Modeling and analysis of hybrid systems

Model checking timed automata

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen

SS 2010

TCTL model checking

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.

TCTL model checking

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

1

$$\begin{array}{llllll} \sigma & \models_{TCTL} & \exists(\psi_1 & \mathcal{U}^J & \psi_2) & \textit{iff} \\ \textit{reset}(z) \textit{ in } \sigma & \models_{TCTL} & \exists\left((\psi_1 \vee \psi_2)\right. & \mathcal{U} & \left.((z \in J) \wedge \psi_2)\right). & \end{array}$$

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. $\sigma \models_{TCTL} \exists(\psi_1 \, \mathcal{U}^J \, \psi_2)$ *iff*
   $\textit{reset}(z) \textit{ in } \sigma \models_{TCTL} \exists\left((\psi_1 \vee \psi_2) \, \mathcal{U} \, ((z \in J) \wedge \psi_2)\right).$

2. $\sigma \models_{TCTL} \forall(\psi_1 \, \mathcal{U}^J \, \psi_2)$ *iff*
   $\textit{reset}(z) \textit{ in } \sigma \models_{TCTL} \forall\left((\psi_1 \vee \psi_2) \, \mathcal{U} \, ((z \in J) \wedge \psi_2)\right).$

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. $\sigma \models_{TCTL} \exists(\psi_1 \, \mathcal{U}^J \, \psi_2)$ *iff* $reset(z) \text{ in } \sigma \models_{TCTL} \exists\left((\psi_1 \vee \psi_2) \, \mathcal{U} \, ((z \in J) \wedge \psi_2\right).$
2. $\sigma \models_{TCTL} \forall(\psi_1 \, \mathcal{U}^J \, \psi_2)$ *iff* $reset(z) \text{ in } \sigma \models_{TCTL} \forall\left((\psi_1 \vee \psi_2) \, \mathcal{U} \, ((z \in J) \wedge \psi_2\right).$
3. $\sigma \models_{TCTL} \exists\mathcal{F}^{\leq 2}\psi_1$ *iff* $reset(z) \text{ in } \sigma \models_{TCTL} \exists\mathcal{F}((z \leq 2) \wedge \psi_1)$

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

1. $$\begin{array}{lll} \sigma & \models_{TCTL} & \exists(\psi_1 \; \mathcal{U}^J \; \psi_2) \textit{ iff} \\ \textit{reset}(z) \textit{ in } \sigma & \models_{TCTL} & \exists((\psi_1 \vee \psi_2) \; \mathcal{U} \; ((z \in J) \wedge \psi_2)\,. \end{array}$$
2. $$\begin{array}{lll} \sigma & \models_{TCTL} & \forall(\psi_1 \; \mathcal{U}^J \; \psi_2) \textit{ iff} \\ \textit{reset}(z) \textit{ in } \sigma & \models_{TCTL} & \forall((\psi_1 \vee \psi_2) \; \mathcal{U} \; ((z \in J) \wedge \psi_2)\,. \end{array}$$
3. $\sigma \models_{TCTL} \exists\mathcal{F}^{\leq 2}\psi_1 \quad \textit{iff} \quad \textit{reset}(z) \textit{ in } \sigma \models_{TCTL} \exists\mathcal{F}((z \leq 2) \wedge \psi_1)$
4. $\sigma \models_{TCTL} \exists\mathcal{G}^{\leq 2}\psi_1 \quad \textit{iff} \quad \textit{reset}(z) \textit{ in } \sigma \models_{TCTL} \exists\mathcal{G}((z \leq 2) \rightarrow \psi_1)$

TCTL model checking

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:

$$\sigma \cong \sigma' \quad \Rightarrow \quad (\sigma \models \psi' \quad \textit{iff} \quad \sigma' \models \psi')\,.$$

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

Bisimulation for two LSTSs

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 $L_1 : \Sigma_1 \to 2^{AP}$ and $L_2 : \Sigma_2 \to 2^{AP}$ 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 $\sigma_1' \approx \sigma_2'$.

Bisimulation for a single LSTS

Definition

Let $LSTS = (\Sigma, Lab, Edge, Init)$ be a state transition system, AP a set of atomic propositions, and $L : \Sigma \to 2^{AP}$ 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 $\sigma_1 \stackrel{a}{\to} \sigma_1'$ there exists $\sigma_2' \in \Sigma$ such that $\sigma_2 \stackrel{a}{\to} \sigma_2'$ and $\sigma_1' \approx \sigma_2'$.

Time abstract bisimulation

Definition

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

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 $\sigma_1' \approx \sigma_2'$.

Bisimulation

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

$$\pi : \sigma \stackrel{\alpha_1}{\rightarrow} \sigma_1 \stackrel{\alpha_2}{\rightarrow} \sigma_2 \stackrel{\alpha_3}{\rightarrow} \ldots$$

of $\mathcal{T}$ there exists a path

$$\pi' : \sigma' \stackrel{\alpha'_1}{\rightarrow} \sigma'_1 \stackrel{\alpha'_2}{\rightarrow} \sigma'_2 \stackrel{\alpha'_3}{\rightarrow} \ldots$$

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

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

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

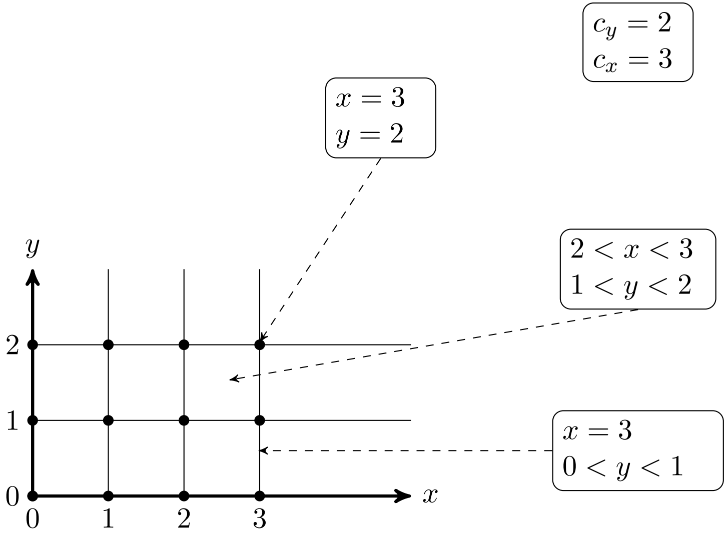
for all $x \in \mathcal{C}$ should be equivalent.

Problem: It would generate infinitely many equivalence classes!

Let c_x be the largest constant which a clock x is compared to in $\mathcal{T}$ 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

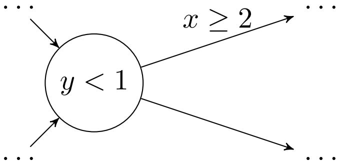
$$
(\nu(x) > c_x \wedge \nu'(x) > c_x) \quad \vee \\
\big(\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor \ \wedge \ \mathit{frac}(\nu(x)) = 0 \ \mathit{iff} \ \mathit{frac}(\nu'(x)) = 0\big)
$$

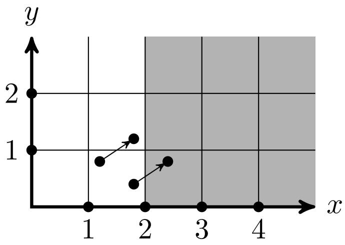
for all $x \in \mathcal{C}$ should be equivalent.


$c_y = 2$
$c_x = 3$
$x = 3$
$y = 2$
$2 < x < 3$
$1 < y < 2$
$x = 3$
$0 < y < 1$
y
x
2
1
0
0
1
2
3

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





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

$$\begin{array}{llll} (\nu(x), \nu'(x) > c_x \wedge \nu(y), \nu'(y) > c_x) & \vee & & \\ (\quad \mathit{frac}(\nu(x)) < \mathit{frac}(\nu(y)) & \mathit{iff} & \mathit{frac}(\nu'(x)) < \mathit{frac}(\nu'(y)) & \wedge \\ \quad \mathit{frac}(\nu(x)) = \mathit{frac}(\nu(y)) & \mathit{iff} & \mathit{frac}(\nu'(x)) = \mathit{frac}(\nu'(y)) & \wedge \\ \quad \mathit{frac}(\nu(x)) > \mathit{frac}(\nu(y)) & \mathit{iff} & \mathit{frac}(\nu'(x)) > \mathit{frac}(\nu'(y))) & \end{array}$$

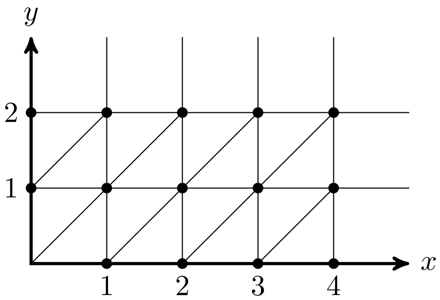
for all $x, y \in \mathcal{C}$ should be equivalent.

Because of symmetry the following is also sufficient:

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

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

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

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

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

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 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 abtraction 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.

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

Two kinds of transitions:
time and discrete

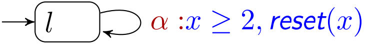
Definition

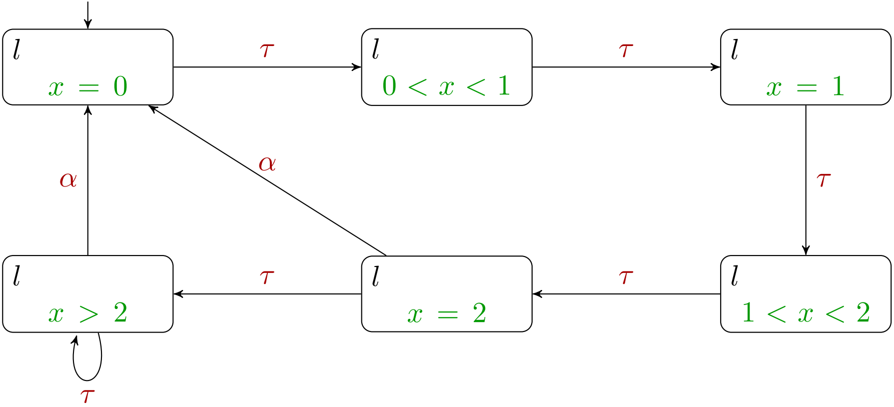
The clock region $r_\infty = \{\nu \in V \mid \forall x \in \mathcal{C}.\ \nu(x) > c_x\}$ is called unbounded.
Let r, r' be two clock regions. The region r' is the successor clock region of r, denoted by $r' = succ(r)$, if either

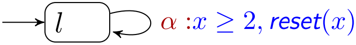
- $r = r' = r_\infty$, or
- $r \neq r_\infty$, $r \neq r'$, and for all $\nu \in r$:

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

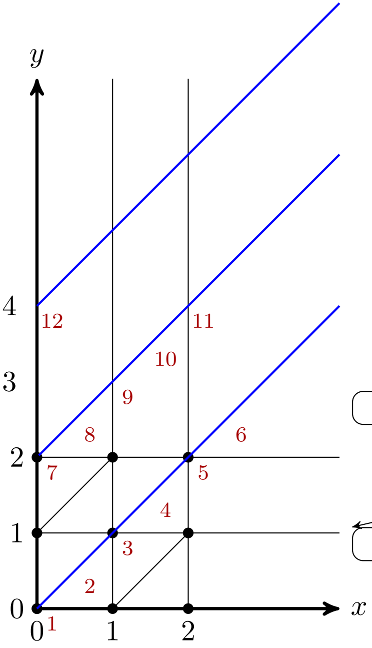
The successor state region is defined as $succ((l, r)) = (l, succ(r))$.

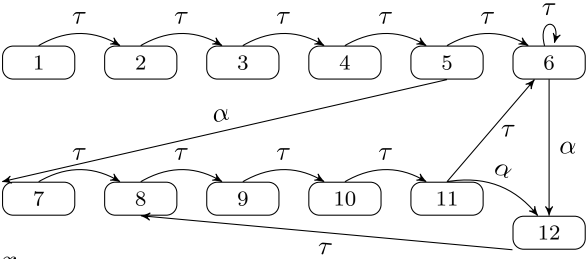

l
α :x ≥ 2, reset(x)

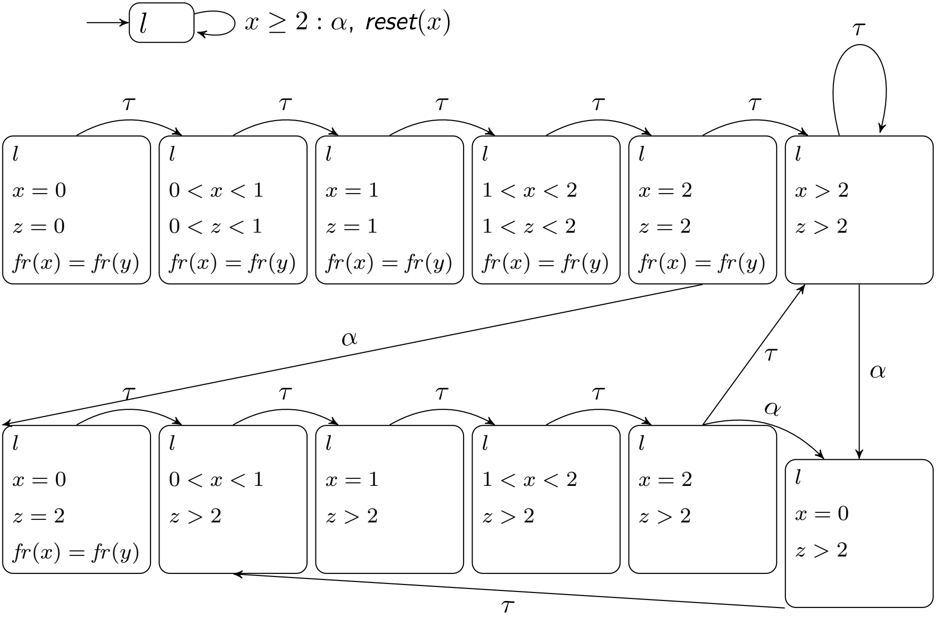

l
x = 0
τ
l
0 < x < 1
τ
l
x = 1
τ
l
1 < x < 2
τ
l
x = 2
τ
l
x > 2
τ
α
α



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







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
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 $x = 0$ $z > 2$

Definition

Let $\mathcal{T} = (Loc, \mathcal{C}, Lab, Edge, Inv, Init)$ be a non-zeno 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 $\mathcal{RTS}(\mathcal{T}, \psi) = (\Sigma', Lab', Edge', Init')$ with atomic propositions AP' and a labeling function L' such that

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

and

Definition

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

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

3. The abstract transition system

Lemma

For non-zeno $\mathcal{T}$ and $\pi = s_0 \to s_1 \to \ldots$ 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-zeno timed automaton $\mathcal{T}$ and a TCLT formula ψ:

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

TCTL model checking

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.

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 $\mathcal{T}$, 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.