Timed Automata

Lecture #15 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: `katoen@cs.rwth-aachen.de`

January 17, 2017

Time-critical systems

- Timing issues are of crucial importance for many systems, e.g.,
  - landing gear controller of an airplane, railway crossing, robot controllers
  - steel production controllers, communication protocols . . . . . .
- In time-critical systems correctness depends on:
  - not only on the logical result of the computation, but
  - also on the time at which the results are produced
- How to model timing issues:
  - discrete-time or continuous-time?

A discrete time domain

- Time has a *discrete* nature, i.e., time is advanced by discrete steps
  - time is modelled by naturals; actions can only happen at natural time values
  - a single transition corresponds to a single time unit
  - $\Rightarrow$ delay between any two events is always a multiple of a single time unit
- Properties can be expressed in traditional temporal logic
  - the next-operator "measures" time passage
  - two time units after being red, the light is green: $\square\,(red \;\Rightarrow\; \bigcirc\bigcirc\, green)$
  - within two time units after red, the light is green:

$$\square\,(red \;\Rightarrow\; \underbrace{(green \;\vee\; \bigcirc\, green \;\vee\; \bigcirc\bigcirc\, green)}_{\bigcirc^{\leqslant 2} green})$$

- Main application area: synchronous systems, e.g., hardware

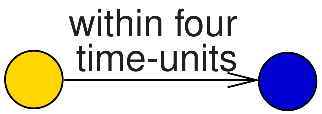
A discrete time domain

- Main advantage: conceptual simplicity
  - labeled transition systems can be taken *as is*
  - temporal logic can be taken *as is*
  - ⇒ traditional model-checking algorithms suffice
  - ⇒ adequate for *synchronous* systems. e.g., hardware systems

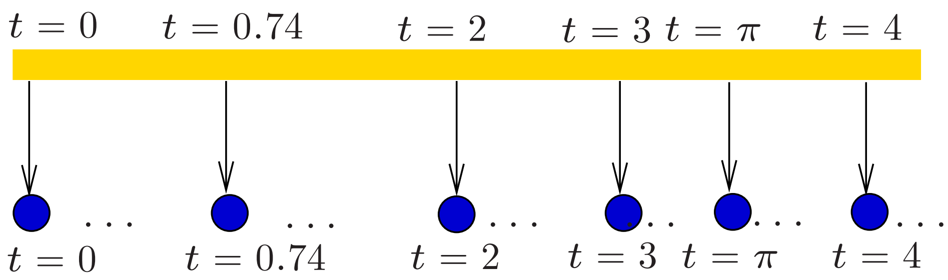
- Main limitations:
  - (minimal) delay between any pair of actions is a multiple of an *a priori* fixed minimal delay
  - ⇒ difficult (or impossible) to determine this in practice
  - ⇒ not invariant against changes of the time scale
  - ⇒ inadequate for *asynchronous* systems. e.g., distributed systems

A continuous time-domain

If time is continuous, state changes can happen at any point in time:



is modeled by



but: infinitely many states and infinite branching

How to check a property like:

once in a yellow state, eventually the system is in a blue state within π time-units?

Approach

- *Restrict expressivity* of the property language
  - e.g., only allow reference to natural time units

$\Longrightarrow$ Timed CTL

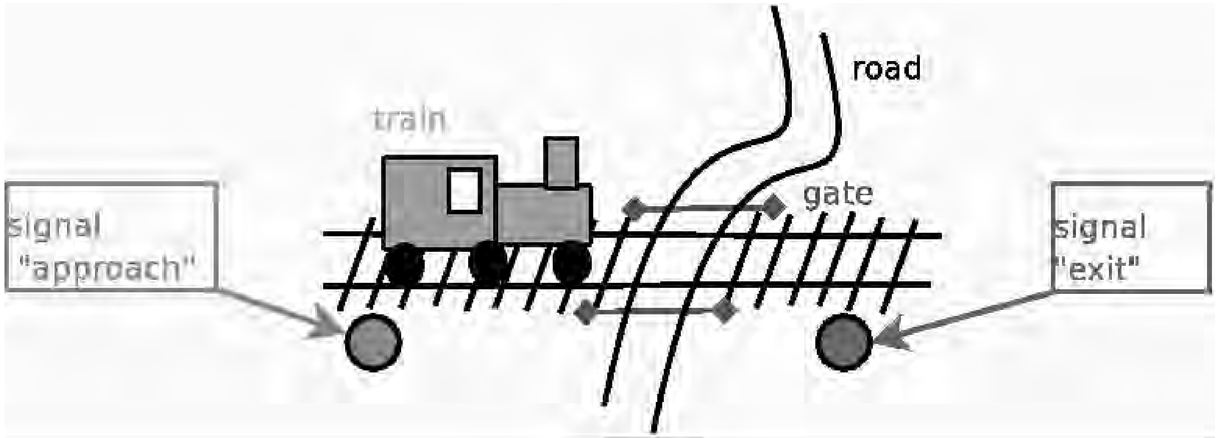
- Model timed systems *symbolically* rather than explicitly
  - in a similar way as program graphs and channel systems

$\Longrightarrow$ Timed Automata

- Consider a *finite quotient* of the infinite state space on-demand
  - i.e., using an equivalence that depends on the property and the timed automaton

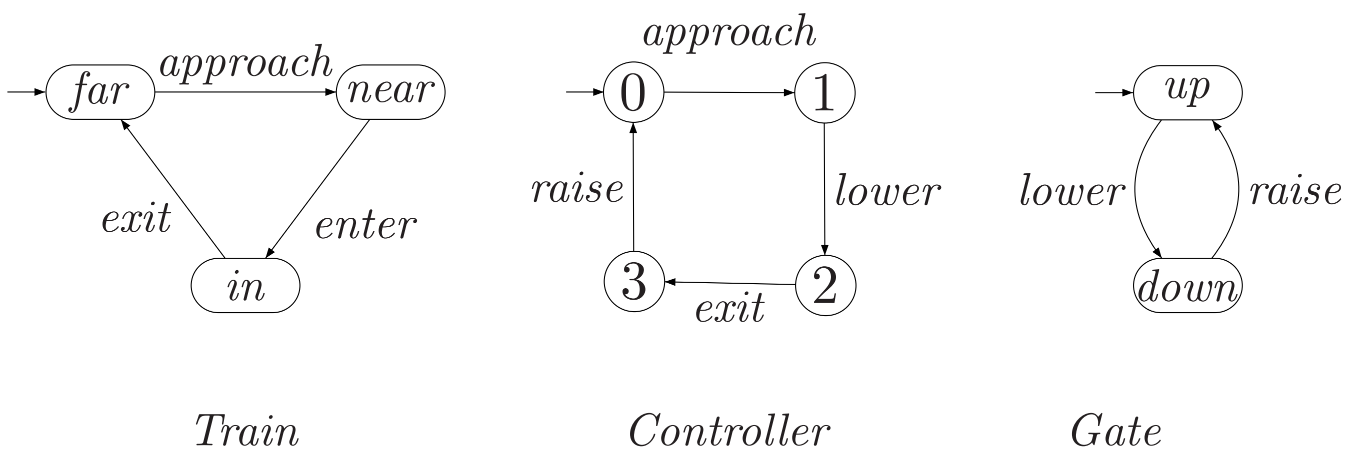
$\Longrightarrow$ Region Automata

A railroad crossing



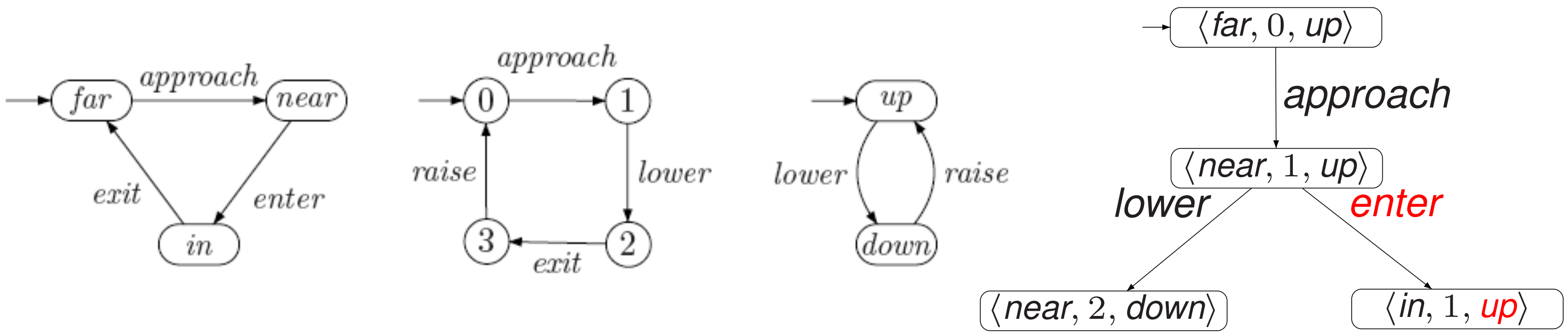
please close and open the gate at the right time!

Modeling using transition systems



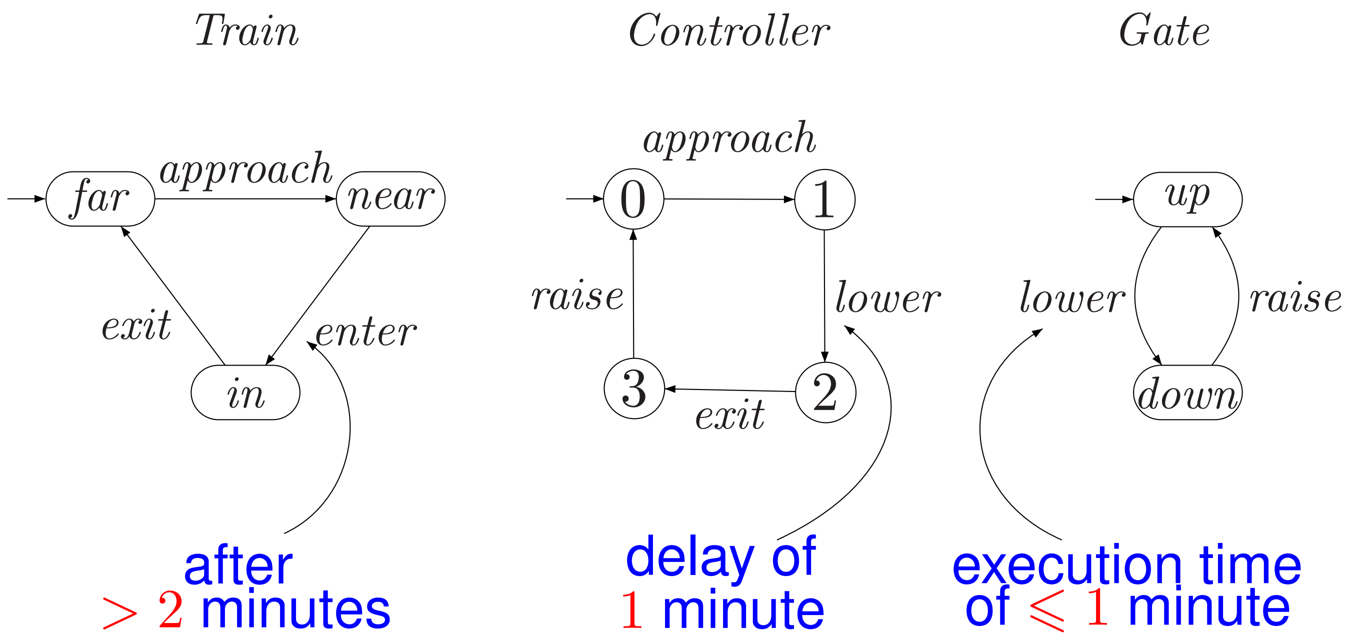
No guarantee that the gate is closed when train is passing

This can be seen as follows

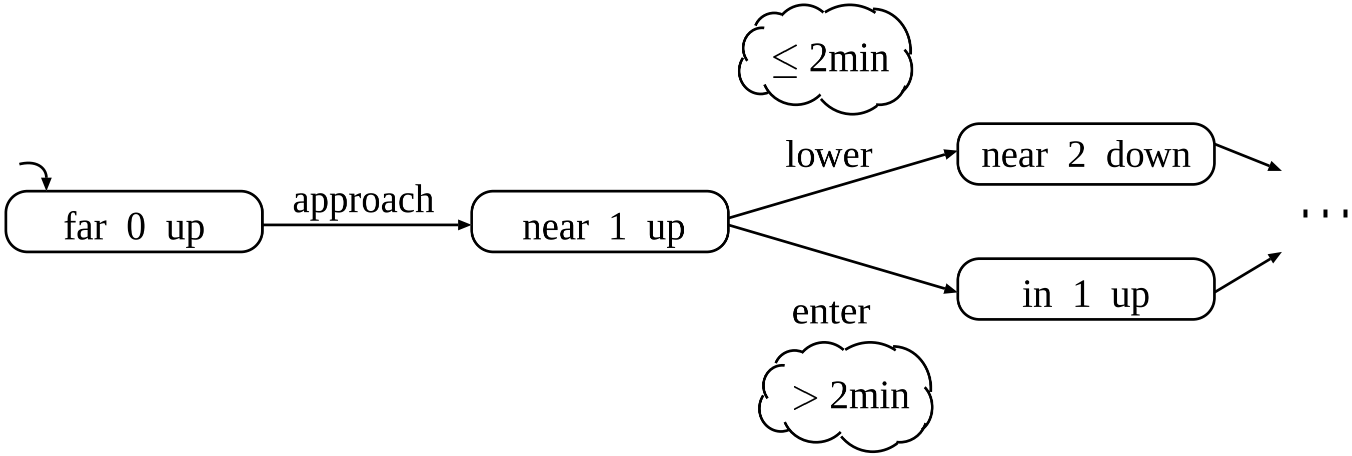


the train can enter the crossing while gate is still open

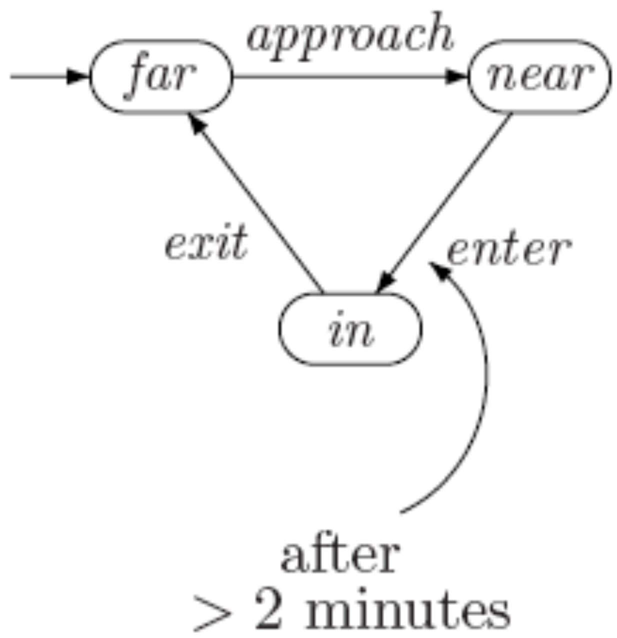
Timing assumptions

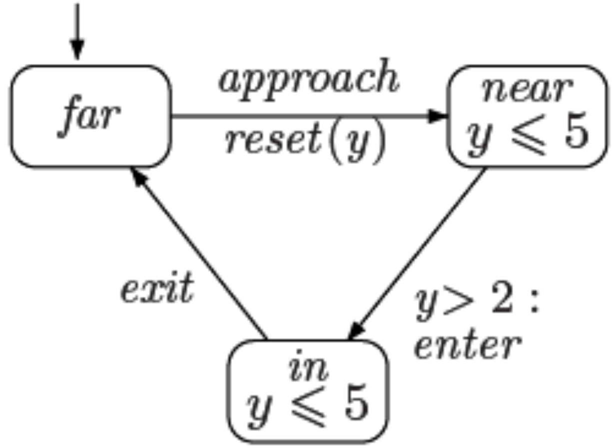


Resulting composite behaviour



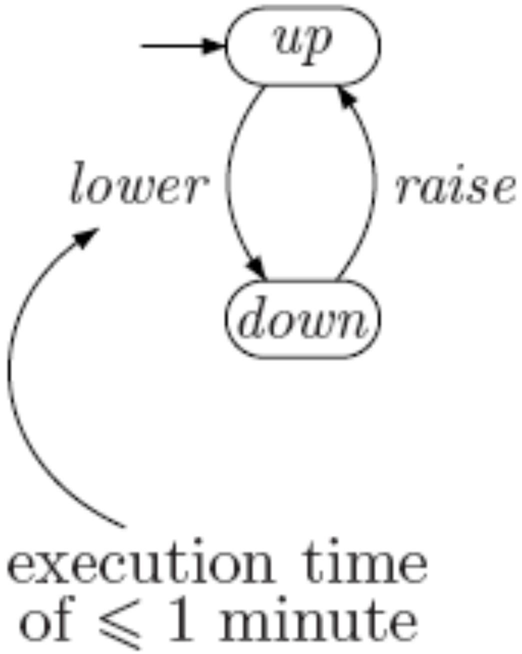
Timed automata model of train

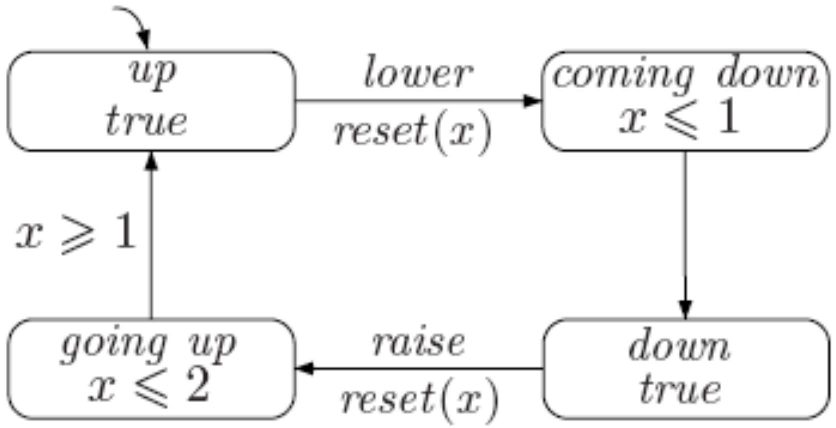




train is now also assumed to leave crossing within five time units

Timed automata model of gate





raising the gate is now also assumed to take between one and two time units

Clocks

- Clocks are variables that take non-negative real values, i.e., in $\mathbb{R}_{\geqslant 0}$
- Clocks increase implicitly, i.e., clock updates are not allowed
- All clocks increase at the same pace, i.e., with rate one
  - after an elapse of d time units, all clocks advance by d
- Clocks may only be inspected and reset to zero
- Boolean conditions on clocks are used as:
  - guards of edges: when is an edge enabled?
  - invariants of locations: how long is it allowed to stay?

Clock constraints

- A *clock constraint* over set C of clocks is formed according to:

$$g ::= \;\; x < c \;\Big|\; x \leqslant c \;\Big|\; x > c \;\Big|\; x \geqslant c \;\Big|\; g \wedge g \quad \text{where } c \in \mathbb{N} \text{ and } x \in C$$

- Let *CC*(C) denote the set of clock constraints over C.
- Clock constraints without any conjunctions are *atomic*
  - let *ACC*(C) denote the set of atomic clock constraints over C

clock difference constraints such as $x - y < c$ can be added at expense of slightly more involved theory

Timed automaton

A *timed automaton* $TA = (Loc, Act, C, \hookrightarrow, Loc_0, Inv, AP, L)$ where:

- Loc is a finite set of locations
- $Loc_0 \subseteq Loc$ is a set of initial locations
- C is a finite set of clocks
- $\hookrightarrow \subseteq Loc \times CC(C) \times Act \times 2^C \times Loc$ is a transition relation
- $Inv : Loc \rightarrow CC(C)$ is an invariant-assignment function, and
- $L : Loc \rightarrow 2^{AP}$ is a labeling function

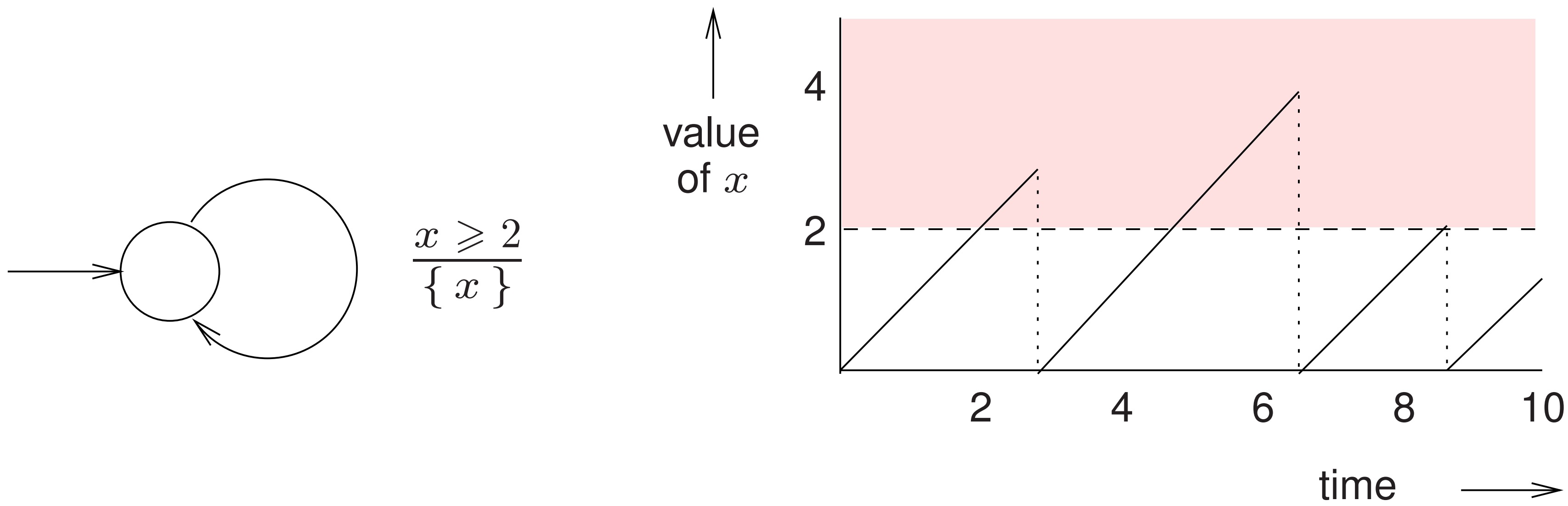
Intuitive interpretation

- Edge $\ell \xrightarrow{g:\alpha,C} \ell'$ means:
  - action α is enabled once guard g holds
  - when moving from location ℓ to ℓ':
    - perform action α, and
    - reset any clock in C will to zero
    - ... all clocks not in C keep their value

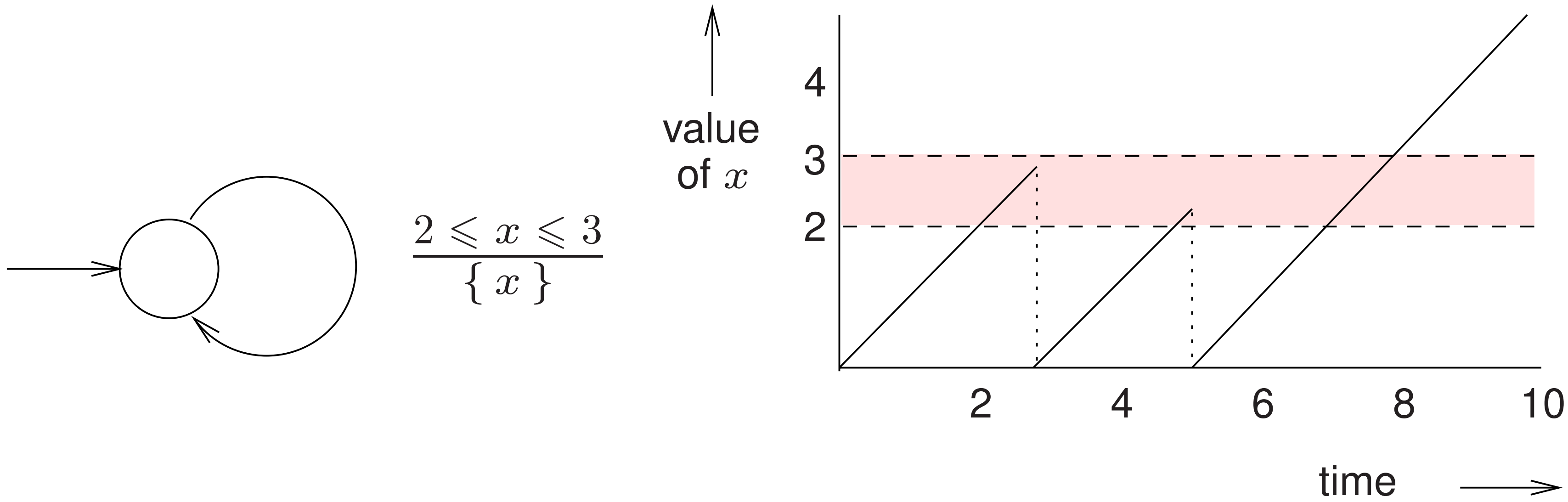
- Nondeterminism if several transitions are enabled

- *Inv*(ℓ) constrains the amount of time that may be spent in location ℓ
  - once the invariant *Inv*(ℓ) becomes invalid, the location ℓ must be left
  - if this is impossible – no enabled transition – no further progress is possible

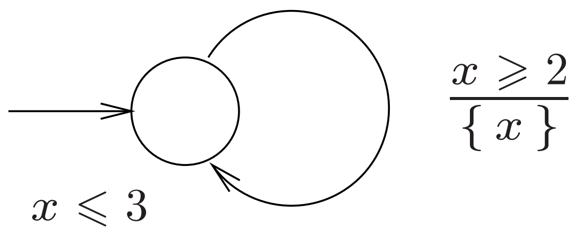
Guards versus invariants

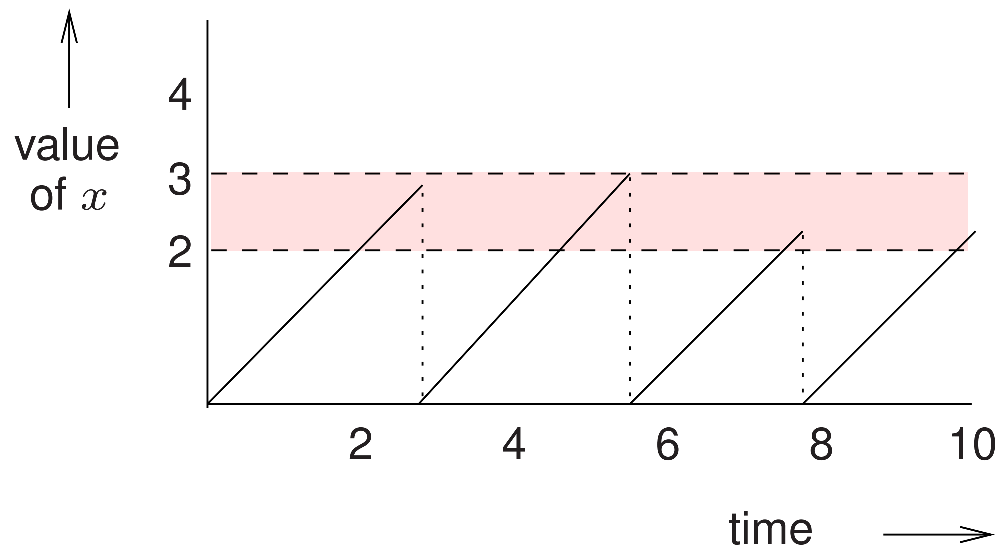


Guards versus invariants

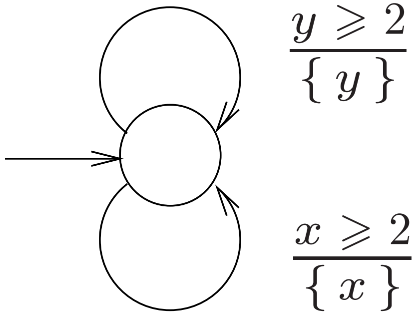


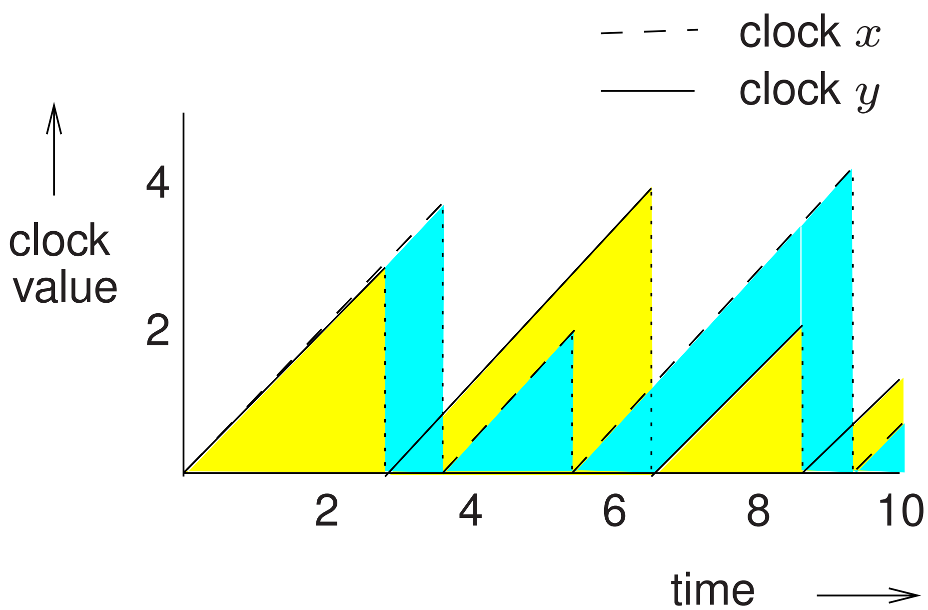
Guards versus invariants





Arbitrary clock differences





This is impossible to model in a discrete-time setting

Fisher's mutual exclusion protocol

Composing timed automata

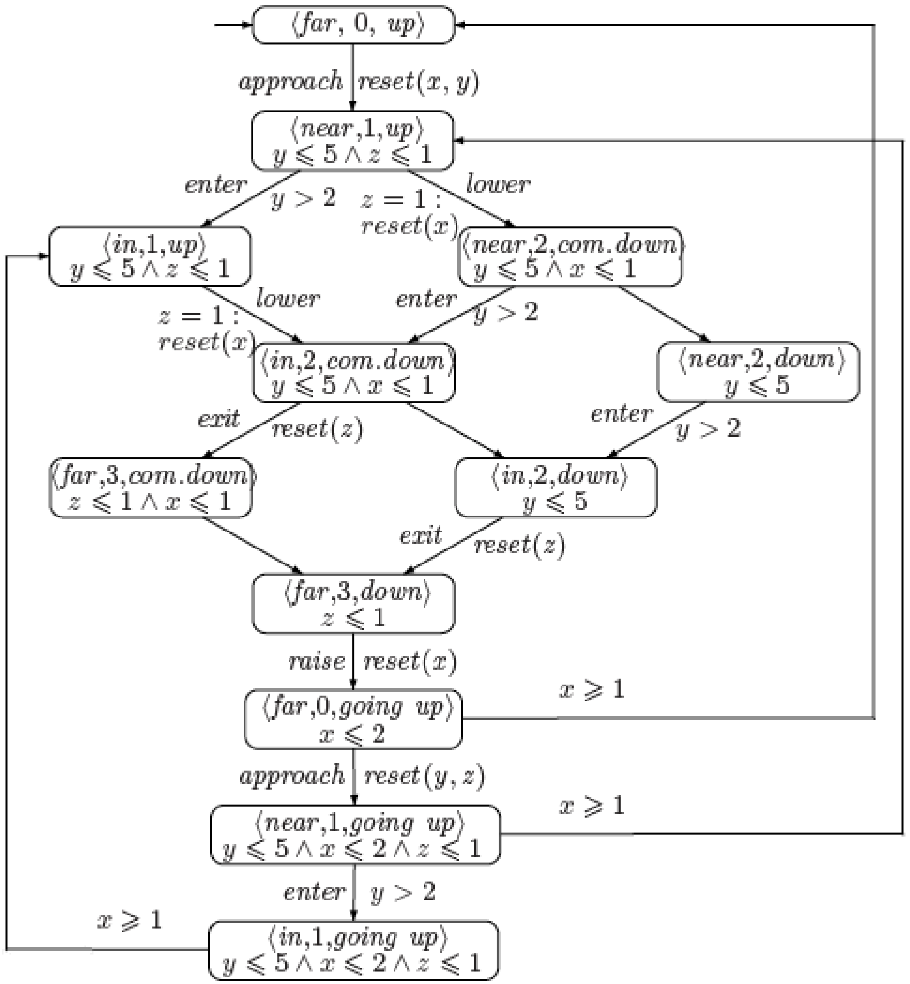
Let $TA_i = (Loc_i, Act_i, C_i, \hookrightarrow_i, Loc_{0,i}, Inv_i, AP, L_i)$ and H an action-set

$$TA_1 \mid\mid_H TA_2 = (Loc, Act_1 \cup Act_2, C, \hookrightarrow, Loc_0, Inv, AP, L) \quad \text{where:}$$

- $Loc = Loc_1 \times Loc_2$ and $Loc_0 = Loc_{0,1} \times Loc_{0,2}$ and $C = C_1 \cup C_2$
- $Inv(\langle \ell_1, \ell_2 \rangle) = Inv_1(\ell_1) \wedge Inv_2(\ell_2)$ and $L(\langle \ell_1, \ell_2 \rangle) = L_1(\ell_1) \cup L_2(\ell_2)$
- $\rightsquigarrow$ is defined by the rules: for $\alpha \in H$ $\dfrac{\ell_1 \overset{g_1:\alpha,D_1}{\hookrightarrow}{}_1 \ell_1' \;\wedge\; \ell_2 \overset{g_2:\alpha,D_2}{\hookrightarrow}{}_2 \ell_2'}{\langle \ell_1, \ell_2 \rangle \overset{g_1 \wedge g_2:\alpha,D_1 \cup D_2}{\hookrightarrow} \langle \ell_1', \ell_2' \rangle}$

for $\alpha \notin H$: $\dfrac{\ell_1 \overset{g:\alpha,D}{\hookrightarrow}{}_1 \ell_1'}{\langle \ell_1, \ell_2 \rangle \overset{g:\alpha,D}{\hookrightarrow} \langle \ell_1', \ell_2 \rangle}$ and $\dfrac{\ell_2 \overset{g:\alpha,D}{\hookrightarrow}{}_2 \ell_2'}{\langle \ell_1, \ell_2 \rangle \overset{g:\alpha,D}{\hookrightarrow} \langle \ell_1, \ell_2' \rangle}$

Example: a railroad crossing


⟨far, 0, up⟩
approach
reset(x, y)
⟨near,1,up⟩
y ≤ 5 ∧ z ≤ 1
enter
y > 2
z = 1 :
reset(x)
lower
⟨in,1,up⟩
y ≤ 5 ∧ z ≤ 1
⟨near,2,com.down⟩
y ≤ 5 ∧ x ≤ 1
lower
enter
y > 2
z = 1 :
reset(x)
⟨in,2,com.down⟩
y ≤ 5 ∧ x ≤ 1
⟨near,2,down⟩
y ≤ 5
exit
reset(z)
enter
y > 2
⟨far,3,com.down⟩
z ≤ 1 ∧ x ≤ 1
⟨in,2,down⟩
y ≤ 5
exit
reset(z)
⟨far,3,down⟩
z ≤ 1
raise
reset(x)
x ≥ 1
⟨far,0,going up⟩
x ≤ 2
approach
reset(y, z)
x ≥ 1
⟨near,1,going up⟩
y ≤ 5 ∧ x ≤ 2 ∧ z ≤ 1
enter
y > 2
x ≥ 1
⟨in,1,going up⟩
y ≤ 5 ∧ x ≤ 2 ∧ z ≤ 1

Clock valuations

- A *clock valuation* η for set C of clocks is a function $\eta : C \longrightarrow \mathbb{R}_{\geqslant 0}$
  - assigning to each clock $x \in C$ its current value $\eta(x)$
- Clock valuation $\eta{+}d$ for $d \in \mathbb{R}_{\geqslant 0}$ is defined by:
  - $(\eta{+}d)(x) = \eta(x) + d$ for all clocks $x \in C$
- Clock valuation reset x in η for clock x is defined by:

$$(\text{reset } x \text{ in } \eta)(y) = \begin{cases} \eta(y) & \text{if } y \neq x \\ 0 & \text{if } y = x. \end{cases}$$

  - reset x in (reset y in η) is abbreviated by reset x, y in η

Satisfaction of clock constraints

Let $x \in C$, $\eta \in \mathit{Eval}(C)$, $c \in \mathbb{N}$, and $g, g' \in \mathit{CC}(C)$

The the relation $\models \subseteq \mathit{Eval}(C) \times \mathit{CC}(C)$ is defined by:

$$
\begin{array}{ll}
\eta \models \text{true} & \\
\eta \models x < c & \text{iff } \eta(x) < c \\
\eta \models x \leqslant c & \text{iff } \eta(x) \leqslant c \\
\eta \models x > c & \text{iff } \eta(x) > c \\
\eta \models x \geqslant c & \text{iff } \eta(x) \geqslant c \\
\eta \models g \wedge g' & \text{iff } \eta \models g \ \wedge \ \eta \models g'
\end{array}
$$

Timed automaton semantics

For timed automaton $\mathit{TA} = \big(\mathit{Loc}, \mathit{Act}, C, \hookrightarrow, \mathit{Loc}_0, \mathit{Inv}, \mathit{AP}, L\big)$:

Transition system $\mathit{TS}(\mathit{TA}) = (S, \mathit{Act}', \rightarrow, I, \mathit{AP}', L')$ where:

- $S = \mathit{Loc} \times \mathit{Eval}(C)$, so states are of the form $s = \langle \ell, \eta \rangle$
- $\mathit{Act}' = \mathit{Act} \cup \mathbb{R}_{\geqslant 0}$, (discrete) actions and time passage actions
- $I = \{ \langle \ell_0, \eta_0 \rangle \mid \ell_0 \in \mathit{Loc}_0 \;\wedge\; \eta_0(x) = 0 \text{ for all } x \in C \}$
- $\mathit{AP}' = \mathit{AP} \cup \mathit{ACC}(C)$
- $L'(\langle \ell, \eta \rangle) = L(\ell) \cup \{ g \in \mathit{ACC}(C) \mid \eta \models g \}$
- $\hookrightarrow$ is the transition relation defined on the next slide

Timed automaton semantics

The transition relation $\rightarrow$ is defined by the following two rules:

- Discrete transition: $\langle \ell, \eta \rangle \xrightarrow{\alpha} \langle \ell', \eta' \rangle$ if all following conditions hold:
  - there is a transition labeled $(g : \alpha, D)$ from location ℓ to ℓ' such that:
  - g is satisfied by η, i.e., $\eta \models g$
  - $\eta' = \eta$ with all clocks in D reset to 0, i.e., $\eta' =$ reset D in η
  - η' fulfills the invariant of location ℓ', i.e., $\eta' \models$ *Inv*(ℓ')
- Delay transition: $\langle \ell, \eta \rangle \xrightarrow{d} \langle \ell, \eta{+}d \rangle$ for $d \in \mathbb{R}_{\geqslant 0}$ if $\eta{+}d \models$ *Inv*(ℓ)

Example