Model Checking Timed Automata

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen

SS09

TCTL model checking

Input: timed automaton TA, TCTL formula Φ
Output: the answer to the question if $TA \models \Phi$

1. Eliminate the timing parameters from Φ, resulting in $\hat{\Phi}$;
2. Make a finite partition of the state space
   $\rightarrow$ finite abstract state space
3. Construct abstract transition system RTS;
4. Apply CTL model checking to check whether $RTA \models \hat{\Phi}$;
5. $TA \models \Phi$ iff $RTA \models \hat{\Phi}$.

TCTL model checking

Input: timed automaton TA, TCTL formula Φ
Output: the answer to the question if $TA \models \Phi$

1. Eliminate the timing parameters from Φ, resulting in $\hat{\Phi}$;
2. Make a finite partition of the state space
   $\rightarrow$ finite abstract state space
3. Construct abstract transition system RTS;
4. Apply CTL model checking to check whether $RTA \models \hat{\Phi}$;
5. $TA \models \Phi$ iff $RTA \models \hat{\Phi}$.

1. Eliminating timing parameters

Let TA be a timed automaton with clock set $\mathcal{C}$ and atomic propositions AP, and $\Phi\ \mathcal{U}^J\ \Psi$ a TCTL formula over $\mathcal{C}$ and AP. Let $TA' = TA \oplus z$ result from TA by adding a fresh clock which never gets reset.

For any state σ of TA it holds that

1. $$\begin{array}{lll} \sigma & \models_{TCTL} & \exists(\Phi \quad \mathcal{U}^J \quad \Psi) \textit{ iff} \\ \sigma\{z := 0\} & \models_{TCTL} & \exists((\Phi \vee \Psi) \quad \mathcal{U} \quad ((z \in J) \wedge \Psi)\,. \end{array}$$

2. $$\begin{array}{lll} \sigma & \models_{TCTL} & \forall(\Phi \quad \mathcal{U}^J \quad \Psi) \textit{ iff} \\ \sigma\{z := 0\} & \models_{TCTL} & \forall((\Phi \vee \Psi) \quad \mathcal{U} \quad ((z \in J) \wedge \Psi)\,. \end{array}$$

1. Eliminating timing parameters: Examples

1. $\exists \Diamond^{\leq 2} \Phi \quad \rightsquigarrow \quad \exists \Diamond ((z \leq 2) \wedge \Phi)$

2. $\exists \Box^{\leq 2} \Phi \quad = \quad \neg \forall \Diamond^{\leq 2} \neg \Phi$
   $\rightsquigarrow \quad \neg \forall \Diamond ((z \leq 2) \wedge \neg \Phi)$
   $= \quad \exists \Box (\neg (z \leq 2) \vee \Phi)$
   $= \quad \exists \Box ((z \leq 2) \rightarrow \Phi)$

3. $\forall (\Phi \; \mathcal{U}^J \; \Psi) \quad \rightsquigarrow \quad \forall \left((\Phi \vee \Psi) \; \mathcal{U} \; ((z \in J) \wedge \Psi) \right.$

TCTL model checking

Input: timed automaton TA, TCTL formula Φ
Output: the answer to the question if $TA \models \Phi$

1. Eliminate the timing parameters from Φ, resulting in $\hat{\Phi}$;
2. Make a finite partition of the state space
   $\rightarrow$ finite abstract state space
3. Construct abstract transition system RTS;
4. Apply CTL model checking to check whether $RTA \models \hat{\Phi}$;
5. $TA \models \Phi$ iff $RTA \models \hat{\Phi}$.

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 on states, such that equivalent states satisfy the same (sub)formulae Ψ occurring in the timed automaton TA or in the specification Φ:

$$\sigma \cong \sigma' \quad \Rightarrow \quad \left(\sigma \models \Psi \quad \textit{iff} \quad \sigma' \models \Psi\right).$$

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

Bisimulation as state relation

Definition

Let TS be a transition system with initial state set $Init$ and labeling function L over the proposition set AP. A *bisimulation* $\mathcal{R} \subseteq \Sigma \times \Sigma$ is a binary relation on the state set Σ such that for all $(\sigma_1, \sigma_2) \in \mathcal{R}$ with $\sigma_i = (l_i, \nu_i)$ it holds:

- $L(l_1) = L(l_2)$
- for all σ_1' with $\sigma_1 \xrightarrow{a} \sigma_1'$ there exists σ_2' such that $\sigma_2 \xrightarrow{a} \sigma_2'$ and $(\sigma_1', \sigma_2') \in \mathcal{R}$
- for all σ_2' with $\sigma_2 \xrightarrow{a} \sigma_2'$ there exists σ_1' such that $\sigma_1 \xrightarrow{a} \sigma_1'$ and $(\sigma_1', \sigma_2') \in \mathcal{R}$.

We write $\sigma_1 \cong \sigma_2$ for $(\sigma_1, \sigma_2) \in \mathcal{R}$.

Bisimulation

Lemma

Given a transition system TA with state space Σ, a bisimulation $\mathcal{R} \subseteq \Sigma \times \Sigma$ is an equivalence relation on Σ.
Furthermore, if $\sigma \cong \sigma'$, then for each path

$$\pi : \sigma \xrightarrow{a_1} \sigma_1 \xrightarrow{a_2} \sigma_2 \xrightarrow{a_3} \dots$$

of TA there exists a path

$$\pi' : \sigma' \xrightarrow{a_1} \sigma'_1 \xrightarrow{a_2} \sigma'_2 \xrightarrow{a_3} \dots$$

of TA of the same length such that $\sigma_i \cong \sigma'_i$ for all i.

2. Finite state space abstraction

Now, back to timed automata. How could an equivalence relation look like?

Since, in general,

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

different locations in TA 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 $r \in \mathbb{R}$: $frac(r) = r - \lfloor r \rfloor$

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:

$$\nu \models x \leq c \;\Leftrightarrow\; \nu(x) \leq c \;\Leftrightarrow\; \lfloor \nu(x) \rfloor < c \vee (\lfloor \nu(x) \rfloor = c \wedge frac(\nu(x)) = 0)\,.$$

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

$$\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor \; and \; frac(\nu(x)) = 0 \; iff \; frac(\nu'(x)) = 0$$

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

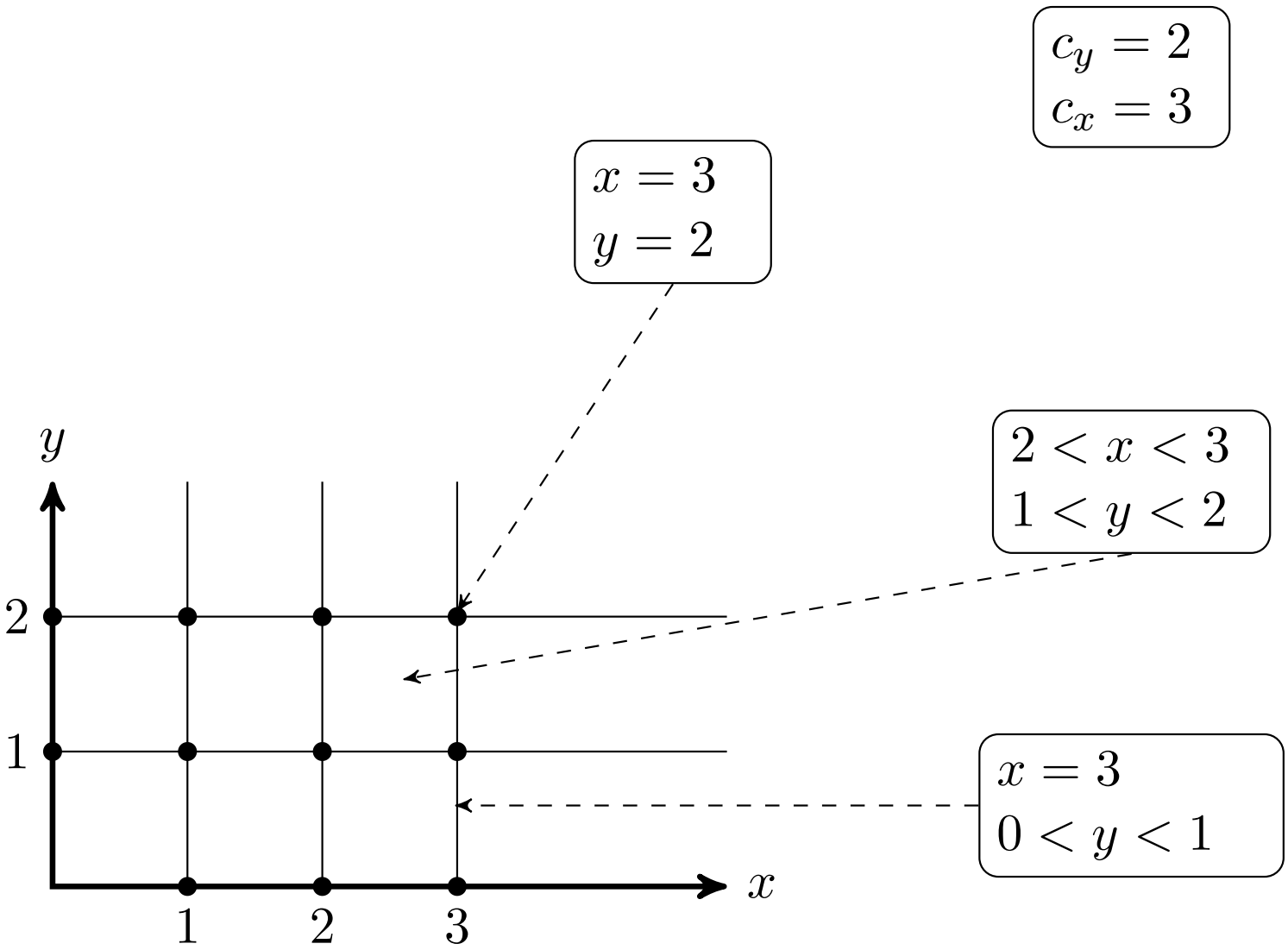
Problem: it would generate infinitely many equivalence classes!

Let c_x be the largest constant which a clock x is compared to in TA or in Φ. 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

$$(\nu(x) > c_x \wedge \nu'(x) > c_x) \quad \vee$$
$$\big(\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor \ \wedge \ frac(\nu(x)) = 0 \ \textit{iff} \ frac(\nu'(x)) = 0\big)$$

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

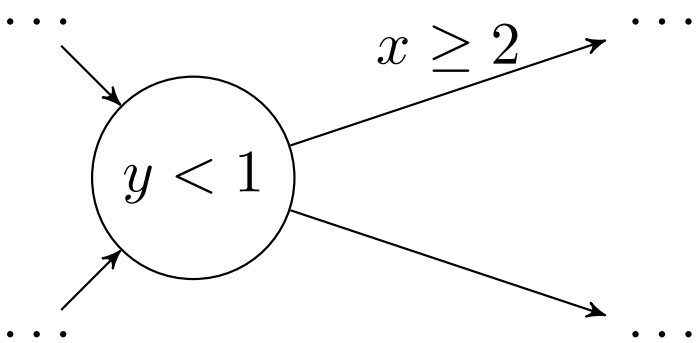
2. Finite state space abstraction

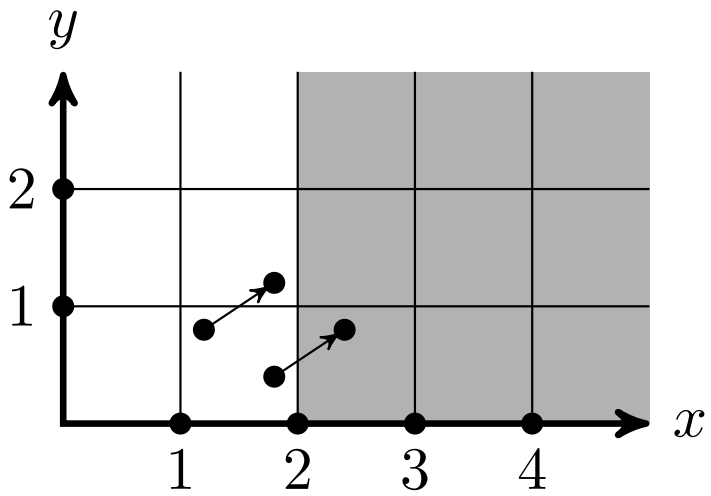


2. Finite state space abstraction

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

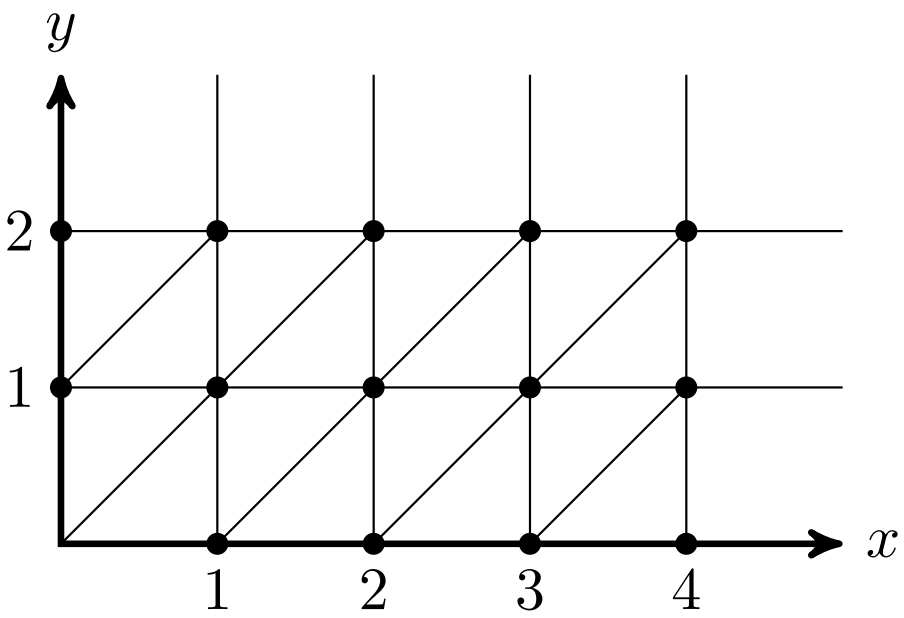
$$\begin{aligned}
&\big(\nu(x) > c_x \wedge \nu'(x) > c_x\big) \quad \vee \\
&\qquad (frac(\nu(x)) < frac(\nu(y)) \quad iff \quad frac(\nu'(x)) < frac(\nu'(y)) \quad \wedge \\
&\qquad\ frac(\nu(x)) = frac(\nu(y)) \quad iff \quad frac(\nu'(x)) = frac(\nu'(y)) \quad \wedge \\
&\qquad\ frac(\nu(x)) > frac(\nu(y)) \quad iff \quad frac(\nu'(x)) > frac(\nu'(y))\big)
\end{aligned}$$

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

Because of symmetry the following is also sufficient:

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

2. Finite state space abstraction



$c_y = 2$
$c_x = 4$

finite index

Definition

For a timed automaton TA 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

$$\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor \ \wedge \ (frac(\nu(x)) = 0 \quad iff \quad frac(\nu'(x)) = 0)$$

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

$$frac(\nu(x)) \leq frac(\nu(y)) \quad iff \quad frac(\nu'(x)) \leq frac(\nu'(y)).$$

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

Lemma

Clock equivalence is a bisimulation over $AP' = AP \cup ACC(TA) \cup ACC(\Phi)$.

TCTL model checking

Input: timed automaton TA, TCTL formula Φ
Output: the answer to the question if $TA \models \Phi$

1. Eliminate the timing parameters from Φ, resulting in $\hat{\Phi}$;
2. Make a finite partition of the state space
   $\rightarrow$ finite abstract state space
3. Construct abstract transition system RTS;
4. Apply CTL model checking to check whether $RTA \models \hat{\Phi}$;
5. $TA \models \Phi$ iff $RTA \models \hat{\Phi}$.

3. The abstract transition system

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

Two kinds of transitions:
time and discrete

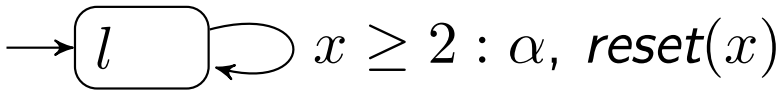
Definition

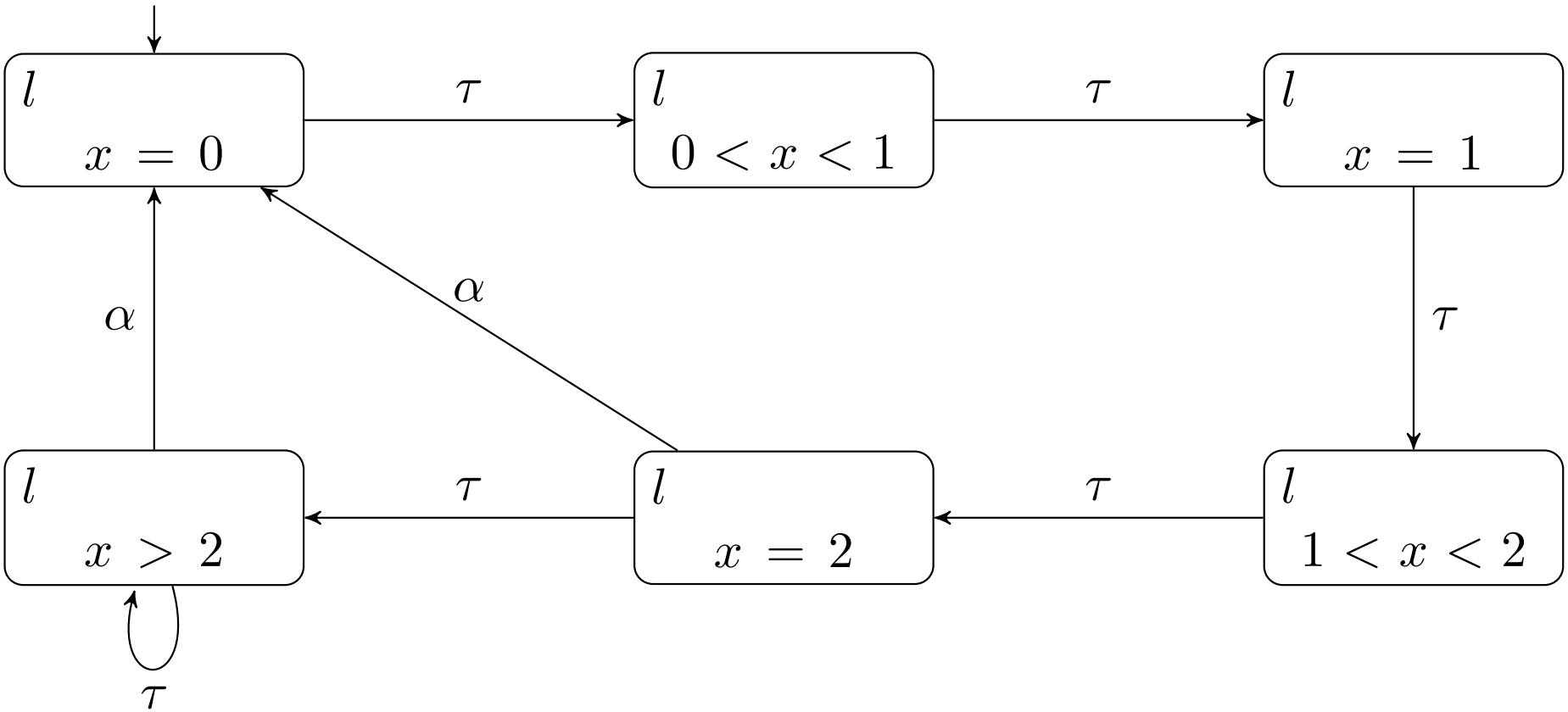
The clock region $r_\infty = \{\nu \in V \mid \forall x \in \mathcal{C}.\ \nu(x) > c_x\}$ is called unbounded. Let r, r' be two clock regions. The region r' is the successor clock region of r, denoted by $r' = succ(r)$, if either

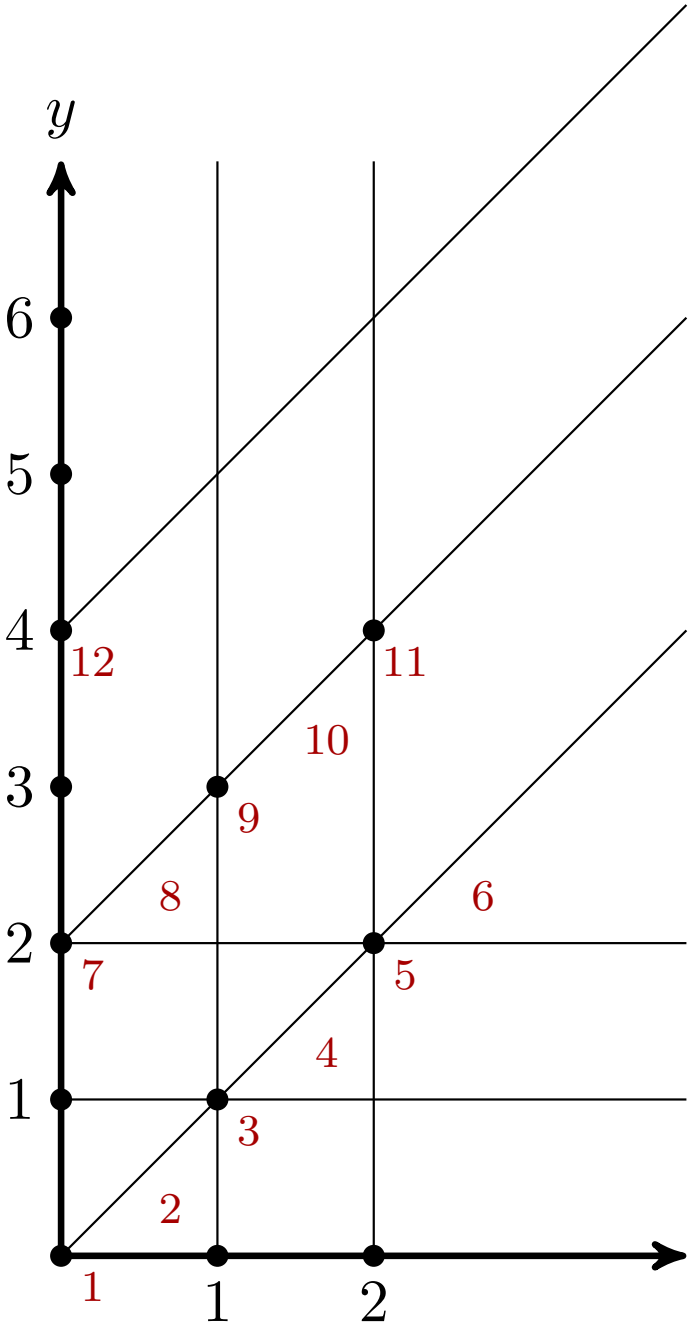
- $r = r' = r_\infty$, or
- $r \neq r_\infty$, $r' \neq r_\infty$, and for all $\nu \in r$:

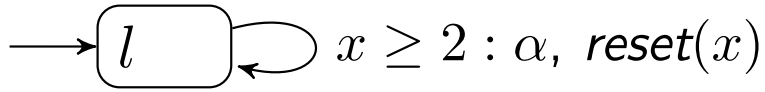
$$\exists d \in \mathbb{R}_{>0}.\ (\nu + d \in r' \ \wedge\ \forall 0 \leq d' \leq d.\ \nu + d' \in r \cup r').$$

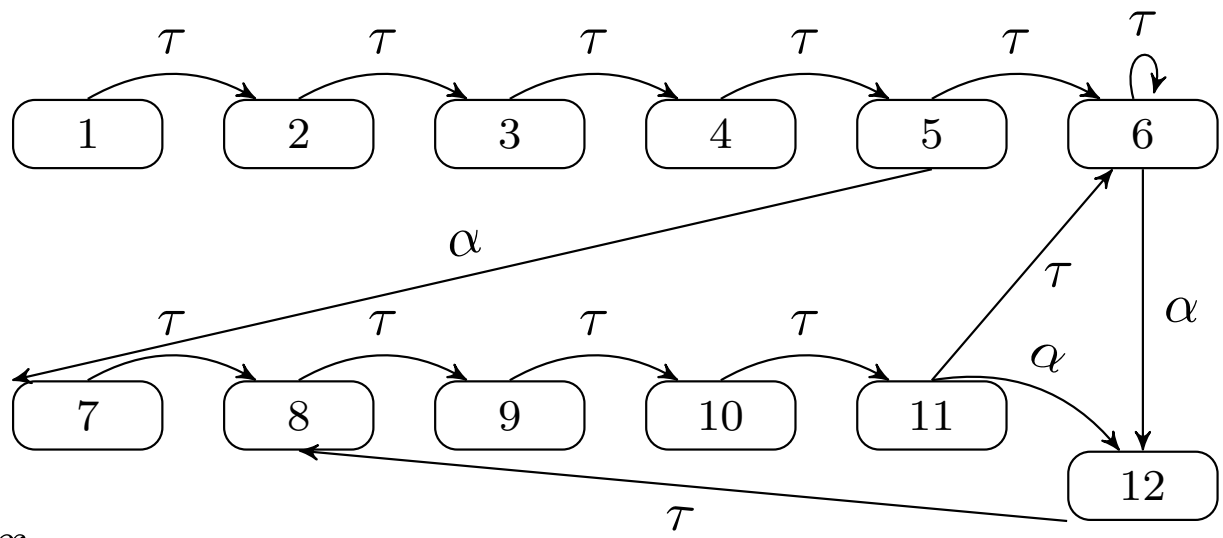
The successor state region is defined as $succ((l, r)) = (l, succ(r))$.

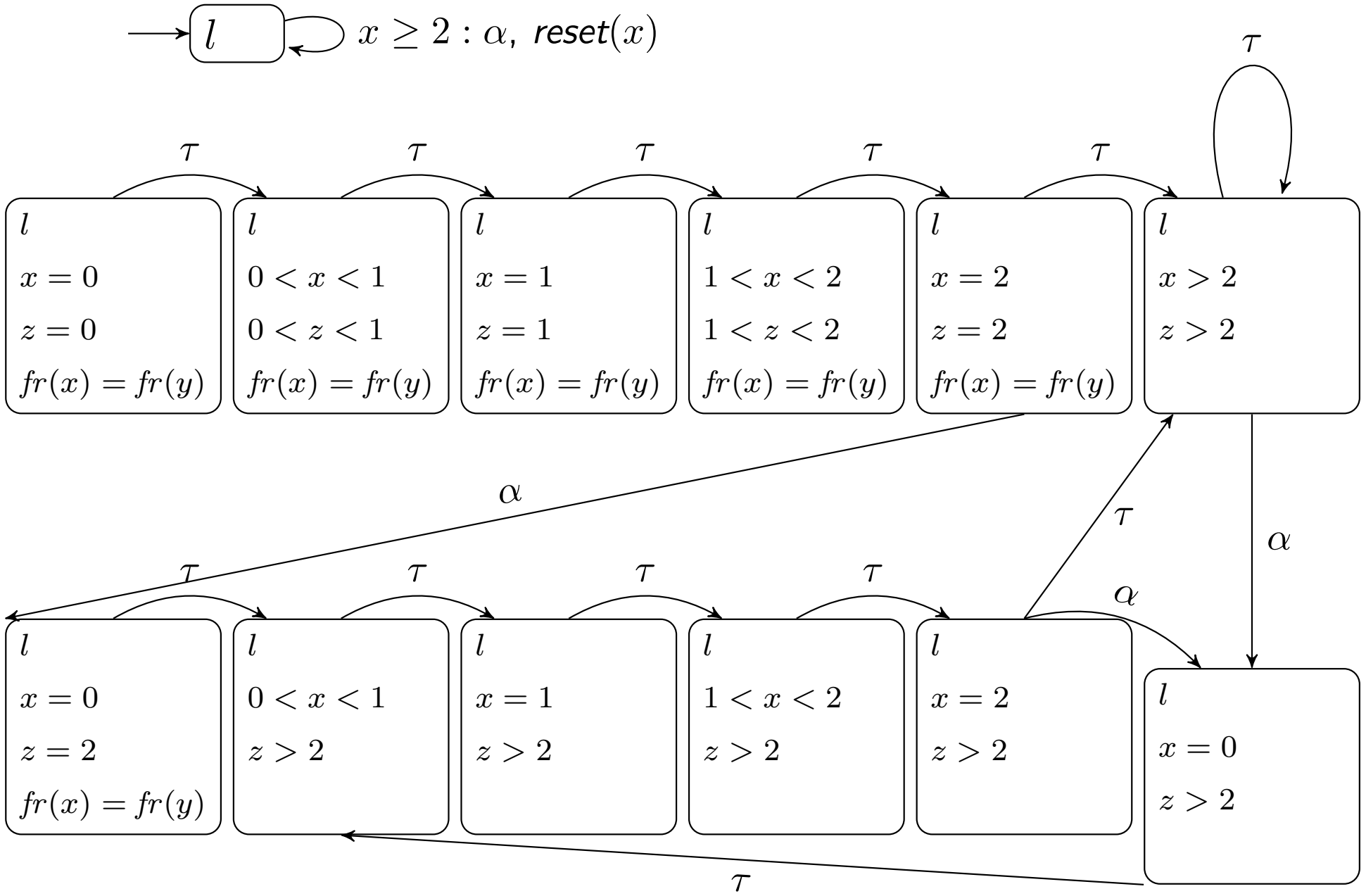

l
$x \geq 2 : \alpha,\ \mathit{reset}(x)$


l
$x = 0$
τ
l
$0 < x < 1$
τ
l
$x = 1$
τ
α
α
l
$x > 2$
τ
l
$x = 2$
τ
l
$1 < x < 2$
τ


y
6
5
4
3
2
1
12
11
10
9
8
6
7
5
4
3
2
1
1
2
x


l
$x \geq 2 : \alpha,\ reset(x)$


τ
τ
τ
τ
τ
τ
1
2
3
4
5
6
α
τ
α
τ
τ
τ
τ
α
7
8
9
10
11
12
τ


l
$x \geq 2 : \alpha,\ reset(x)$
τ
l $x = 0$ $z = 0$ $fr(x) = fr(y)$
l $0 < x < 1$ $0 < z < 1$ $fr(x) = fr(y)$
l $x = 1$ $z = 1$ $fr(x) = fr(y)$
l $1 < x < 2$ $1 < z < 2$ $fr(x) = fr(y)$
l $x = 2$ $z = 2$ $fr(x) = fr(y)$
l $x > 2$ $z > 2$
α
τ
α
τ
α
l $x = 0$ $z = 2$ $fr(x) = fr(y)$
l $0 < x < 1$ $z > 2$
l $x = 1$ $z > 2$
l $1 < x < 2$ $z > 2$
l $x = 2$ $z > 2$
l $x = 0$ $z > 2$
τ

3. The abstract transition system

Definition

Let $\mathcal{TA} = (Loc, \mathcal{C}, Lab, Edge, Inv, Init)$ be a non-zeno timed automaton and let Φ be a TCTL$_\Diamond$ formula. The region transition system of TA for Φ is a labelled state transition system
$\mathcal{RTS}(TA, \Phi) = (\Sigma', Lab', \Delta', \Sigma'_0, AP', L')$ with

- Σ' the finite set of all state regions
- $\Sigma'_0 = \{[\sigma] \mid \sigma \in Init\}$
- $AP' = AP \cup ACC(TA) \cup ACC(\Phi)$
- $L'((l, r)) = L(l) \cup \{g \in AP' \backslash AP \mid r \models g\}$

and

Definition

$$\frac{(l, a, (g, \mathcal{R}), l') \in Edge \qquad r \models g \quad r' = \textit{reset } \mathcal{R} \textit{ in } r \quad r' \models Inv(l')}{(l, [\nu]) \xrightarrow{a} (l', [\nu'])} \quad \texttt{Rule}_{\texttt{Discrete}}$$

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

Lemma

For non-zeno TA and $\pi = s_0 \to s_1 \to \ldots$ an initial, infinite path of TA:

- *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-zeno timed automaton TA and a TCLT$_\Diamond$ formula Φ:

$$TA \models_{TCTL} \Phi \quad \textit{iff} \quad \mathcal{RTS}(TA, \Phi) \models_{CTL} \Phi$$

3. The abstract transition system

Questions:

- We have seen, that clock equivalence is a bisimulation equivalence over AP'.
  But: Is the region transition system bisimilar to the (transition system of the) timed automaton?
- Are they stutter-equivalent?
- What is the consequence if we add self-loops on regions?
- Could we add delay transitions connecting non-successor regions to represent longer time steps?

TCTL model checking

Input: timed automaton TA, TCTL formula Φ
Output: the answer to the question if $TA \models \Phi$

1. Eliminate the timing parameters from Φ, resulting in $\hat{\Phi}$;
2. Make a finite partition of the state space
   $\rightarrow$ finite abstract state space
3. Construct abstract transition system RTS;
4. Apply CTL model checking to check whether $RTA \models \hat{\Phi}$;
5. $TA \models \Phi$ iff $RTA \models \hat{\Phi}$.

TCTL model checking

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 TA, TCTL formula Φ
Output: the answer to the question if $TA \models \Phi$

1. Eliminate the timing parameters from Φ, resulting in $\hat{\Phi}$;
2. Make a finite partition of the state space
   $\rightarrow$ finite abstract state space
3. Construct abstract transition system RTS;
4. Apply CTL model checking to check whether $RTA \models \hat{\Phi}$;
5. $TA \models \Phi$ iff $RTA \models \hat{\Phi}$.