

# Modeling and Analysis of Hybrid Systems

## Model checking timed automata

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems  
RWTH Aachen University

SS 2015

# Basic method: Abstraction

- **Given:** a **concrete** system  
(here: a timed automaton)
- **Goal:** **reduce the size** of the system by abstraction  
(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 by abstraction  
(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.**

# Basic method: Abstraction

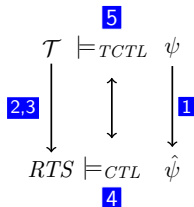
- **Given:** a **concrete** system  
(here: a **timed automaton**)
- **Goal:** **reduce the size** of the system by abstraction  
(here: **reduce the infinite state space to a finite one**)
- **Result:** **abstract** system  
(here: **region transition system**)
- **Conservative (safe) abstraction:**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

# TCTL model checking

**Input:** timed automaton  $\mathcal{T}$ , TCTL formula  $\psi$

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

- 1 Eliminate the timing parameters from the TCTL formula  $\psi$ , resulting in a CTL formula  $\hat{\psi}$  (with clock constraints as atomic propositions).
- 2 Make a finite abstraction of the state space of  $\mathcal{T}$ .
- 3 Construct abstract state transition system  $RTS$  (with the states labeled by clock constraints as atomic propositions) such that  $\mathcal{T} \models_{TCTL} \psi$  iff  $RTS \models_{CTL} \hat{\psi}$ .
- 4 Apply CTL model checking to check whether  $RTS \models_{CTL} \hat{\psi}$ .
- 5 Return the model checking result.

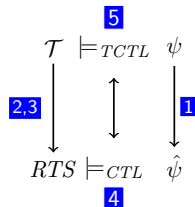


# TCTL model checking

**Input:** timed automaton  $\mathcal{T}$ , TCTL formula  $\psi$

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

- 1 Eliminate the timing parameters from the TCTL formula  $\psi$ , resulting in a CTL formula  $\hat{\psi}$  (with clock constraints as atomic propositions).
- 2 Make a finite abstraction of the state space of  $\mathcal{T}$ .
- 3 Construct abstract state transition system  $RTS$  (with the states labeled by clock constraints as atomic propositions) such that  $\mathcal{T} \models_{TCTL} \psi$  iff  $RTS \models_{CTL} \hat{\psi}$ .
- 4 Apply CTL model checking to check whether  $RTS \models_{CTL} \hat{\psi}$ .
- 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  $\sigma$  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  $\sigma$  of  $\mathcal{T}$  it holds that

$$\boxed{1} \quad \sigma \quad \models_{TCTL} \mathbf{E}(\psi_1 \quad \mathcal{U}^J \quad \psi_2) \text{ iff}$$

# 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  $\sigma$  of  $\mathcal{T}$  it holds that

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

# 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  $\sigma$  of  $\mathcal{T}$  it holds that

- $\sigma \models_{TCTL} \mathbf{E}(\psi_1 \ \mathcal{U}^J \ \psi_2)$  iff  
 $\text{reset}(z) \text{ in } \sigma \models_{TCTL} \mathbf{E}(\psi_1 \ \mathcal{U} \ ((z \in J) \wedge \psi_2))$ .
- $\sigma \models_{TCTL} \mathbf{A}(\psi_1 \ \mathcal{U}^J \ \psi_2)$  iff

# 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  $\sigma$  of  $\mathcal{T}$  it holds that

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

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

# 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  $\sigma$  of  $\mathcal{T}$  it holds that

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

# 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  $\sigma$  of  $\mathcal{T}$  it holds that

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

# 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  $\sigma$  of  $\mathcal{T}$  it holds that

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

# 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  $\sigma$  of  $\mathcal{T}$  it holds that

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

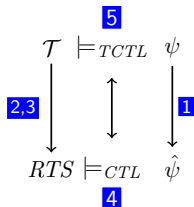


# TCTL model checking

**Input:** timed automaton  $\mathcal{T}$ , TCTL formula  $\psi$

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

- 1 Eliminate the timing parameters from the TCTL formula  $\psi$ , resulting in a CTL formula  $\hat{\psi}$  (with clock constraints as atomic propositions).
- 2 Make a finite abstraction of the state space of  $\mathcal{T}$ .
- 3 Construct abstract state transition system  $RTS$  (with the states labeled by clock constraints as atomic propositions) such that  $\mathcal{T} \models_{TCTL} \psi$  iff  $RTS \models_{CTL} \hat{\psi}$ .
- 4 Apply CTL model checking to check whether  $RTS \models_{CTL} \hat{\psi}$ .
- 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  $\psi'$  occurring in the timed automaton  $\mathcal{T}$  or in the specification  $\psi$ :

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

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

## Definition

Let  $LSTS = (\Sigma, Lab, Edge, Init)$  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  $\Sigma$ , and a time-abstract 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  $\alpha_i \in \text{Lab}$  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?

## 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,



## 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  $(l, \nu)$  and  $(l', \nu')$  satisfying  $l = l'$  should be equivalent.**

## 2. Finite state space abstraction

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

## 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  $r \in \mathbb{R}$ :  $\text{frac}(r) = r - \lfloor r \rfloor$

## 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:

$$\nu \models x < c \Leftrightarrow$$

## 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:

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

## 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:

$$\nu \models x \leq c \Leftrightarrow$$

## 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:

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

## 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  $(l, \nu)$  and  $(l, \nu')$  satisfying



## 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  $(l, \nu)$  and  $(l, \nu')$  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.

## 2. Finite state space abstraction

Problem: It would generate infinitely many equivalence classes!

## 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  $\psi$ . Then there is no observation which could distinguish between the  $x$ -values in  $(l, \nu)$  and  $(l, \nu')$  if  $\nu(x) > c_x$  and  $\nu'(x) > c_x$ .

## 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  $\psi$ . Then there is no observation which could distinguish between the  $x$ -values in  $(l, \nu)$  and  $(l, \nu')$  if  $\nu(x) > c_x$  and  $\nu'(x) > c_x$ .

**i.e., only states  $(l, \nu)$  and  $(l, \nu')$  satisfying**

## 2. Finite state space abstraction

**Problem:** It would generate infinitely many equivalence classes!

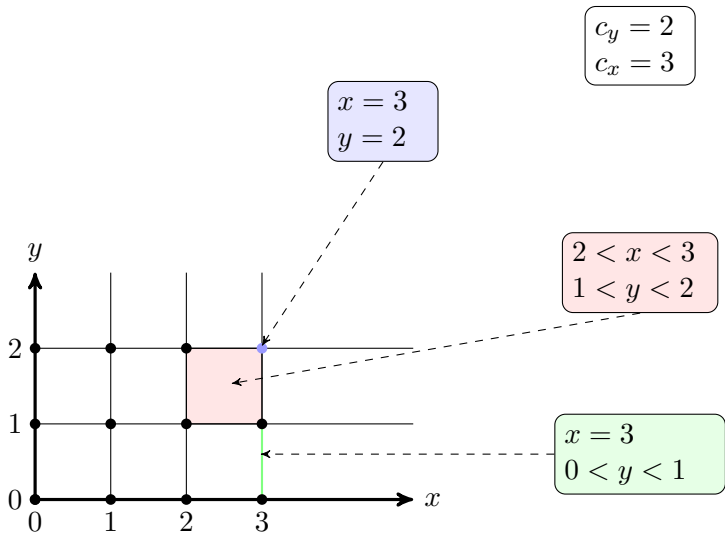
Let  $c_x$  be the largest constant which a clock  $x$  is compared to in  $\mathcal{T}$  or in  $\psi$ . Then there is no observation which could distinguish between the  $x$ -values in  $(l, \nu)$  and  $(l, \nu')$  if  $\nu(x) > c_x$  and  $\nu'(x) > c_x$ .

i.e., only states  $(l, \nu)$  and  $(l, \nu')$  satisfying

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

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

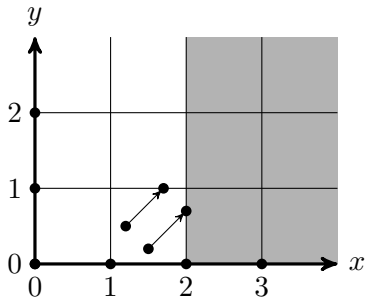
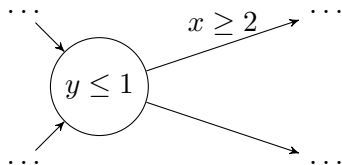
## 2. Finite state space abstraction



## 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  $(l, \nu)$  and  $(l, \nu')$  satisfying

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

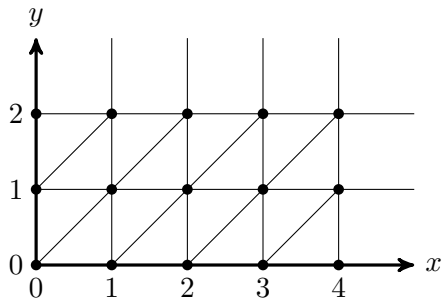
for all  $x, y \in \mathcal{C}$  should be equivalent.

Because of symmetry the following is also sufficient:

$$\begin{aligned} & (\nu(x), \nu'(x) > c_x \wedge \nu(y), \nu'(y) > c_y) \quad \vee \\ & (\text{frac}(\nu(x)) \leq \text{frac}(\nu(y)) \quad \text{iff} \quad \text{frac}(\nu'(x)) \leq \text{frac}(\nu'(y))) \end{aligned}$$

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  $\psi$ , 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  $\nu(y), \nu'(y) \leq c_y$  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  $[\sigma] = \{(l, \nu') \in \Sigma \mid \nu \cong \nu'\}$ .

## 2. Finite state space abstraction

### Lemma

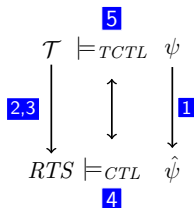
*Clock equivalence is a bisimulation over  $AP' = AP \cup ACC(\mathcal{T}) \cup ACC(\psi)$ .*

# TCTL model checking

**Input:** timed automaton  $\mathcal{T}$ , TCTL formula  $\psi$

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

- 1 Eliminate the timing parameters from the TCTL formula  $\psi$ , resulting in a CTL formula  $\hat{\psi}$  (with clock constraints as atomic propositions).
- 2 Make a finite abstraction of the state space of  $\mathcal{T}$ .
- 3 Construct abstract state transition system  $RTS$  (with the states labeled by clock constraints as atomic propositions) such that  $\mathcal{T} \models_{TCTL} \psi$  iff  $RTS \models_{CTL} \hat{\psi}$ .
- 4 Apply CTL model checking to check whether  $RTS \models_{CTL} \hat{\psi}$ .
- 5 Return the model checking result.



### 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

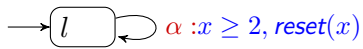
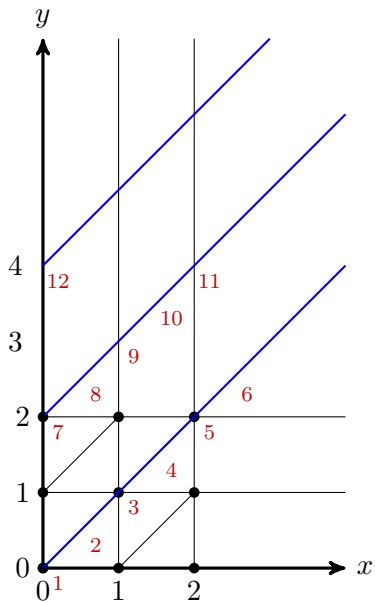
#### Definition

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  $\nu \in r$ :

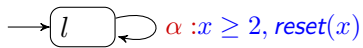
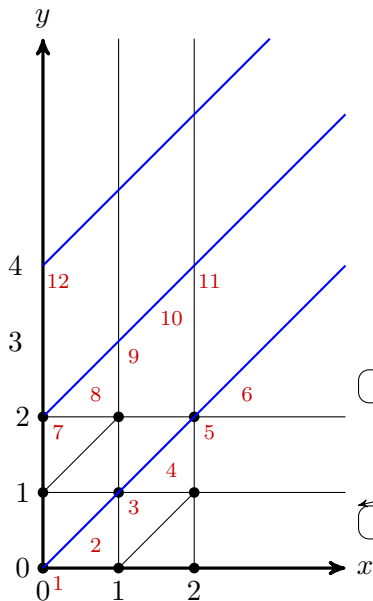
$$\exists d \in \mathbb{R}_{>0}. (\nu + d \in r' \wedge \forall 0 \leq d' \leq d. \nu + d' \in (r \cup r')).$$

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

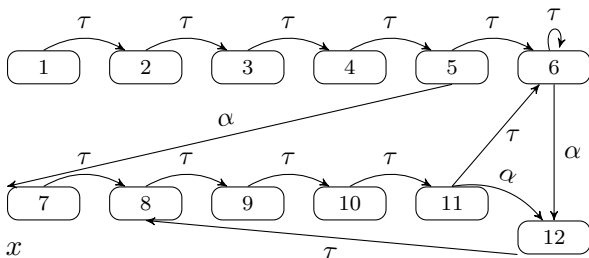


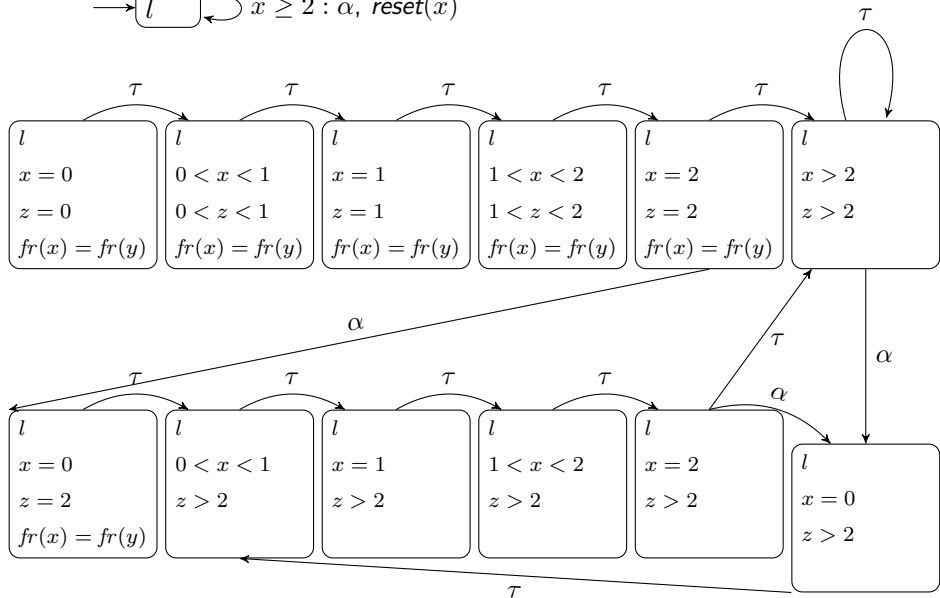
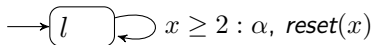
$\mathbf{EF}^{(0,2]} x = 0$





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





### 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

- $\Sigma' = \{(l, [\nu]) \mid (l, \nu) \in \Sigma \wedge \nu \in Inv(l)\}$
- $Init' = \{(l, [\nu]) \in \Sigma' \mid (l, \nu) \in Init\}$
- $AP' = AP \cup ACC(\mathcal{T}) \cup ACC(\hat{\psi})$
- $L'((l, [\nu])) = L(l) \cup \{g \in AP' \setminus AP \mid \nu \models g\}$

and

### 3. The abstract transition system

#### Definition

$$\frac{(l, \nu) \xrightarrow{a} (l', \nu')}{(l, [\nu]) \xrightarrow{a} (l', [\nu'])} \quad \text{Rule}_{\text{Discrete}}$$

$$\frac{\text{succ}(r) \models \text{Inv}(l)}{(l, r) \xrightarrow{t} (l, \text{succ}(r))} \quad \text{Rule}_{\text{Time}}$$

### 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  $\pi$  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  $\pi$  is time-convergent.

#### Lemma

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

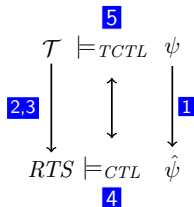
$$\mathcal{T} \models_{TCTL} \psi \quad \text{iff} \quad \text{RTS}(\mathcal{T}, \hat{\psi}) \models_{CTL} \hat{\psi}$$

# TCTL model checking

**Input:** timed automaton  $\mathcal{T}$ , TCTL formula  $\psi$

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

- 1 Eliminate the timing parameters from the TCTL formula  $\psi$ , resulting in a CTL formula  $\hat{\psi}$  (with clock constraints as atomic propositions).
- 2 Make a finite abstraction of the state space of  $\mathcal{T}$ .
- 3 Construct abstract state transition system  $RTS$  (with the states labeled by clock constraints as atomic propositions) such that  $\mathcal{T} \models_{TCTL} \psi$  iff  $RTS \models_{CTL} \hat{\psi}$ .
- 4 Apply CTL model checking to check whether  $RTS \models_{CTL} \hat{\psi}$ .
- 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

# TCTL model checking

**Input:** timed automaton  $\mathcal{T}$ , TCTL formula  $\psi$

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

- 1 Eliminate the timing parameters from the TCTL formula  $\psi$ , resulting in a CTL formula  $\hat{\psi}$  (with clock constraints as atomic propositions).
- 2 Make a finite abstraction of the state space of  $\mathcal{T}$ .
- 3 Construct abstract state transition system  $RTS$  (with the states labeled by clock constraints as atomic propositions) such that  $\mathcal{T} \models_{TCTL} \psi$  iff  $RTS \models_{CTL} \hat{\psi}$ .
- 4 Apply CTL model checking to check whether  $RTS \models_{CTL} \hat{\psi}$ .
- 5 Return the model checking result.

