Cyber-Physical Systems – Timed Automata

Matthias Althoff

TU München

05. June 2015

Overview

Timed Automata

- Modeling
- Runs of timed automata
- Verification of timed automata

Timed automata are already hybrid systems since time is real-valued.

Why include time into finite state automata?

- In finite state automata, only the sequence of events is important, not the time when it happens.
- In most engineering applications, timing is crucial, e.g.
  - time when a railway signal is changed,
  - coordination and timing of production processes,
  - scheduling of unmanned subway trains, etc.
- In some cases, wrong timing can lead to catastrophic events, e.g.
  - the system might become unstable,
  - a different event sequence is generated,
  - a deadlock is created, etc.

Timed events

Time sequence

We introduce a time sequence

$$\tau = t_1, t_2, t_3, \dots$$

where $t_i \leq t_{i+1}$ and $\forall t \in \mathbb{R} \quad \exists i \geq 1$ such that $t_i > t$ (time progresses).

Timed event

We refer to a pair (e_i, t_i) as a timed event, where e_i is the event occurring at time t_i.

Timed sequence

We refer to a sequence of timed events

$$\big((e_1, t_1), (e_2, t_2), (e_3, t_3), \dots\big)$$

as a timed sequence.

Timed Automata

Timed automata describe the dynamics of systems that can be described by a finite set of states z_i and clocks $c_i \in \mathbb{R}_0^+$. Starting from an initial state $z(t_0)$, initial clock values $c_i(t_0)$, and a timed input sequence

$$\bar{\mathbf{u}} = \big((u(t_0), t_0), (u(t_1), t_1), (u(t_2), t_2), \dots\big)$$

a finite state automaton creates a timed output sequence

$$\bar{\mathbf{y}} = \big((y(\tilde{t}_0), \tilde{t}_0), (y(\tilde{t}_1), \tilde{t}_1), (y(\tilde{t}_2), \tilde{t}_2), \dots\big),$$

where it is not required that the times t_i and $\tilde{t}_i$ are synchronized.

- For simplification we use the notation $u(k)$, $y(k)$ instead of $u(t_k)$, $y(t_k)$.
- Static timed automata $y(t_k) = g(u(t_k))$ have synchronized inputs and outputs and can as well be described by finite state automata.

Syntax of Timed Automata

Definition

A timed automaton TA is a tuple (ordered set):

$$\mathrm{TA} = (\mathcal{Z}, \mathcal{C}, \mathcal{U}, \mathcal{Y}, \mathrm{T}, \mathrm{g}, \mathrm{h}, z_0),$$

where z_0 is the initial state,

$\mathcal{Z} = \{z_1, \ldots, z_n\}$	set of states
$\mathcal{C} = \{c_1, \ldots, c_n\}, c_i \in \mathbb{R}_0^+$	set of clocks
$\mathcal{U} = \{\tilde{u}_1, \ldots, \tilde{u}_p\}$	set of input symbols (input alphabet)
$\mathcal{Y} = \{\tilde{y}_1, \ldots, \tilde{y}_q\}$	set of output symbols (output alphabet)
$\mathrm{T} \subseteq \mathcal{Z} \times \mathcal{U} \times \mathcal{Z} \times \mathcal{Y}$	set of transitions
$\mathrm{g} : \mathrm{T} \to P(\mathcal{C})$	guard function
$\mathrm{h} : \mathrm{T} \times \mathcal{C} \to \mathcal{C}$	jump function

There exist many variations of definitions of timed automata in the literature.

Semantics of Timed Automata

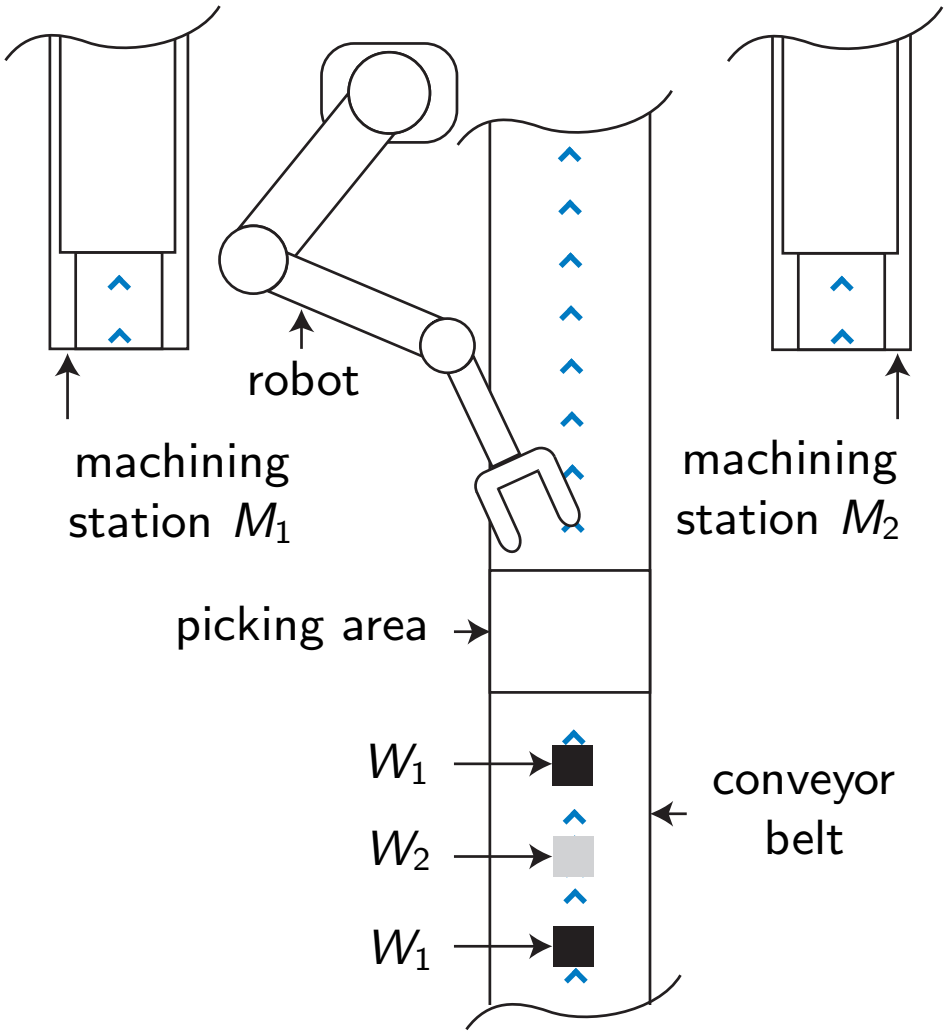
Our definition of a timed automaton has the following semantics:

- Initially ($t = t_0$) all clocks have the value 0 ($\forall i : c_i(t_0) = 0$) if not specified differently.
- Clock values progress: $\forall i : \dot{c}_i = 1$.
- As soon as the vector of clocks c is within a guard set $\mathrm{g}(z, u, z', y)$ of a transition $(z, u) \rightarrow (z', y)$, the corresponding transition is activated.
- As soon as the input event u of an activated transition occurs, the transition is taken and the output event y is generated.
- After a transition is taken, the jump function resets the clock values:

$$c' = \mathrm{h}\big((z, u, z', y), c\big), \quad h_i = \begin{cases} c_i, \text{ if } i \notin \mathrm{r}\big((z, u, z', y)\big) \\ 0, \text{ if } i \in \mathrm{r}\big((z, u, z', y)\big) \end{cases}$$

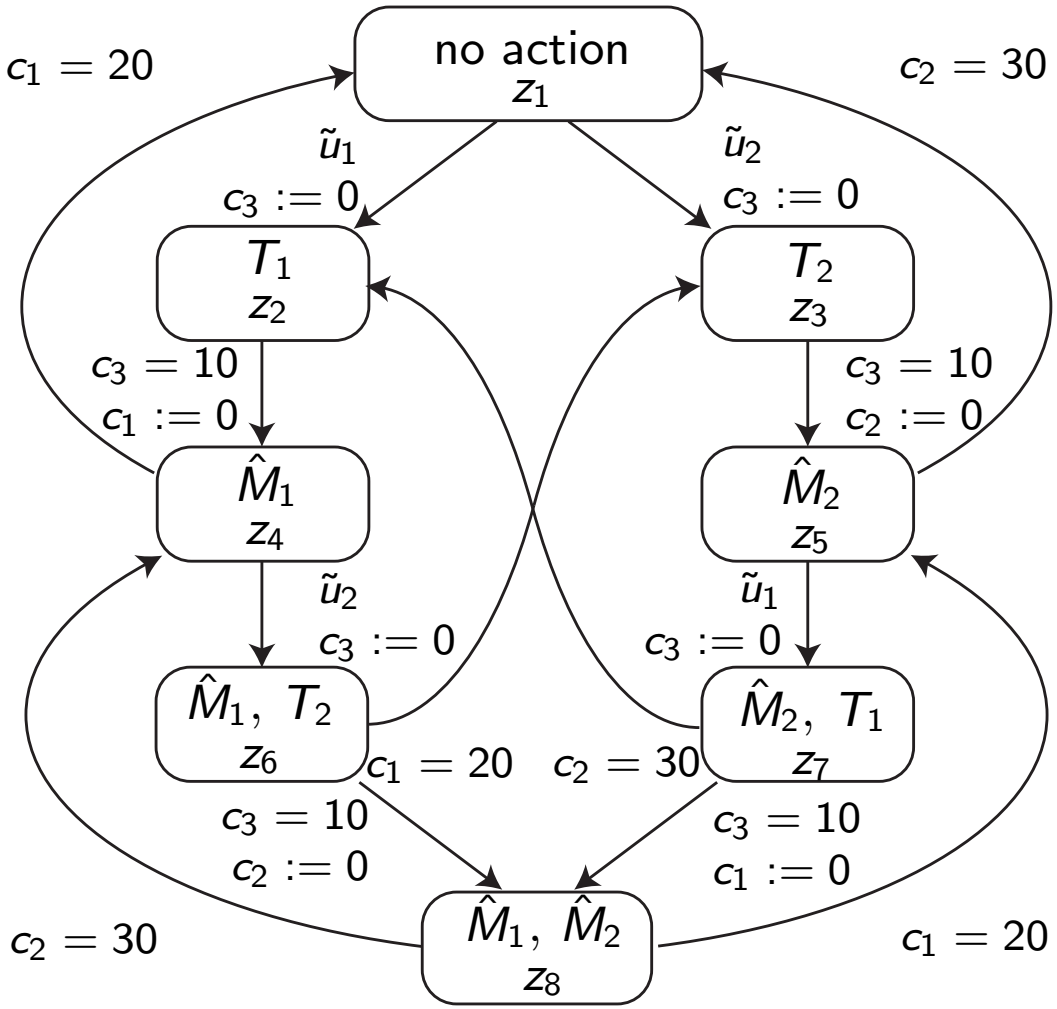
and $\mathrm{r} : \mathrm{T} \rightarrow \mathbb{N}$ is a clock reset function.

Timed Automaton of a Production Plant



- A conveyor belt transports workpieces W_1 and W_2, where W_1 can only be machined by M_1 and W_2 only by M_2.
- The workpieces are transported to the machining stations via a robot from where they are automatically forwarded.
- If the robot is not available, the workpieces continue their journey on the conveyor belt.
- The robot should only transport a workpiece to a machining station if it is idle.
- The robot waits at M_1 or M_2.

Production Plant as Timed Automaton



Parameters/Events:

- Transport time (maching station $\rightarrow$ picking area $\rightarrow$ maching station): 10 [s]
- Machining time in M_1: 20 [s]
- Machining time in M_2: 30 [s]
- Input events:
  - $\tilde{u}_1$: W_1 in picking area
  - $\tilde{u}_2$: W_2 in picking area

Abbreviations for states:

- $\hat{M}_i \hat{=}\ W_i$ in M_i,
- $T_i \hat{=}\ W_i$ is transported.

Timed Automaton of the Production Plant (I)

We first introduce the empty event ϵ. The empty event occurs as soon as a guard region is entered.

- Initial state: $z(0) = z_1$
- Set of states: $\mathcal{Z} = \{z_1, z_2, \ldots, z_8\}$
- Set of clocks: $\mathcal{C} = \{c_1, c_2, c_3\}$
- Input symbols: $\mathcal{U} = \{\tilde{u}_1, \tilde{u}_2, \epsilon\}$
- Output symbols: $\mathcal{Y} = \{\epsilon\}$
- Set of transitions: $\mathrm{T} = \big\{(z_1, \tilde{u}_1, z_2, \epsilon),\ (z_1, \tilde{u}_2, z_3, \epsilon),\ (z_2, \epsilon, z_4, \epsilon),\ (z_3, \epsilon, z_5, \epsilon),\ (z_4, \tilde{u}_2, z_6, \epsilon),\ (z_5, \tilde{u}_1, z_7, \epsilon),\ (z_6, \epsilon, z_8, \epsilon),\ (z_7, \epsilon, z_8, \epsilon),\ (z_8, \epsilon, z_4, \epsilon),\ (z_8, \epsilon, z_5, \epsilon),\ (z_4, \epsilon, z_1, \epsilon),\ (z_5, \epsilon, z_1, \epsilon),\ (z_7, \epsilon, z_2, \epsilon),\ (z_6, \epsilon, z_3, \epsilon)\big\}$

Continued on the next slide...

Timed Automaton of the Production Plant (II)

- guard sets:

$$\begin{aligned}
g\big((z_1, \tilde{u}_1, z_2, \epsilon)\big) &= \{c \in (\mathbb{R}_0^+)^3\} \\
g\big((z_1, \tilde{u}_2, z_3, \epsilon,)\big) &= \{c \in (\mathbb{R}_0^+)^3\} \\
g\big((z_2, \epsilon, z_4, \epsilon)\big) &= \{c \in (\mathbb{R}_0^+)^3 | c_3 = 10\} \\
g\big((z_3, \epsilon, z_5, \epsilon)\big) &= \{c \in (\mathbb{R}_0^+)^3 | c_3 = 10\} \\
&\vdots \quad \vdots
\end{aligned}$$

- jump functions specified by reset functions:

$$\begin{aligned}
r\big((z_1, \tilde{u}_1, z_2, \epsilon)\big) &= \{3\} \\
r\big((z_1, \tilde{u}_2, z_3, \epsilon)\big) &= \{3\} \\
r\big((z_2, \epsilon, z_4, \epsilon)\big) &= \{1\} \\
r\big((z_3, \epsilon, z_5, \epsilon)\big) &= \{2\} \\
&\vdots \quad \vdots
\end{aligned}$$

Run of the Timed Automaton

- The timed input sequence $(\tilde{u}_1, 1), (\tilde{u}_2, 12), (\tilde{u}_1, 33)$ results in the state sequence

$$\underset{c=[0,0,0]}{(z_1,0)} \rightarrow \underset{c=[1,1,0]}{(z_2,1)} \rightarrow \underset{c=[0,11,10]}{(z_4,11)} \rightarrow \underset{c=[1,12,0]}{(z_6,12)}$$
$$\rightarrow \underset{c=[11,0,10]}{(z_8,22)} \rightarrow \underset{c=[20,9,19]}{(z_5,31)} \rightarrow \underset{c=[22,11,0]}{(z_7,33)}$$

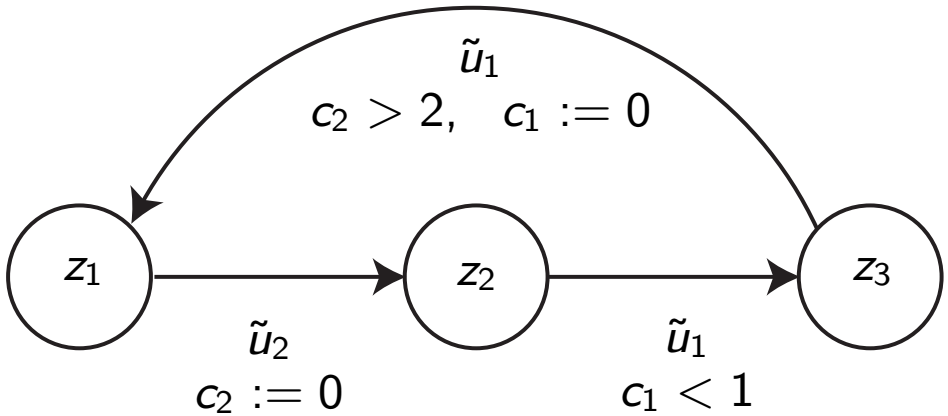
- For the timed input sequence $(\tilde{u}_1, 1), (\tilde{u}_1, 3)$ the timed automaton is not specified. One can assume that undefined input events cause a self transition, resulting in the state sequence

$$\underset{c=[0,0,0]}{(z_1,0)} \rightarrow \underset{c=[1,1,0]}{(z_2,1)} \rightarrow \underset{c=[3,3,2]}{(z_2,3)} \rightarrow \underset{c=[0,11,10]}{(z_4,11)}$$

Reachability Analysis of Timed Automata

An important analysis technique for timed automata is the computation of reachable states, which are a subset of $\mathcal{Z} \times \mathcal{C}$. Reachable sets are obtained by considering all possible clock values and all possible input events at all times.

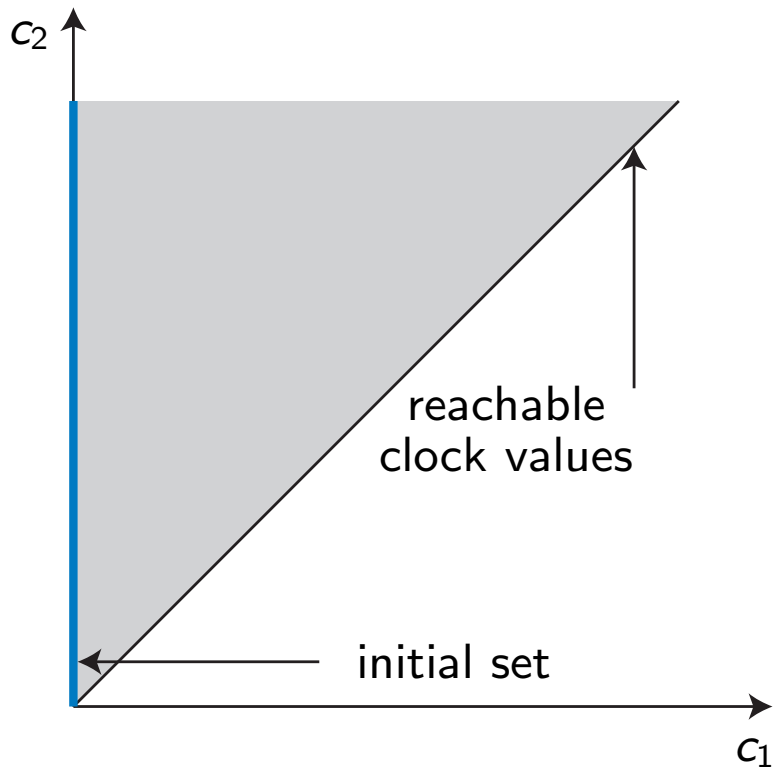
The concept of reachability is best introduced by direct applying it to an example:



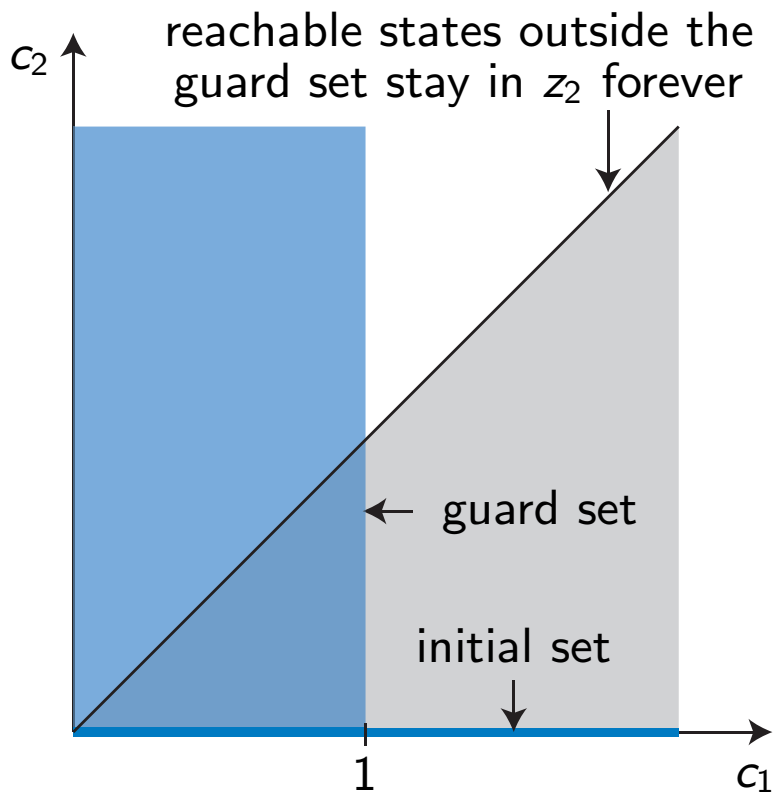
The initial discrete state is $z(0) = z_1$ and the initial clock values are $\{c | c_1 = 0, c_2 \in \mathbb{R}_0^+\}$.

Reachability Analysis of the Example (Step 1 and 2)

Step 1: location z_1



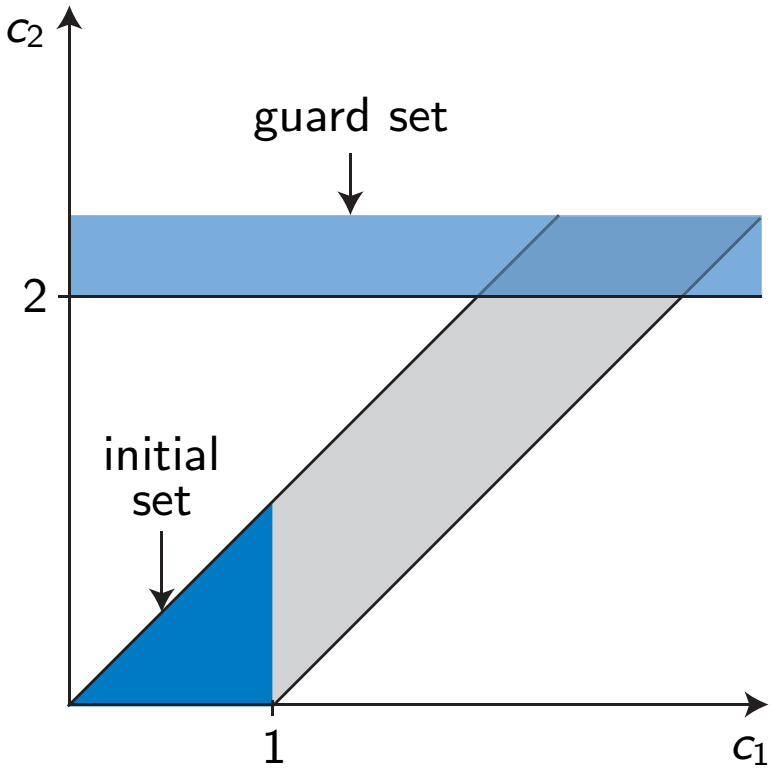
Step 2: location z_2



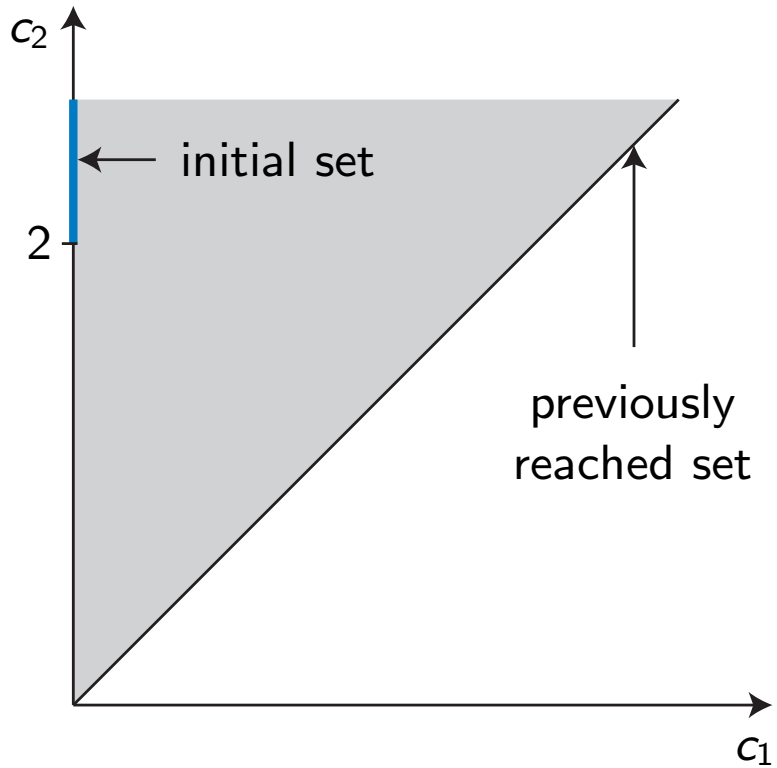
In contrast to the reachable set of finite state automata, the reachable set is no longer countable due to the real-valued clocks.

Reachability Analysis of the Example (Step 3 and 4)

Step 3: location z_3



Step 4: location z_1



In location z_1 no new regions are reached so that the reachability analysis is complete and terminates.

Simulating Timed Automata by Finite State Automata

Under certain restrictions of the guard sets, a timed automaton can be simulated by a finite state automaton.

Those restrictions also make the question decidable whether a certain state can be visited infinitely often, see R. Alur and D. L. Dill: A Theory of Timed Automata, Theoretical Computer Science 126 (1994), pages 183-235.

Guard restrictions for decidable timed automata

Given a value $\xi \in \mathbb{Q}_0^+$ (nonnegative rational number) and a clock value c_i, clock constraints δ are constructed inductively using the Backus-Naur form as

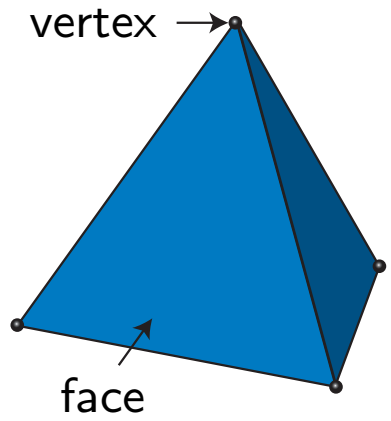
$$\delta ::= c_i \leq \xi | \xi \leq c_i | \neg\delta | \delta_1 \wedge \delta_2$$

Note that a timed automaton is undecidable when $\xi \in \mathbb{R}_0^+$.

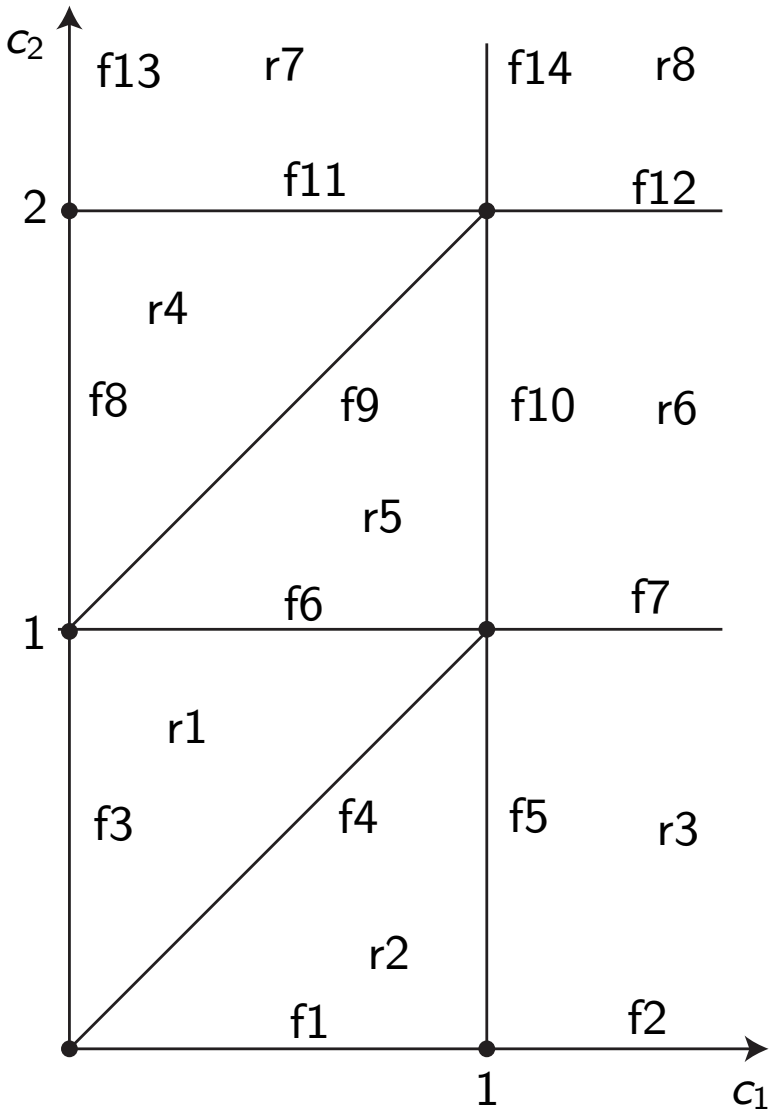
Clock Regions

The restriction $\xi \in \mathbb{Q}_0^+$ makes it possible to use clock regions to represent reachable sets in a standardized way.

- To standardize the clock regions, we multiply all clock bounds ξ_i with the least common multiple (kleinstes gemeinsames Vielfache) of each denominator of all ξ_i, such that all clock constraints become integers. This is only possible since $\xi_i \in \mathbb{Q}_0^+$!
- We introduce the largest clock constraint value $\overline{\xi}_i = \max(\xi_i^{(1)}, \xi_i^{(2)}, \ldots, \xi_i^{(p_i)})$ of each clock c_i.
- Since we only allow one to use the symbols $\leq$, $\neg$ and $\wedge$ for clock constraints, we only require 3 types of clock regions:
  - Vertices,
  - open faces (the vertices are excluded),
  - open regions (the faces and vertices are excluded) represented by simplices (n-dimensional triangles).



Clock Regions of the Previous Example

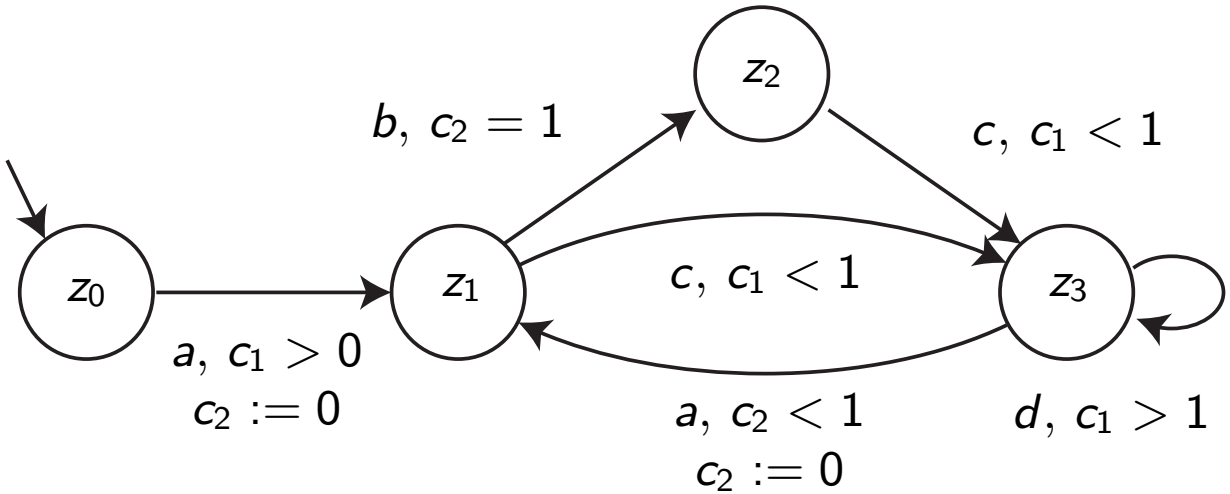


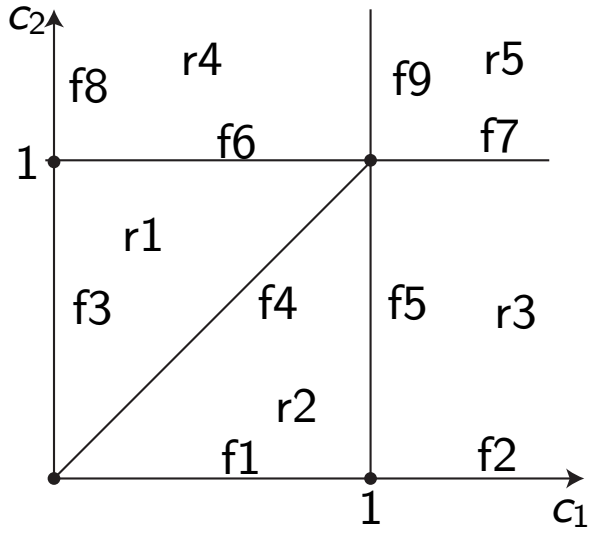
We have $\overline{\xi} = [1, 2]^T$ and all values ξ_i are already integers.

- 6 vertices
- 14 faces (in 2D: line segments): f1, . . . , f14
- 8 regions (in 2D: triangles): r1, . . . , r8

Simulating Finite State Automaton (I)

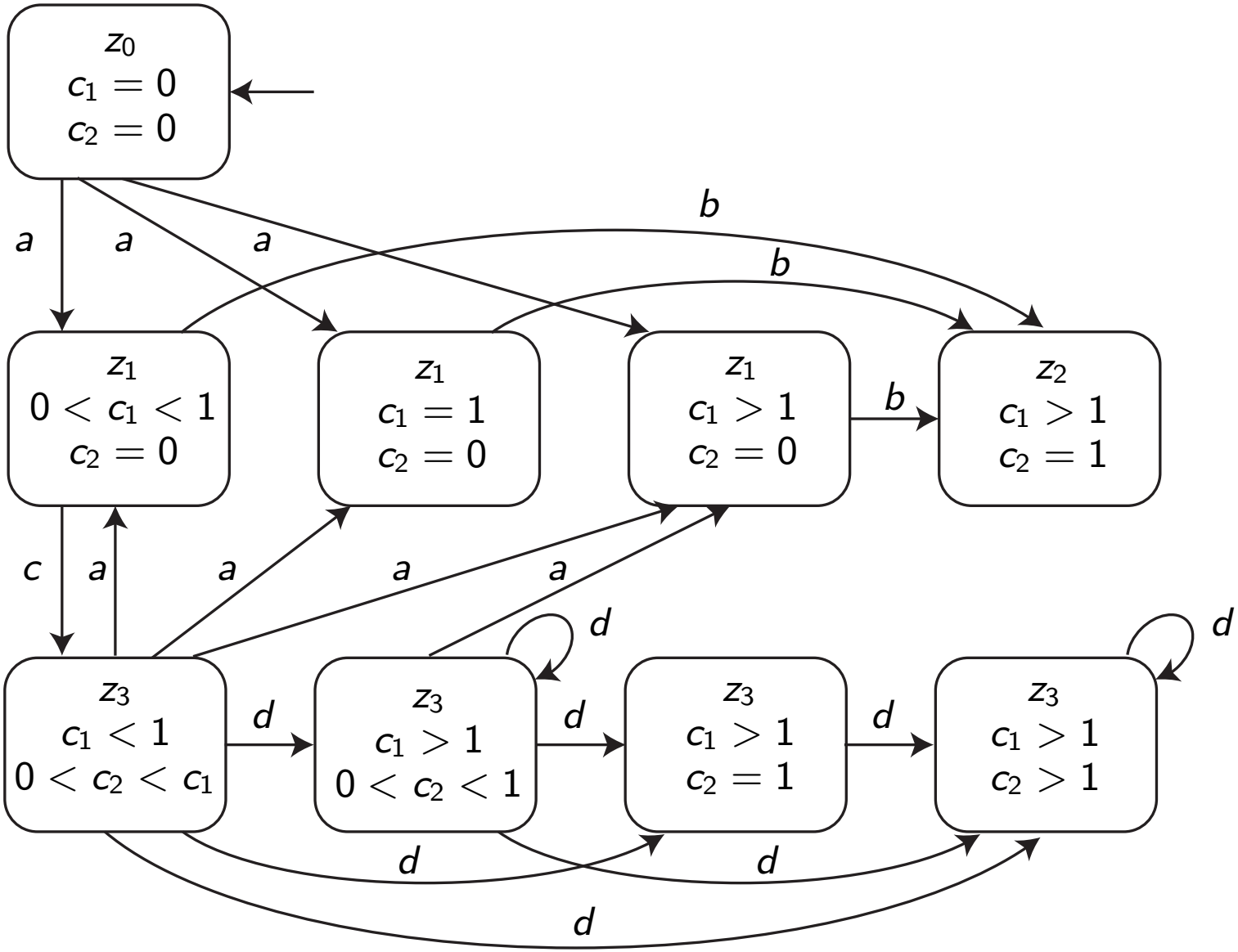
Instead of providing an algorithm for constructing a simulating finite state automaton, we demonstrate the technique by an example from R. Alur and D. L. Dill: A Theory of Timed Automata, Theoretical Computer Science 126 (1994), pages 183-235:





$$\overline{\xi} = [1,\ 1]^T$$

Simulating Finite State Automaton (II)



Simulating Finite State Automaton (III)

- The transition from z_2 to z_3 is impossible.
- The finite state automaton has two absorbing set of states, i.e. states from which one can never escape: $z_2, c \in (\mathbb{R}_0^+)^2$ and $z_2, c_1 > 1, c_2 > 1$.
- The simulating finite state automaton makes it possible to use Model Checking algorithms as introduced earlier.

Further reading

- R. Alur and D. L. Dill: A Theory of Timed Automata, Theoretical Computer Science 126 (1994), pages 183-235.
- C. G. Cassandras and S. Lafortune: Introduction to Discrete Event systems, Springer (2008), chap. 5.

Conclusions

- Timed automata add the crucial aspect of timing constraints to finite state automata.
- Timed automata produce a timed output sequence for a given timed input sequence.
- Reachability analysis makes it possible to analyze all possible behaviors of timed automata.
- When guard regions are restricted to expressions including $\leq$, $\neg$, $\wedge$ and rational numbers, the timed automata can be simulated by a finite state automaton.