(x) \leq c \Leftrightarrow \lfloor \nu(x) \rfloor < c \vee (\lfloor \nu(x) \rfloor = c \wedge \text{frac}(\nu(x)) = 0).$$

I.e., only states (l, ν) and (l, ν') satisfying

$$\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor \text{ and } \text{frac}(\nu(x)) = 0 \text{ iff } \text{frac}(\nu'(x)) = 0$$

for all $x \in \mathcal{C}$ should be equivalent.

$$V(x) \approx 1.2$$

$$V'(x) = 1.0$$

$$\underline{x < 1}$$

$$\underline{x = 1}$$

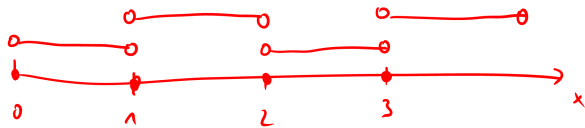
$$\underline{x > 1}$$

$$V(x) = 1.2$$

$$V'(x) = 0.8$$

—

—



$$x=0$$

$$x=1$$

$$x=2$$

$$x=3$$

2. Finite state space abstraction

Problem: It would generate infinitely many equivalence classes!

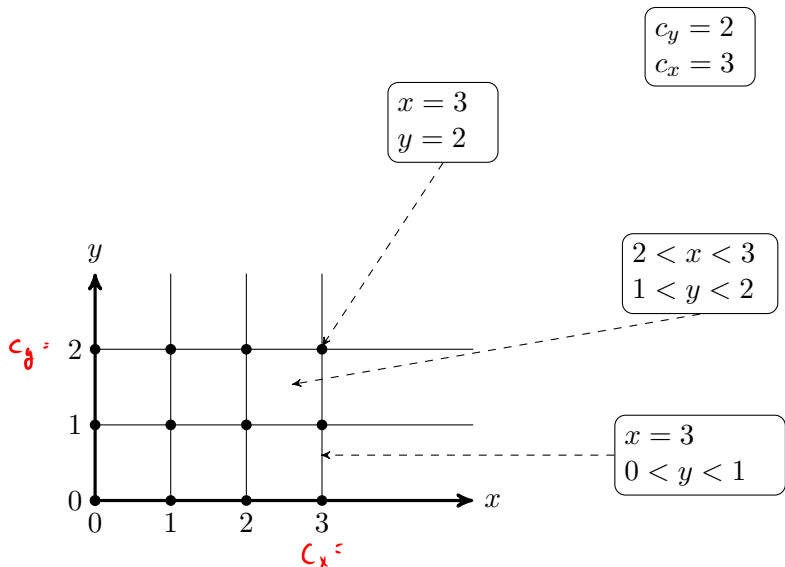
Let c_x be the largest constant which a clock x is compared to in \mathcal{T} or in ψ . Then there is no observation which could distinguish between the x -values in (l, ν) and (l, ν') if $\nu(x) > c_x$ and $\nu'(x) > c_x$.

i.e., only states (l, ν) and (l, ν') satisfying

$$\begin{aligned} & (\nu(x) > c_x \wedge \nu'(x) > c_x) \quad \vee \\ & \underline{(\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor) \wedge \underline{\text{frac}(\nu(x)) = 0 \text{ iff } \text{frac}(\nu'(x)) = 0}}, \end{aligned}$$

for all $x \in \mathcal{C}$ should be equivalent.

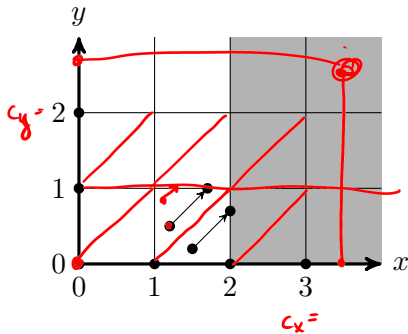
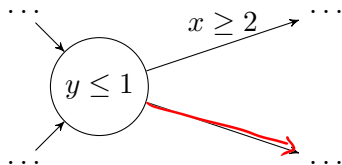
2. Finite state space abstraction



2. Finite state space abstraction

As the following example illustrates, we must make a further refinement of the abstraction, since it does not distinguish between states satisfying different formulae.

2. Finite state space abstraction



2. Finite state space abstraction

What we need is a refinement taking the **order of the fractional parts of the clock values** into account. However, again only for values below the largest constants to which the clocks get compared.

I.e., only states (l, ν) and (l, ν') satisfying

$$\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$$

$$\begin{aligned} & (\nu(x), \nu'(x) > c_x \wedge \nu(y), \nu'(y) > c_y) \quad \vee \\ & \left\{ \begin{array}{l} \rightarrow \text{frac}(\nu(x)) < \text{frac}(\nu(y)) \quad \text{iff} \quad \text{frac}(\nu'(x)) < \text{frac}(\nu'(y)) \quad \wedge \\ \text{frac}(\nu(x)) = \text{frac}(\nu(y)) \quad \text{iff} \quad \text{frac}(\nu'(x)) = \text{frac}(\nu'(y)) \quad \wedge \\ \rightarrow \text{frac}(\nu(x)) > \text{frac}(\nu(y)) \quad \text{iff} \quad \text{frac}(\nu'(x)) > \text{frac}(\nu'(y)) \end{array} \right. \end{aligned}$$

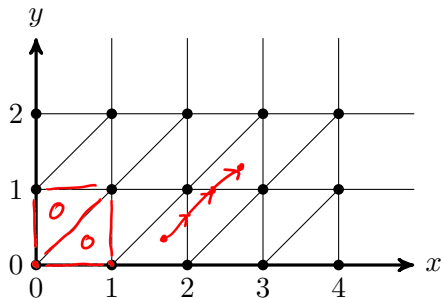
for all $x, y \in \mathcal{C}$ should be equivalent.

Because of symmetry the following is also sufficient:

$$\left\{ \begin{array}{l} (\nu(x), \nu'(x) > c_x \wedge \nu(y), \nu'(y) > c_y) \quad \vee \\ \text{frac}(\nu(x)) \leq \text{frac}(\nu(y)) \quad \text{iff} \quad \text{frac}(\nu'(x)) \leq \text{frac}(\nu'(y)) \end{array} \right.$$

for all $x, y \in \mathcal{C}$ should be equivalent.

2. Finite state space abstraction



$$c_y = 2$$

$$c_x = 4$$

finite index

2. Finite state space abstraction

Definition

For a timed automaton \mathcal{T} and a TCTL formula ψ , both over a clock set \mathcal{C} , we define the **clock equivalence relation** $\cong \subseteq \Sigma \times \Sigma$ by $(l, \nu) \cong (l', \nu')$ iff $l = l'$ and

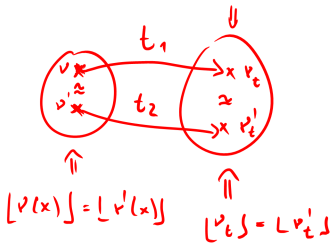
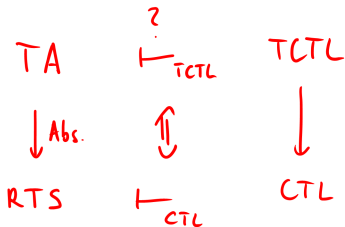
- for all $x \in \mathcal{C}$, either $\nu(x) > c_x \wedge \nu'(x) > c_x$ or

$$\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor \wedge (\text{frac}(\nu(x)) = 0 \text{ iff } \text{frac}(\nu'(x)) = 0)$$

- for all $x, y \in \mathcal{C}$ if $\nu(x), \nu'(x) \leq c_x$ and $\nu(y), \nu'(y) \leq c_x$ then

$$\text{frac}(\nu(x)) \leq \text{frac}(\nu(y)) \text{ iff } \text{frac}(\nu'(x)) \leq \text{frac}(\nu'(y)).$$

The **clock region** of an evaluation $\nu \in V$ is the set $[\nu] = \{\nu' \in V \mid \nu \cong \nu'\}$.
The **clock region** of a state $\sigma = (l, \nu) \in \Sigma$ is the set $[\sigma] = \{(l, \nu') \in \Sigma \mid \nu \cong \nu'\}$.



$$(l, v) \models E \varphi_1 \mathcal{U}^{\exists} \varphi_2$$

$$(l, v_t) \models E \varphi_1 \mathcal{U}^{\exists-t_1} \varphi_2$$

$$(l, v'_t) \models E \varphi_1 \mathcal{U}^{\exists-t_2} \varphi_2$$

$$(l, v') \models E \varphi_1 \mathcal{U}^{\exists} \varphi_2$$

2. Finite state space abstraction

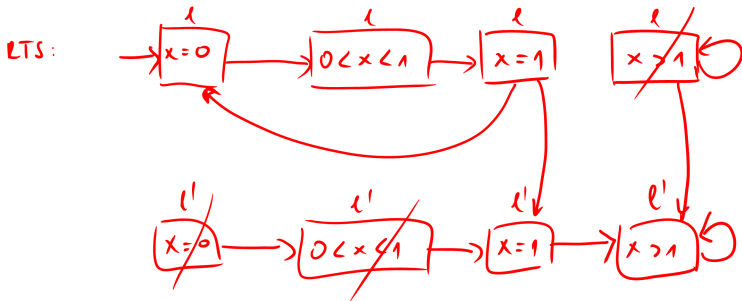
Lemma

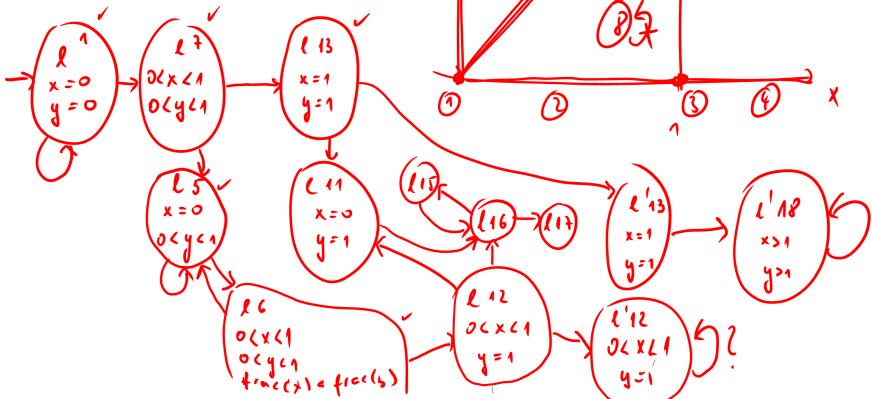
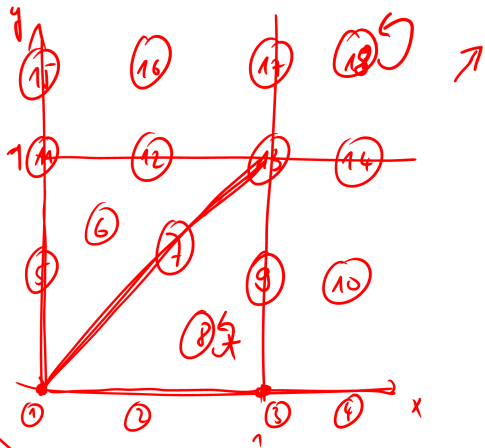
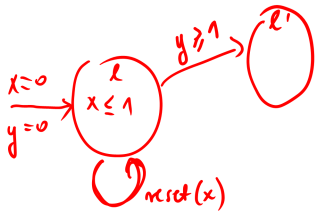
Clock equivalence is a bisimulation over $AP' = AP \cup ACC(\mathcal{T}) \cup ACC(\psi)$.

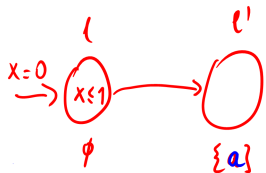
Input: timed automaton \mathcal{T} , TCTL formula ψ

Output: the answer to the question if $\mathcal{T} \models \psi$

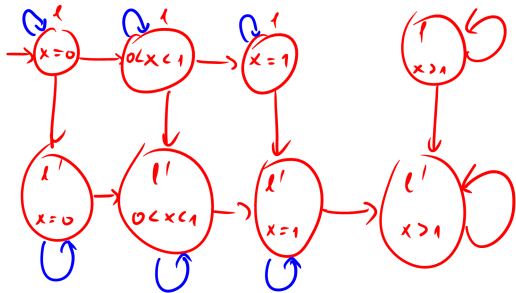
- 1 Eliminate the timing parameters from ψ , resulting in $\hat{\psi}$;
- 2 Make a finite abstraction of the state space
- 3 Construct abstract transition system *RTS* with
 $\mathcal{T} \models \psi$ iff *RTS* $\models \hat{\psi}$.
- 4 Apply CTL model checking to check whether *RTS* $\models \hat{\psi}$;
- 5 Return the model checking result.







$$(l, x=0) \rightarrow (l, x=\frac{1}{2}) \rightarrow (l, x=\frac{3}{4}) \rightarrow \dots$$



$A \bar{F} a$

3. The abstract transition system

We have defined regions as abstract states,
now we connect them by abstract transitions.

Two kinds of transitions:
time and discrete

3. The abstract transition system

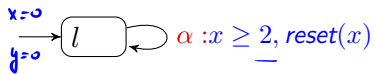
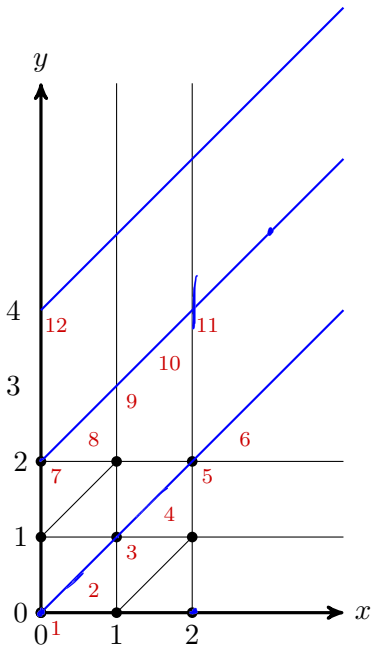
Definition

The clock region $r_\infty = \{\nu \in V \mid \forall x \in \mathcal{C}. \nu(x) > c_x\}$ is called **unbounded**. Let r, r' be two clock regions. The region r' is the **successor clock region** of r , denoted by $r' = \mathit{succ}(r)$, if either

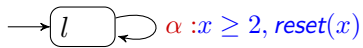
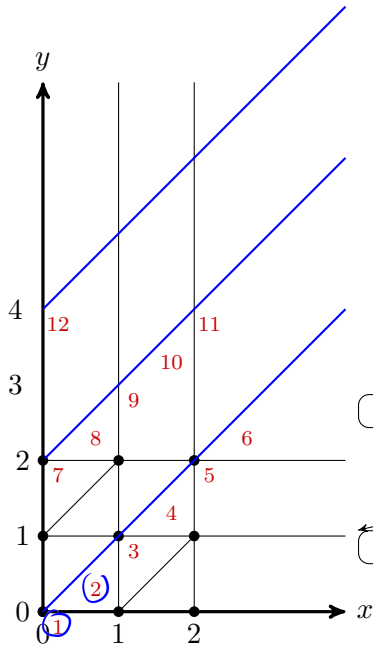
- $r = r' = r_\infty$, or
- $r \neq r_\infty$, $r \neq r'$, and for all $\nu \in r$:

$$\exists d \in \mathbb{R}_{>0}. (\nu + d \in r') \wedge \forall 0 \leq d' \leq d. \nu + d' \in r \cup r'.$$

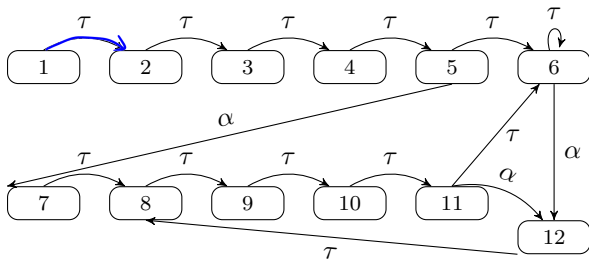
The **successor state region** is defined as $\mathit{succ}((l, r)) = (l, \mathit{succ}(r))$.

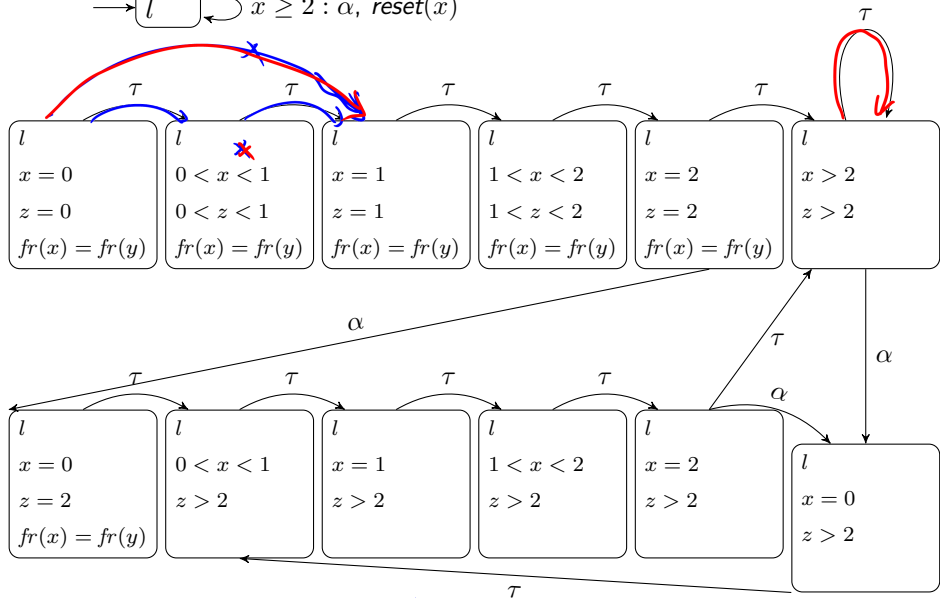
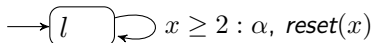


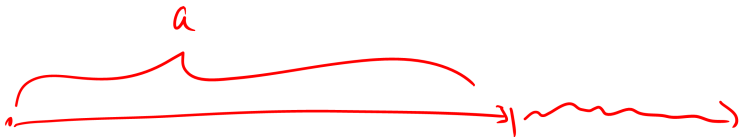
$$\mathbf{E}\mathcal{F}^{(0,2]} x = 0$$



$\mathbf{EF}^{(0,2]} x = 0$







\bar{f}^a

f^a

3. The abstract transition system

Definition

Let $\mathcal{T} = (Loc, \mathcal{C}, Lab, Edge, Inv, Init)$ be a non-zero timelock-free timed automaton with an atomic proposition set AP and a labeling function L , and let $\hat{\psi}$ be an unbounded TCTL formula over \mathcal{C} and AP .

The **region transition system** of \mathcal{T} for $\hat{\psi}$ is a labelled state transition system $\mathcal{RTS}(\mathcal{T}, \hat{\psi}) = (\Sigma', Lab', Edge', Init')$ with atomic propositions AP' and a labeling function L' such that

- Σ' the finite set of all state regions
- $\underline{Init}' = \{[\sigma] \mid \sigma \in Init\}$
- $\underline{AP}' = AP \cup ACC(\mathcal{T}) \cup ACC(\hat{\psi})$
- $\underline{L}'((l, r)) = L(l) \cup \{g \in AP' \setminus AP \mid r \models g\}$

and

3. The abstract transition system

Definition

$$\frac{\begin{array}{l} (l, a, (g, C), l') \in Edge \\ r \models \textcircled{g} \quad r' = \text{reset}(C) \text{ in } r \quad r' \models \underline{Inv}(l') \end{array}}{(l, \underline{r}) \xrightarrow{\textcircled{a}} (\underline{l'}, \underline{r'})} \quad \text{Rule}_{\text{Discrete}}$$
$$\frac{\left(\underline{r} \models \underline{Inv}(l) \right) \quad \text{succ}(r) \models \underline{Inv}(l)}{(\underline{l}, \underline{r}) \xrightarrow{t} (\underline{l}, \underline{\text{succ}(r)})} \quad \text{Rule}_{\text{Time}}$$

3. The abstract transition system

Lemma

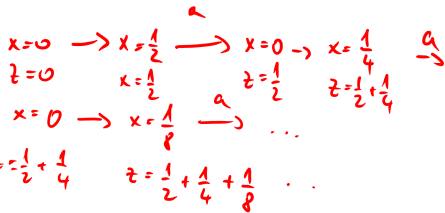
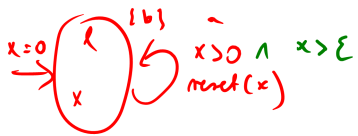
For non-zero \mathcal{T} and $\pi = s_0 \rightarrow s_1 \rightarrow \dots$ an initial, infinite path of \mathcal{T} :

- if π is time-convergent, then there is an index j and a state region (l, r) such that $s_i \in (l, r)$ for all $i \geq j$.
- if there is a state region (l, r) with $r \neq r_\infty$ and an index j such that $s_i \in (l, r)$ for all $i \geq j$ then π is time-convergent.

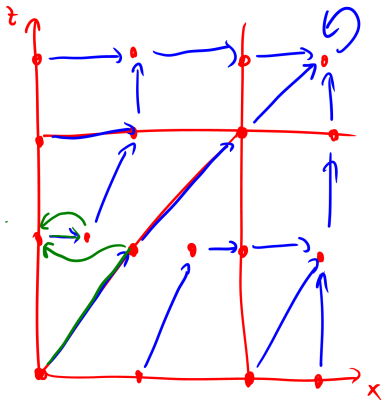
Lemma

For a non-zero timed automaton \mathcal{T} and a TCTL formula ψ :

$$\mathcal{T} \models_{TCTL} \psi \quad \text{iff} \quad \text{RTS}(\mathcal{T}, \hat{\psi}) \models_{CTL} \hat{\psi}$$



$A \bar{F} = 1 \quad b \rightarrow A \bar{F} (z=1 \wedge b)$





Input: timed automaton \mathcal{T} , TCTL formula ψ

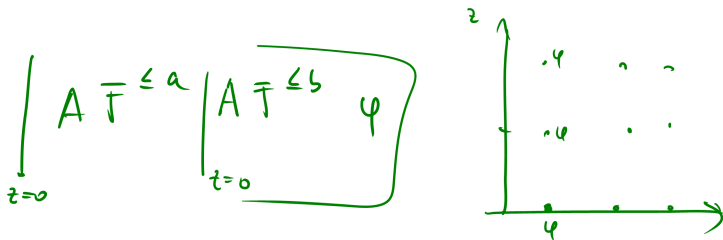
Output: the answer to the question if $\mathcal{T} \models \psi$

- 1 Eliminate the timing parameters from ψ , resulting in $\hat{\psi}$;
- 2 Make a finite abstraction of the state space
- 3 Construct abstract transition system RTS with $\mathcal{T} \models \psi$ iff $RTS \models \hat{\psi}$.
- 4 Apply CTL model checking to check whether $RTS \models \hat{\psi}$;
- 5 Return the model checking result.

The procedure is quite similar to CTL model checking for finite automata.

One difference:

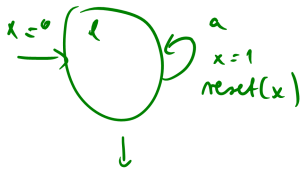
- Handling nested time bounds in TCTL formulae



Input: timed automaton \mathcal{T} , TCTL formula ψ

Output: the answer to the question if $\mathcal{T} \models \psi$

- 1 Eliminate the timing parameters from ψ , resulting in $\hat{\psi}$;
- 2 Make a finite abstraction of the state space
- 3 Construct abstract transition system RTS
 $\mathcal{T} \models \psi$ iff $RTS \models \hat{\psi}$
- 4 Apply CTL model checking to check whether $RTS \models \hat{\psi}$;
- 5 Return the model checking result.



\mathbb{R}
 \mathbb{Z}
 \mathbb{Z}
 \mathbb{R}

$$A \bar{f}^{(0,1]} (x=0)$$



$$A \bar{f} (\underbrace{z > 0 \wedge z \leq 1 \wedge x=0}_{\varphi_1})$$

φ_1



φ_2

$$\forall g \forall \bar{f}^{(0,1]} \quad x=0$$



$$\exists \bar{f} \exists \bar{f}^{(0,1]} \quad x=0$$



$\neg \varphi_1$



φ_3



φ_4

