Timed Automata

CS60030 FORMAL SYSTEMS

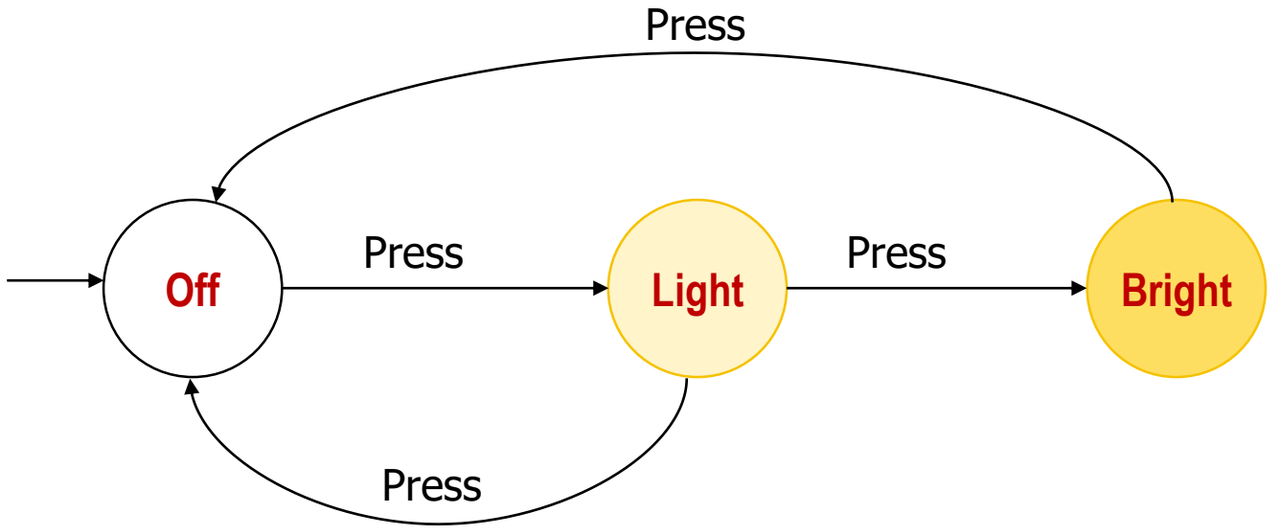
PALLAB DASGUPTA,
FNAE, FASc,
A K Singh Distinguished Professor in AI,
Dept of Computer Science & Engineering
Indian Institute of Technology Kharagpur
Email: pallab@cse.iitkgp.ac.in
Web: http://cse.iitkgp.ac.in/~pallab

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

FMSAFE

FORMAL METHODS FOR SAFETY CRITICAL SYSTEMS

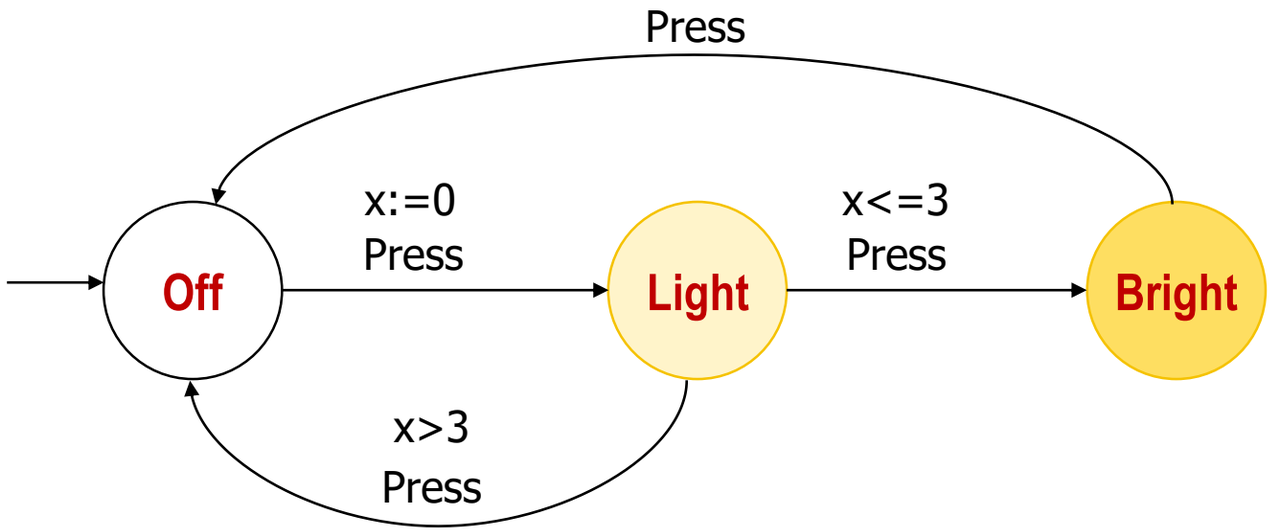
Simple Light Control



WANT: from the Off state, if pressed twice quickly
then the light will get brighter;
otherwise the light will be turned off.

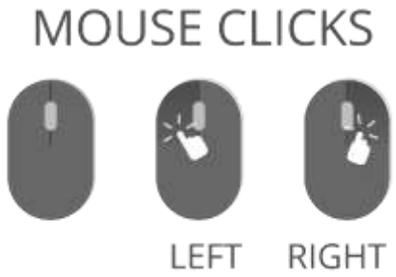
Some of these slides are adapted from Prof. Rajeev Alur's presentations

Simple Light Control



Solution: Add a real-valued clock x

Adding continuous variables to state machines



SIMILARLY: if left button of mouse is pressed twice quickly
then treat it as a double click;
otherwise treat it as a single click.

Systems and Automata

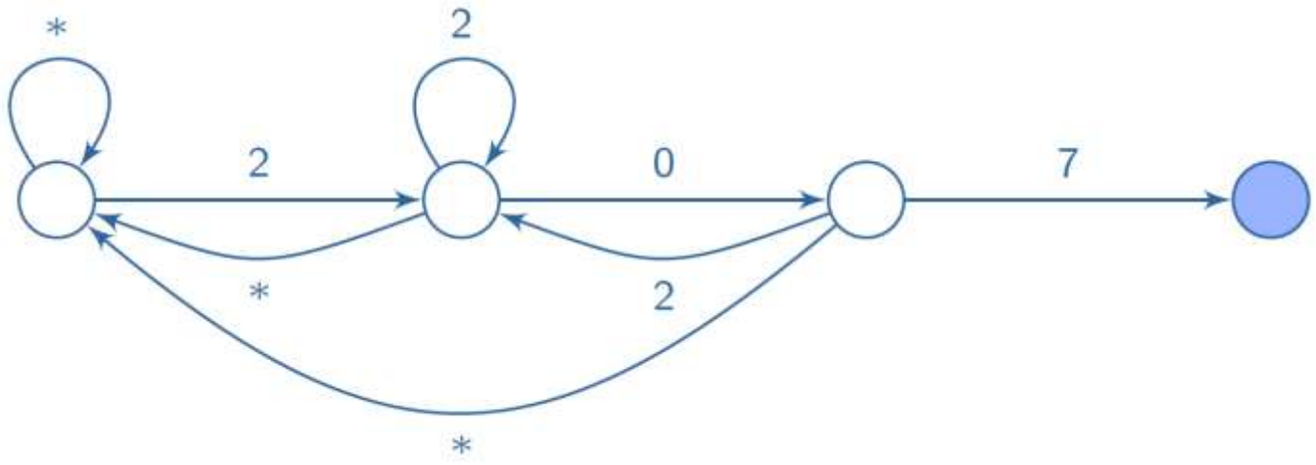
Systems under analysis are modeled and represented as transition systems:

- **Finite automata**
- **Pushdown automata**
- **Program graphs**
- **Timed automata**
- **Hybrid automata**
- **Petri Nets**
- **Channel Systems**
- **Message Sequence Charts**
- **…**

Examples of Models

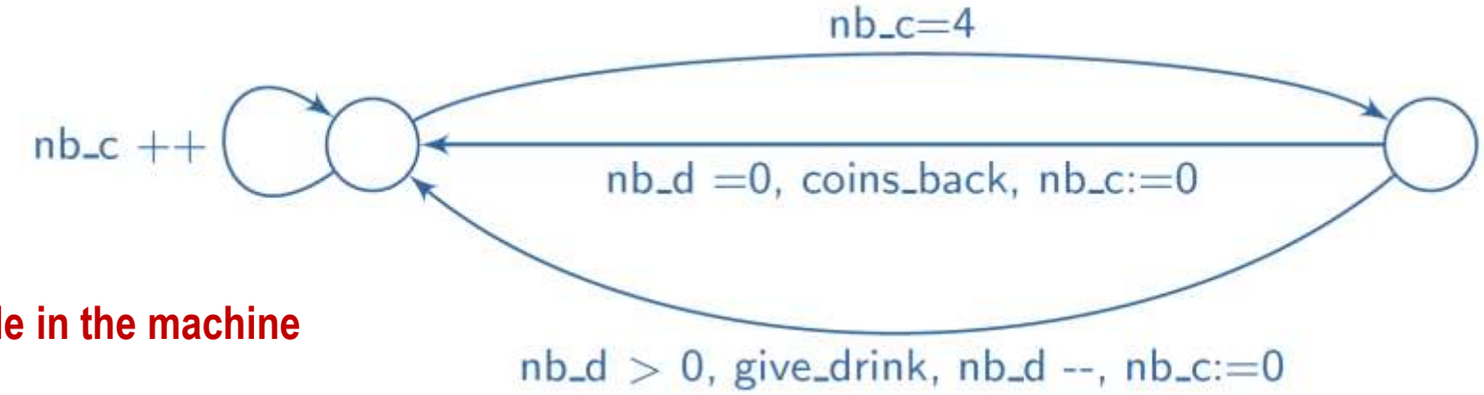
- A numerical code door lock:

  Code: 207

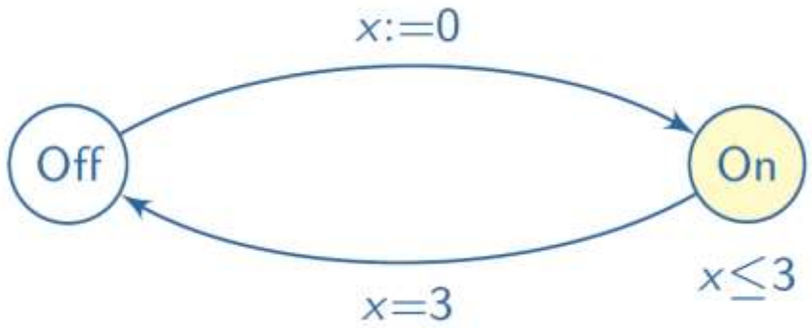


- A vending machine:

  nb_c: number of coins entered
  nb_d: number of drinks available in the machine
  Each drink costs 4 coins

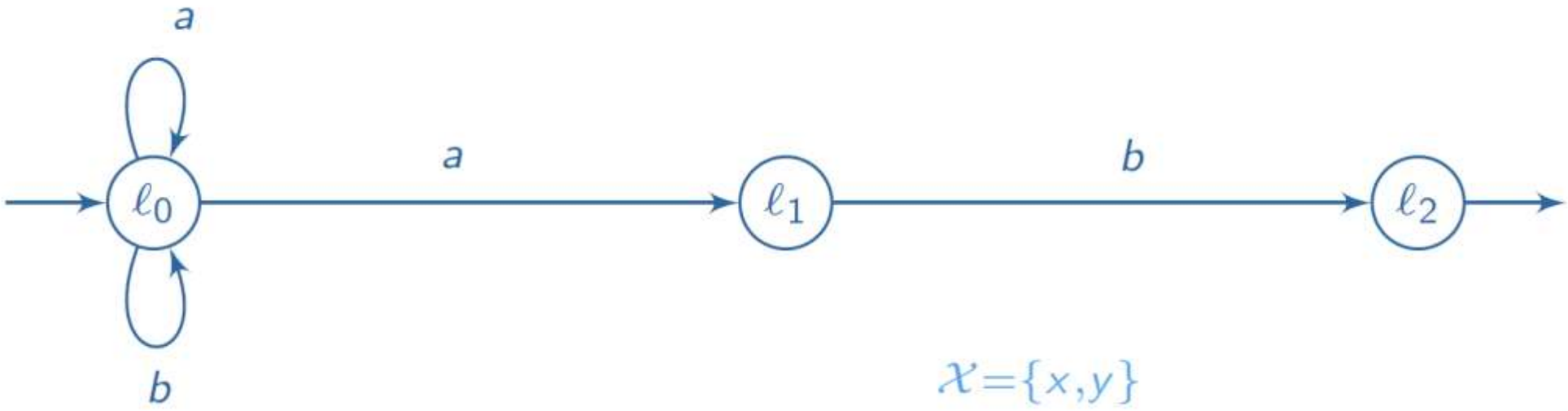


- A timed - switch:



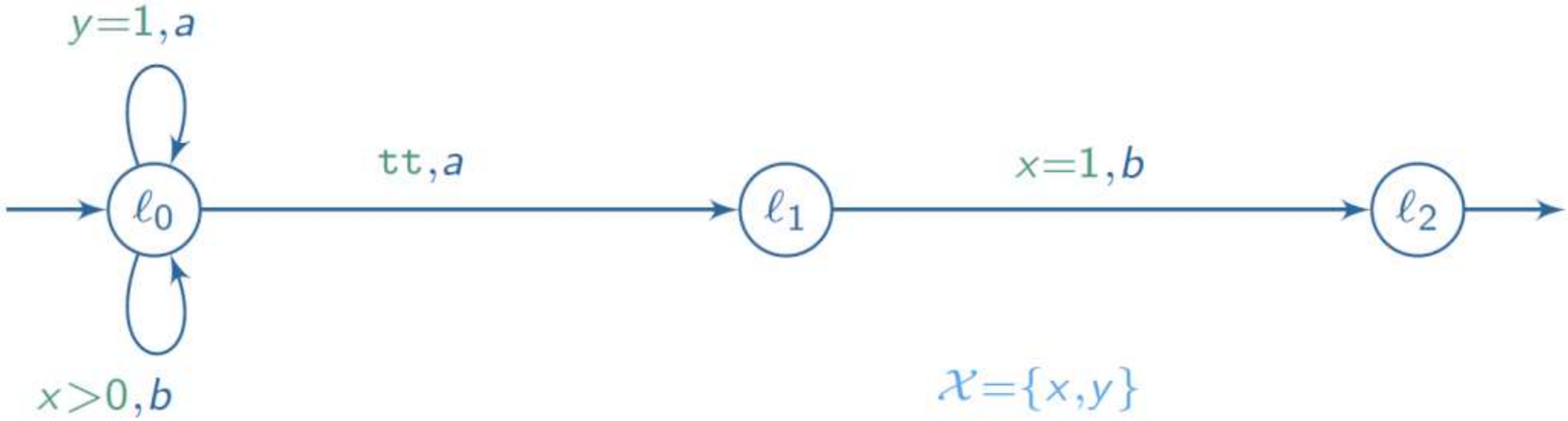
Timed Automata - informally

Timed automaton: **Finite automaton** enriched with **clocks**



Timed Automata - informally

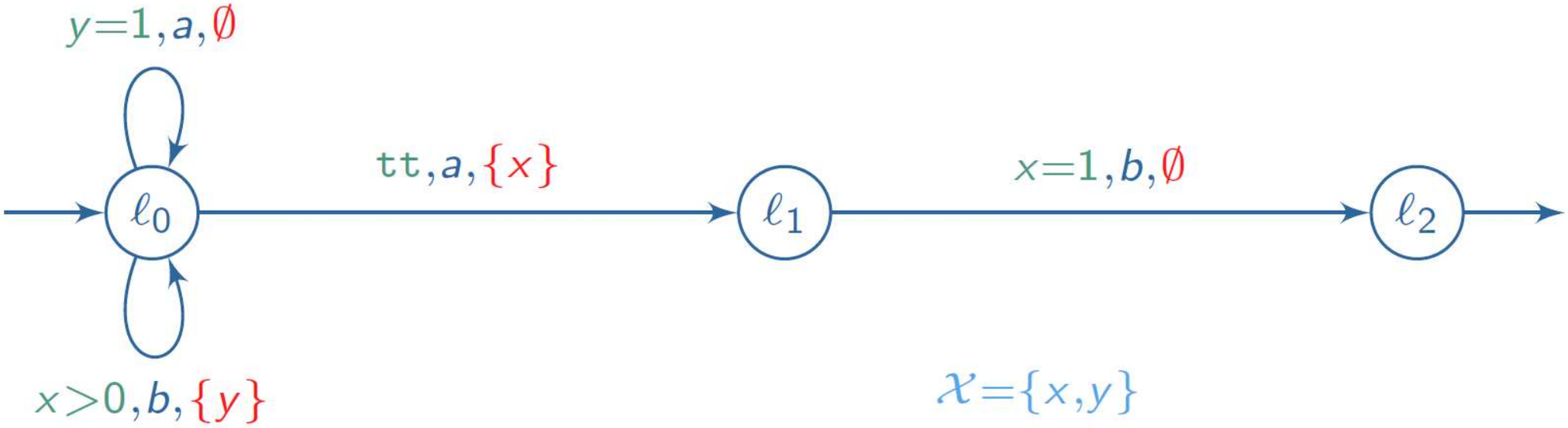
Timed automaton: **Finite automaton** enriched with **clocks**



Transitions: equipped with **guards**

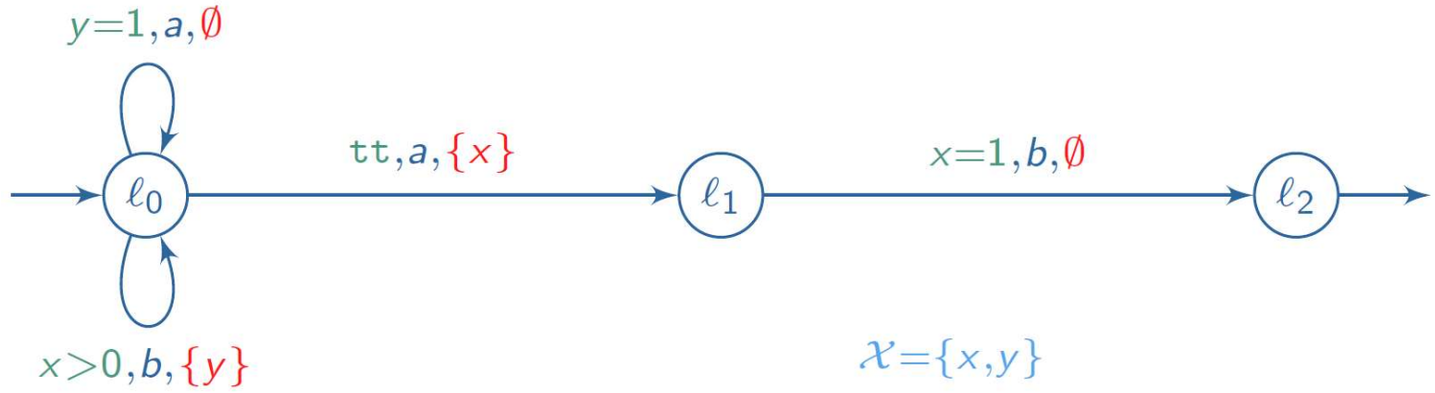
Timed Automata - informally

Timed automaton: **Finite automaton** enriched with **clocks**



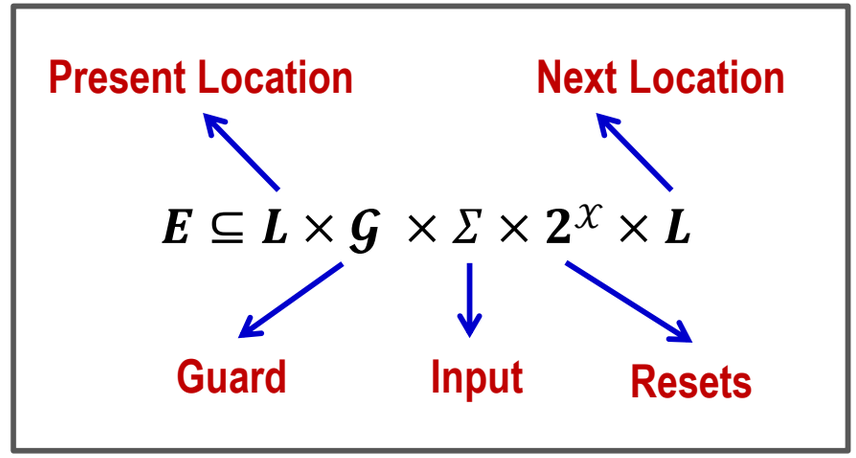
Transitions: equipped with **guards** and sets of **reset** clocks

Timed Automaton - Model Structure



A Timed Automaton is a tuple $A = (L, L_0, Lacc, \Sigma, \mathcal{X}, E)$ where,

- L is a *finite* set of locations,
- $L_0 \subseteq L$, is the initial set of locations
- $L_{acc} \subseteq L$, is the set of accepting locations
- Σ is the *finite* alphabet
- $\mathcal{X}$ is the *finite* set of clocks
- $E \subseteq L \times \mathcal{G} \times \Sigma \times 2^{\mathcal{X}} \times L$, is the set of edges
  - $\mathcal{G} = \{\wedge\, x \bowtie c \mid x \in \mathcal{X}, c \in \mathbb{N}\}$ is the set of guards, $\bowtie \in \{<, \leq, =, >, \geq\}$



Timed Automaton - Semantics

Valuation: $v \in \mathbb{R}_{+}^{X}$, **assigns to each clock a *clock-value*.**

State: $(l, v) \in L \times \mathbb{R}_{+}^{X}$, **is composed of a valuation and a location.**

Transitions between states of A:

- **Delay transitions:** $(l, v) \xrightarrow{\tau} (l , v + \tau)$
- **Discrete transitions:** $(l, v) \xrightarrow{a} (l' , v')$
  - **If** $\exists (l, g, a, Y, l') \in E$ **with** $v \vDash g$ **and** $\begin{cases} v'(x) = 0 & if \ \ x \in Y \\ v'(x) = v(x) & otherwise \end{cases}$

Runs, Sequences, Words, Languages

Run of $\mathcal{A}$: $(l_0, v_0) \xrightarrow{\tau_1} (l_0, v_0+\tau_1) \xrightarrow{a_1} (l_1, v_1) \xrightarrow{\tau_2} (l_1, v_1+\tau_2) \xrightarrow{a_2} \dots \xrightarrow{a_k} (l_k, v_k)$

or simply: $(l_0, v_0) \xrightarrow{\tau_1 a_1} (l_1, v_1) \xrightarrow{\tau_2, a_2} \dots \xrightarrow{\tau_k, a_k} (l_k, v_k)$

Time Sequence: $\boldsymbol{t} = (t_i)_{1 \le i \le k}$ **is a finite non-decreasing sequence over** $\mathbb{R}_+$.

Timed Word: **w =** $(\sigma, \boldsymbol{t}) = (a_i, t_i)_{1 \le i \le k}$ **where** $a_i \in \Sigma$ **and** $\boldsymbol{t}$ **is a time sequence.**

Accepted Timed Word: **A timed word w =** $(a_1, t_1)(a_2, t_2) \dots (a_k, t_k)$ **is accepted in** $\mathcal{A}$**, if there is a run**

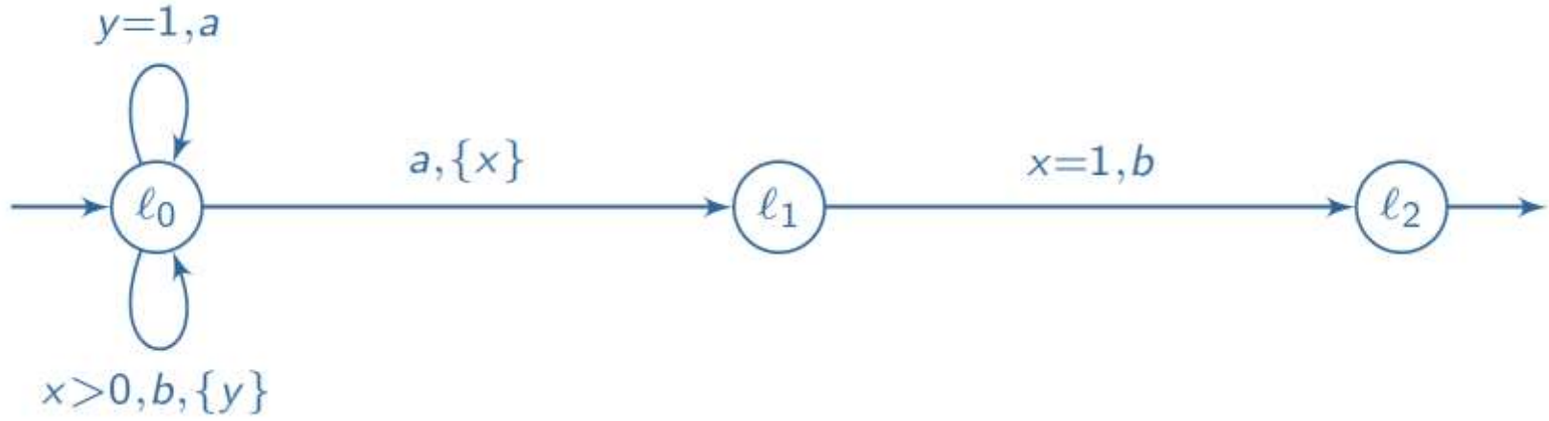
$\rho = (l_0, v_0) \xrightarrow{\tau_1 a_1} (l_1, v_1) \xrightarrow{\tau_2, a_2} \dots \xrightarrow{\tau_{k+1}, a_{k+1}} (l_{k+1}, v_{k+1})$ **with** $l_0 \in \mathrm{L}_0$, $l_k \in \mathrm{L}_{\mathrm{acc}}$ **and** $t_i = \sum_{j<i} t_j$

Accepted timed language: $\mathcal{L}(\mathrm{A})$ = $\{\mathrm{w} \mid \mathrm{w} \text{ accepted by A}\}$

An Example

We omit:

- **Guards when they are identity**
- **Reset set when empty**

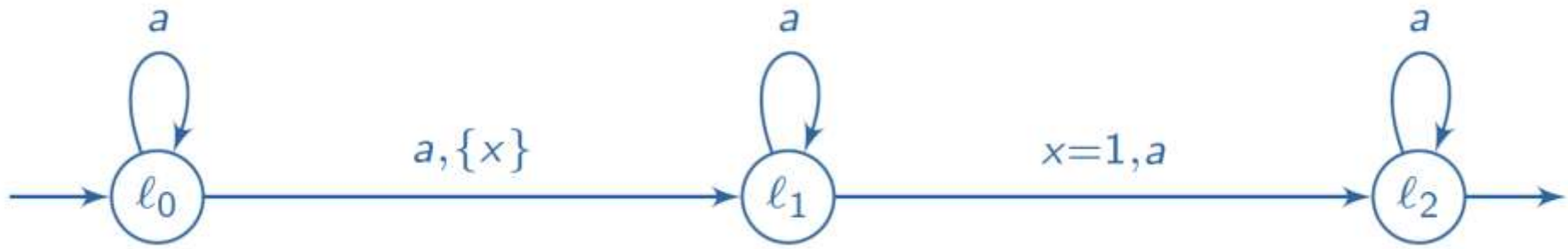


w = (b, 0.1)(b, 0.3)(a, 1.3)(b, 1.5)(a, 1.5)(b, 2.5) is an accepted timed word

An accepting run for w is

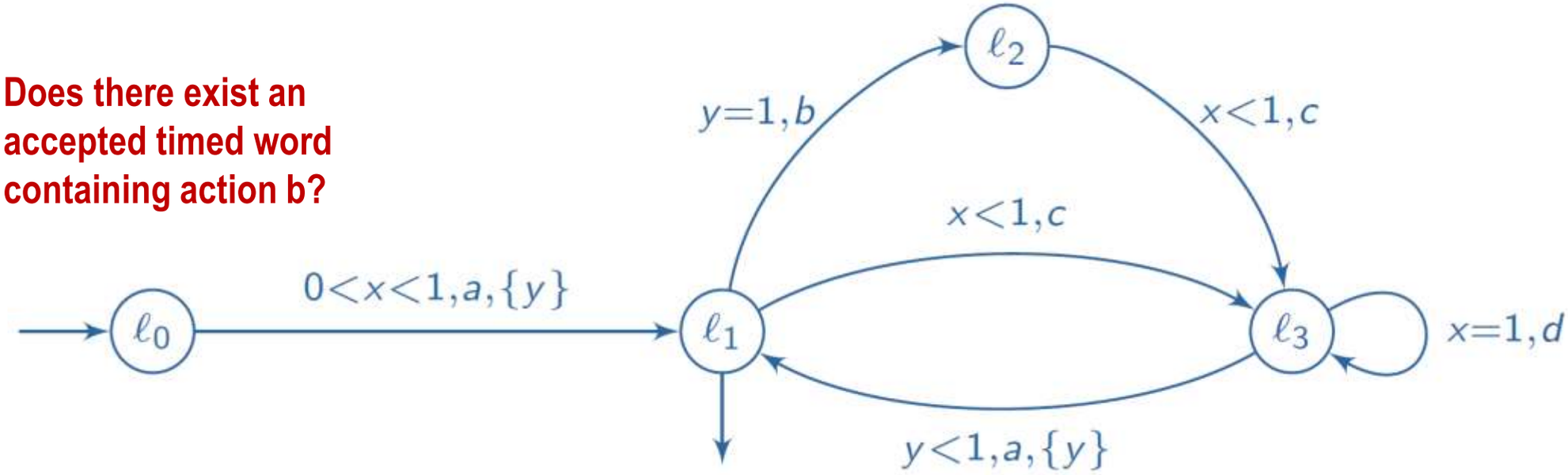
$$(l_0,0,0) \xrightarrow{0.1,b} (l_0,0.1,0) \xrightarrow{0.2,b} (l_0,0.3,0) \xrightarrow{1,a} (l_0,1.3,1) \xrightarrow{0.2,b} (l_0,1.5,0) \xrightarrow{0,a} (l_1,0,0) \xrightarrow{1,b} (l_2,1,1)$$

More Examples

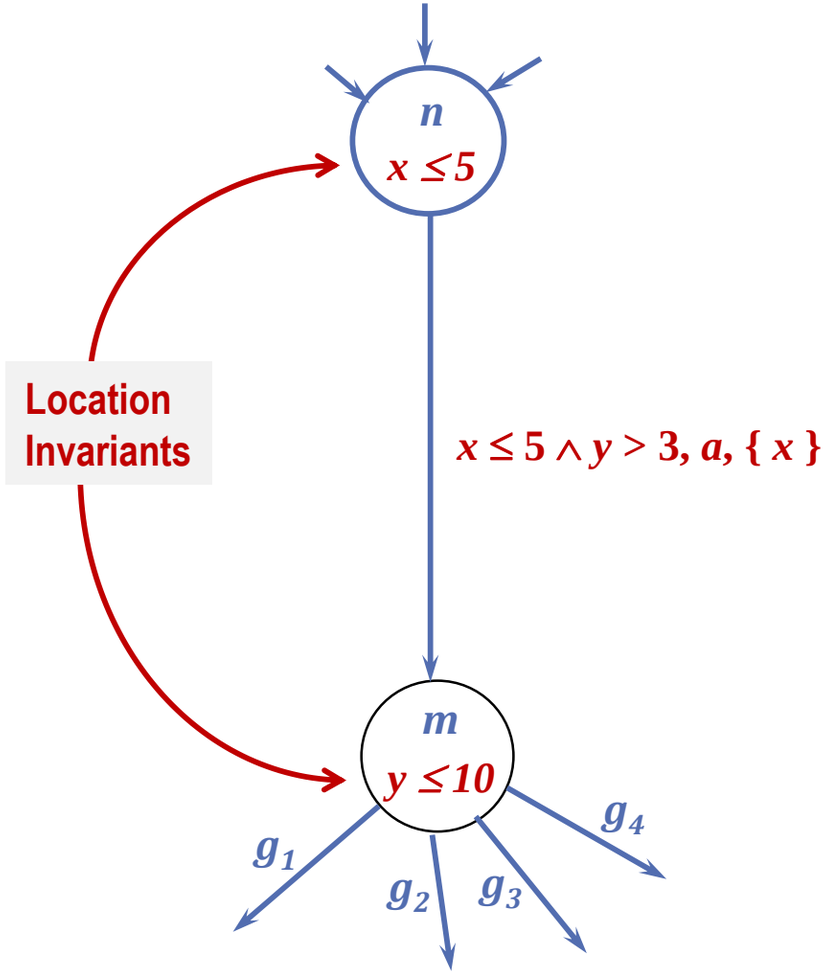


$\mathcal{L}(\mathrm{A}) = \{(\mathrm{a}, \mathrm{t}_1), (\mathrm{a}, \mathrm{t}_2), \ldots, (\mathrm{a}, t_k) | \; \exists i < j, t_j - t_i = 1\}$

Does there exist an accepted timed word containing action b?



Adding Invariants

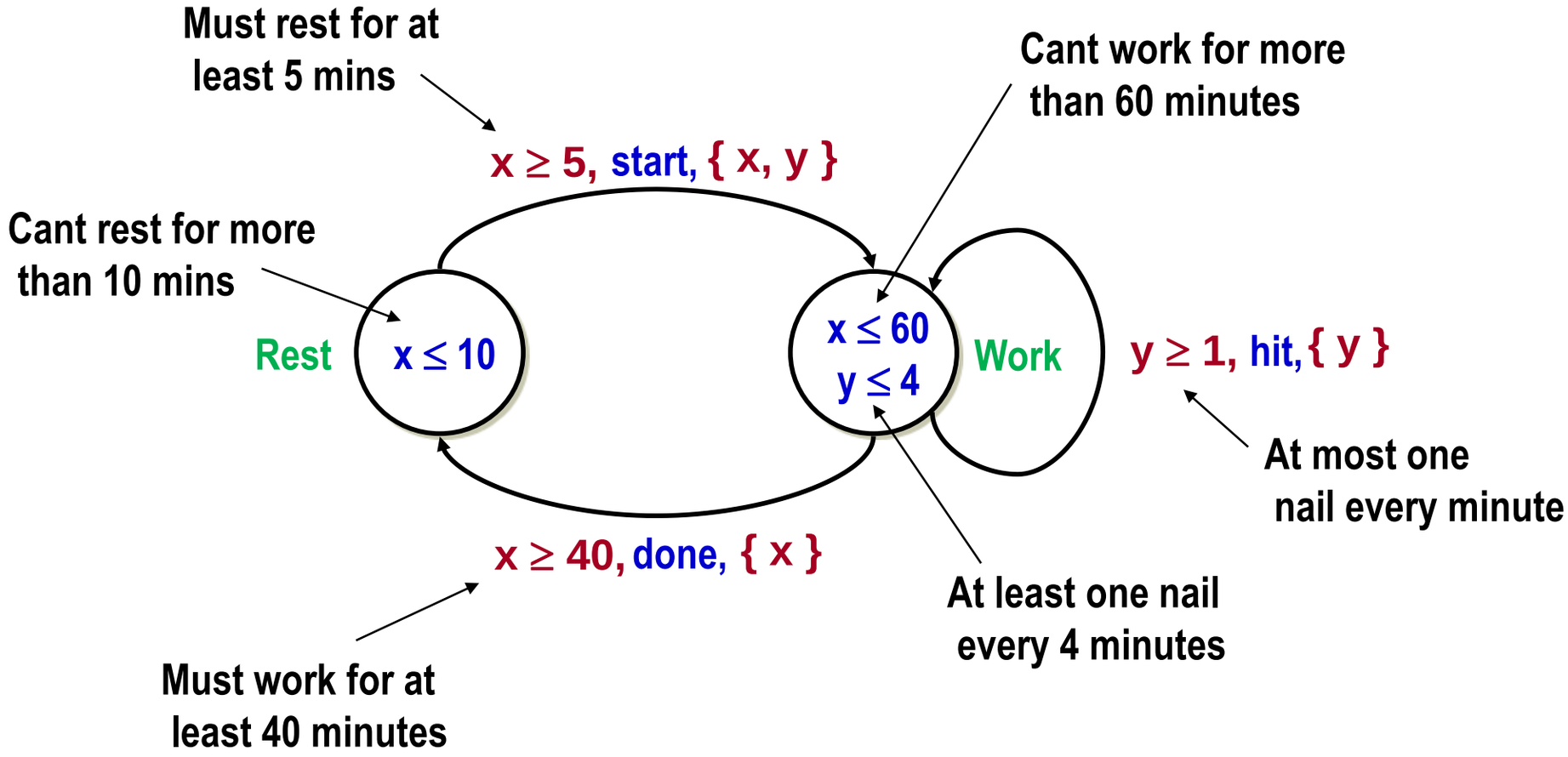


$(\, n\, ,\, x{=}2.4\, ,\, y{=}3.1415\,) \xrightarrow{\textit{wait(3.2)}} \times$

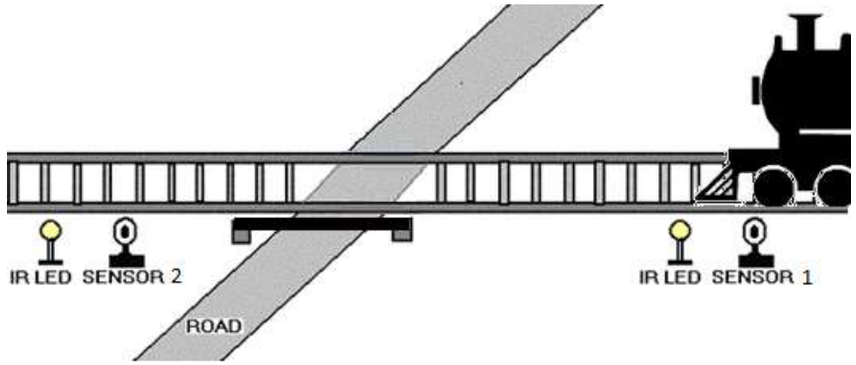
$(\, n\, ,\, x{=}2.4\, ,\, y{=}3.1415\,) \xrightarrow{\textit{wait(1.1)}} (\, n\, ,\, x{=}3.5\, ,\, y{=}4.2415\,)$

Invariants ensure progress!!

Another Example: Model of a Small Jobshop !!

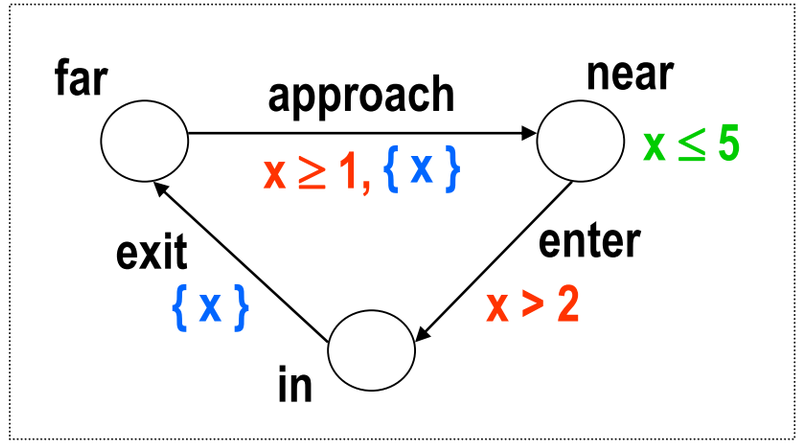


Rail Gate Crossing

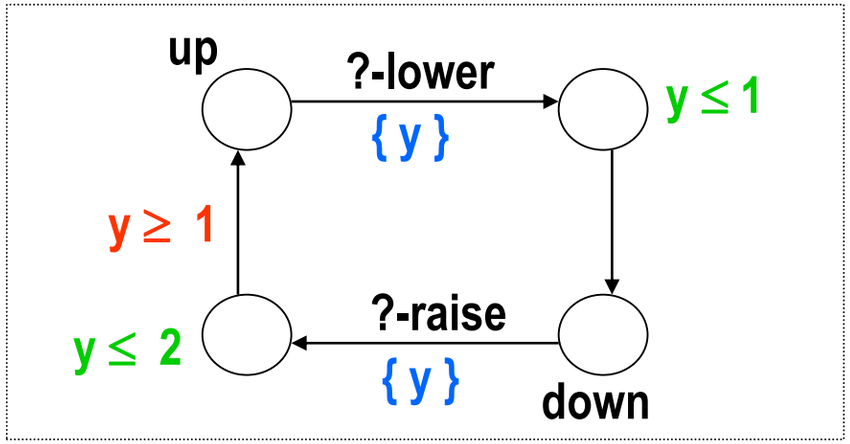


Can the train enter the crossing before the gate is down?

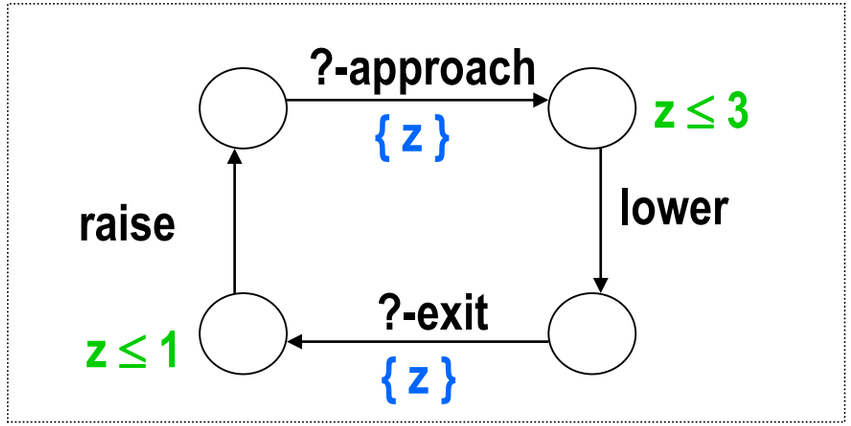
Train



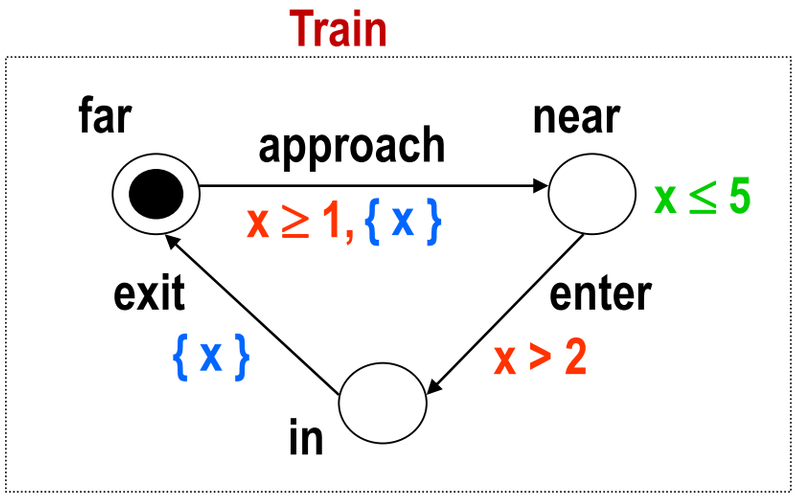
Gate

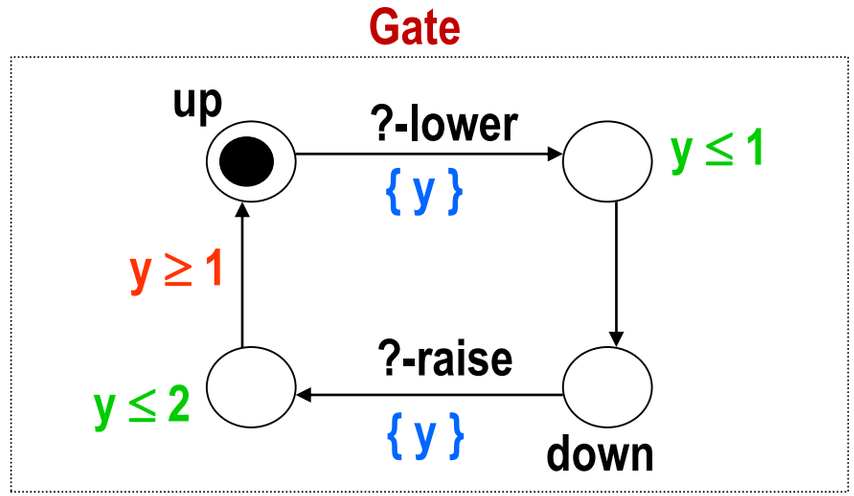


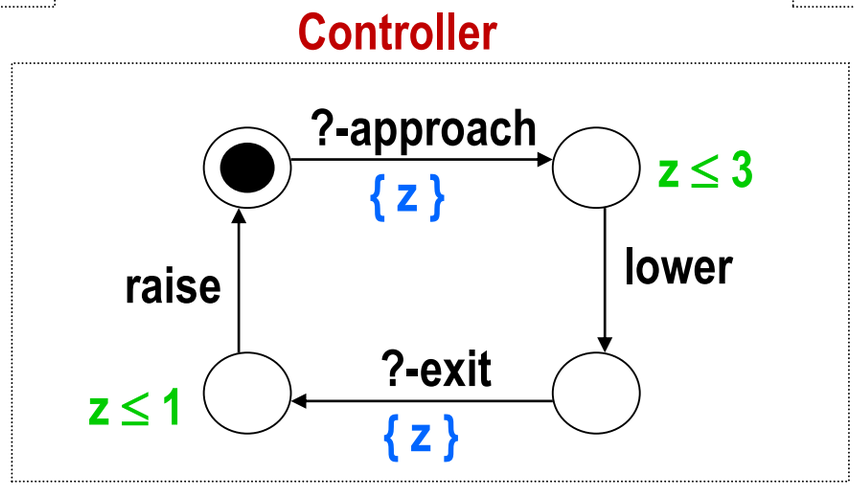
Controller



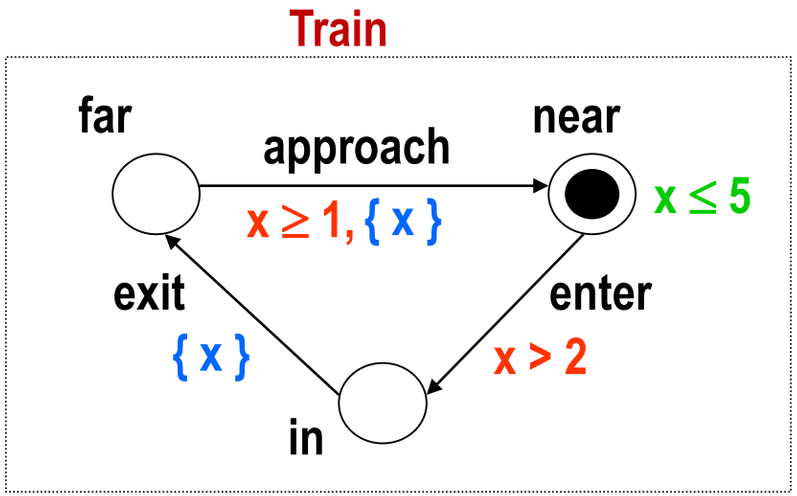
Rail Gate Crossing

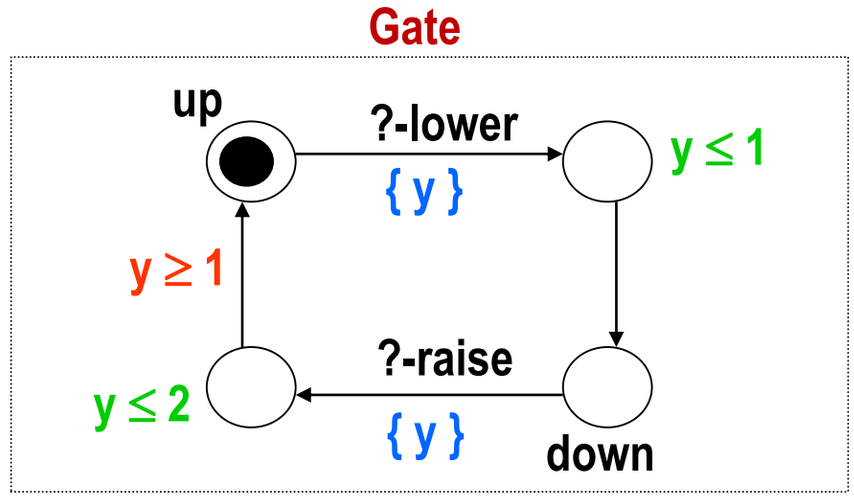


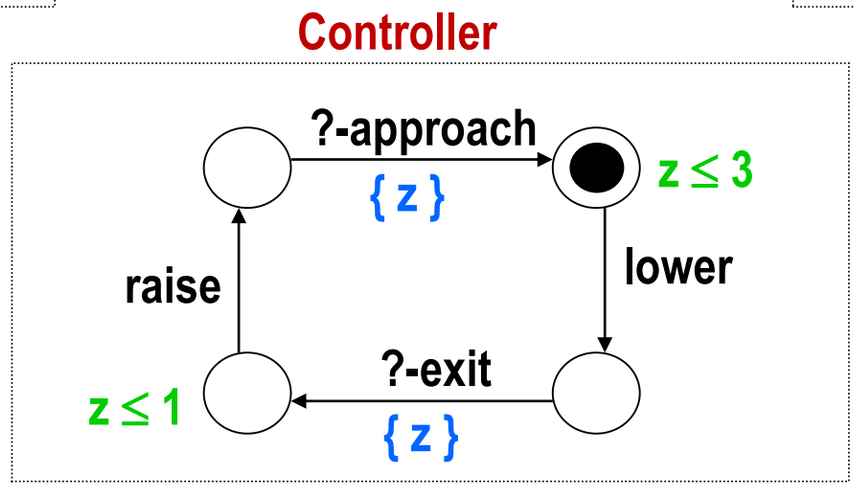


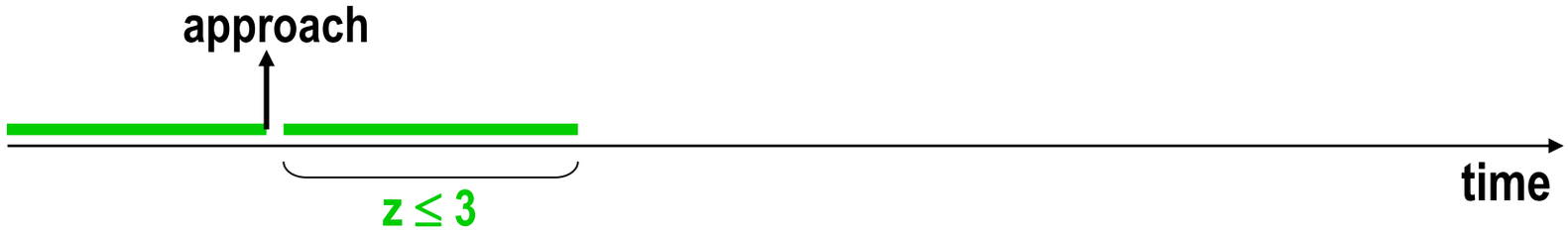


Rail Gate Crossing

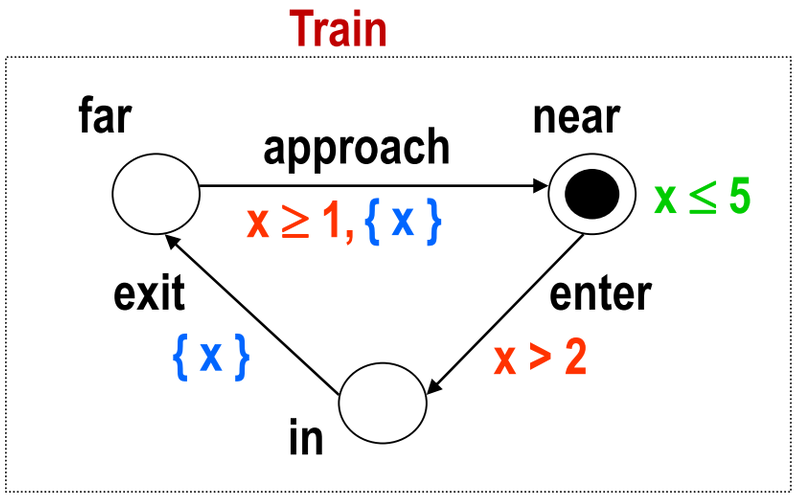


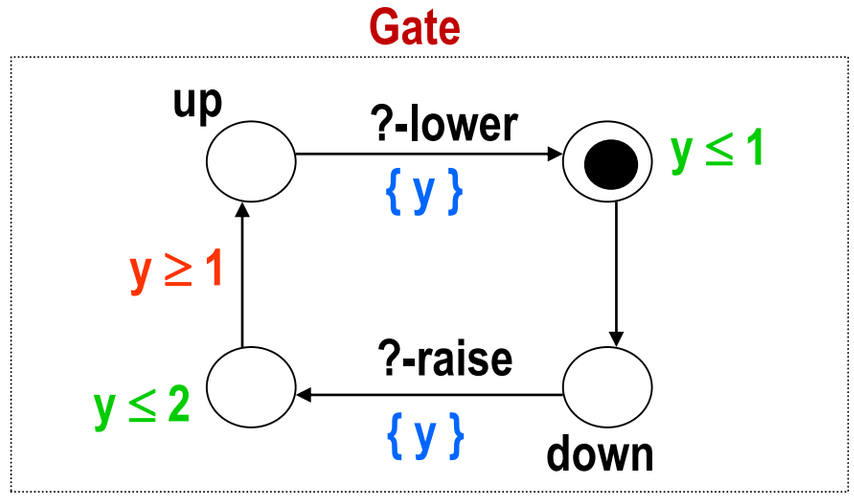


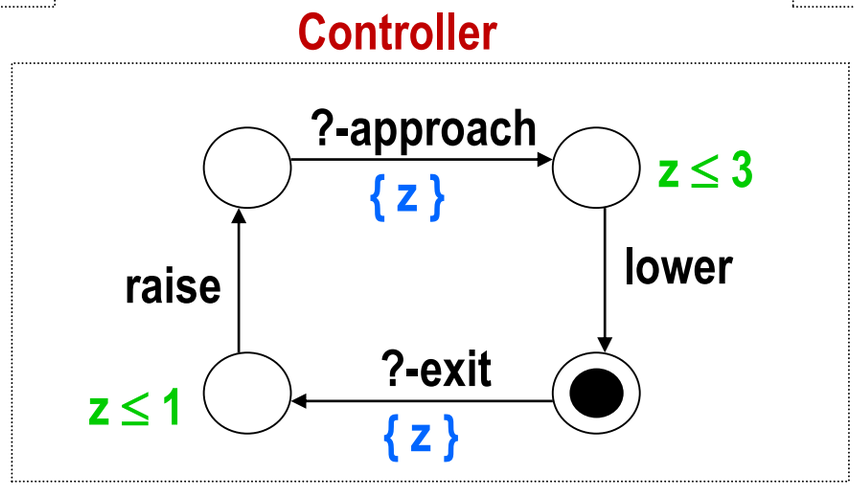


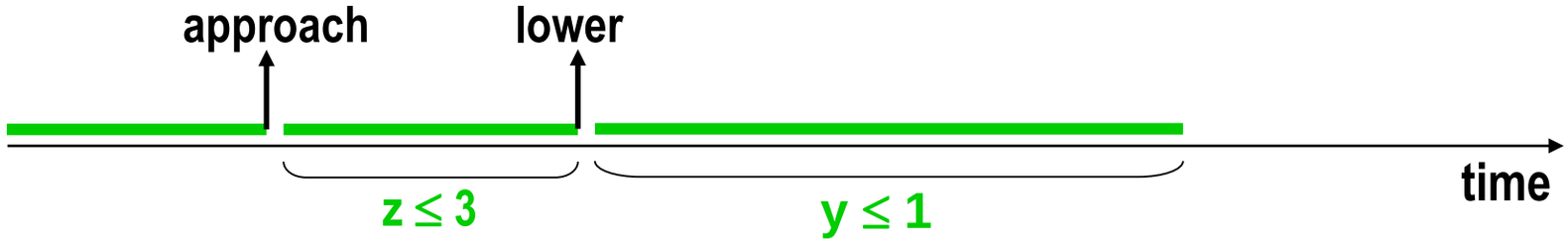


Rail Gate Crossing

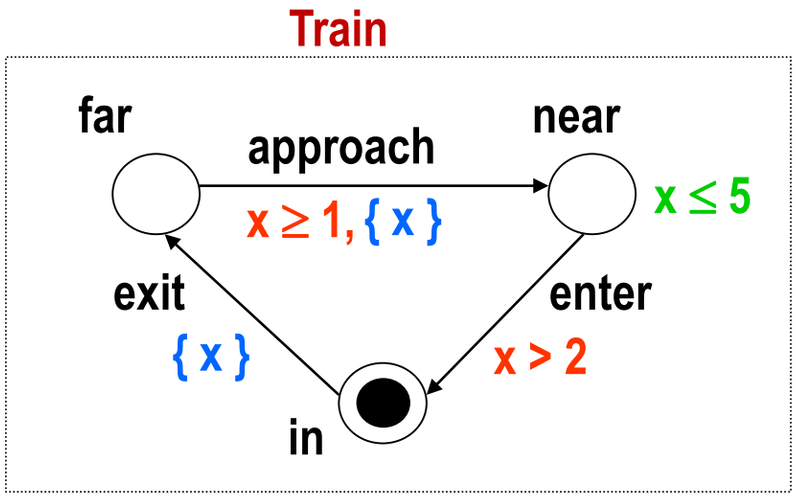


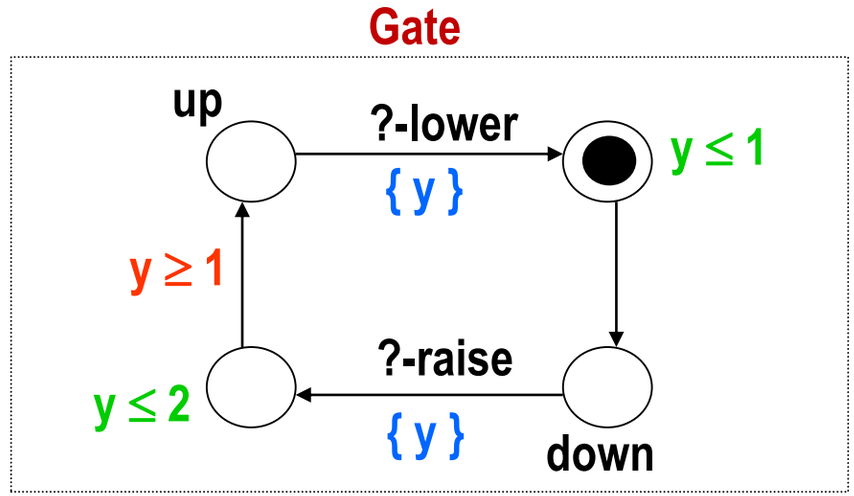


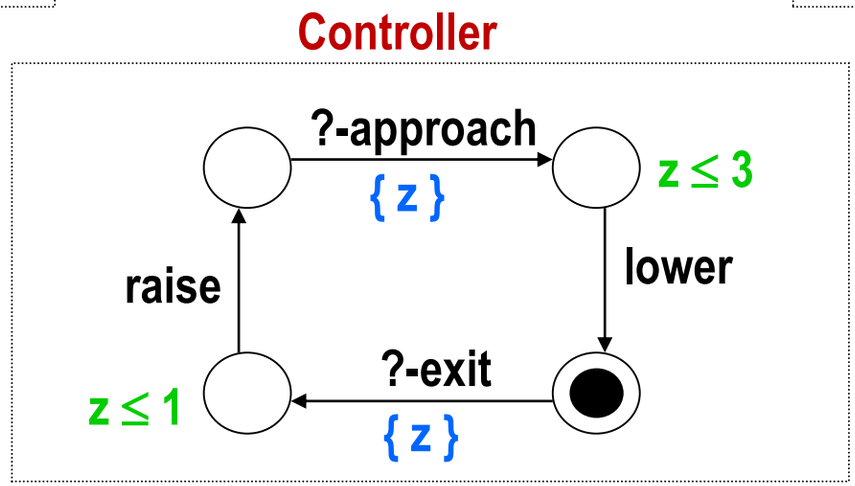


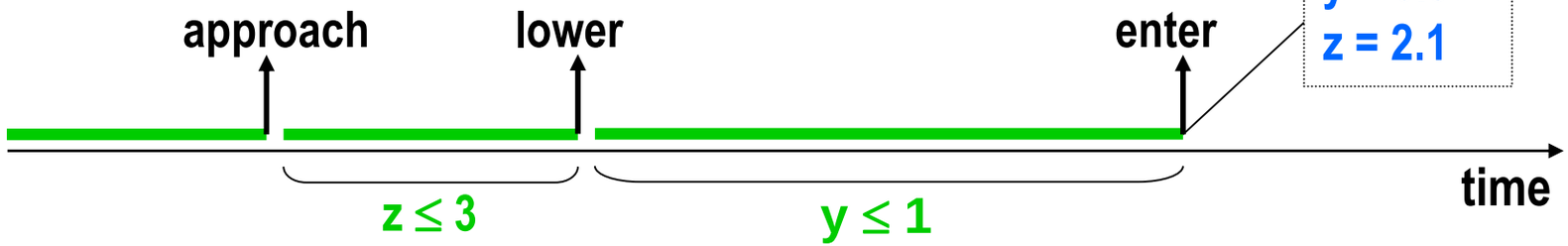


Rail Gate Crossing









Time Convergence, Timelocks, Zenoness

- Not all paths in a timed automaton represent realistic behaviours.
- Three essential phenomena:
  - Time Convergence
  - Timelock
  - Zenoness

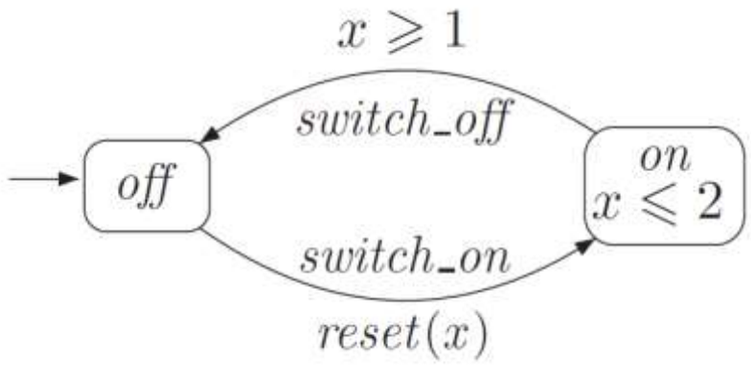
Time Divergence

- A path is time divergent if *the sum of the delays over this path is infinite*.
- Time convergence - a path for which the sum of the delays are bounded by some natural numbers. Time over this path will never increase above a constant.
  - Represents unrealistic behaviours, cannot be avoided in the theory.
  - For analysis ignore time convergent paths and only consider time divergent ones.

Execution time for an execution ρ: $(l_0, v_0) \xrightarrow{\tau_1 a_1} (l_1, v_1) \xrightarrow{\tau_2, a_2}$ ... with $l_0 \in L_0$ is given as ExecTime(ρ) = $\sum_{i=1}^{\infty} \tau_i$

Time divergence: An infinite path fragment π is ***time divergent*** if and only if ExecTime(π) = ∞. Otherwise the path fragment is *time convergent*.

Time Convergent Paths



Time divergent path: ⟨ off, 0 ⟩⟨ off, 1 ⟩⟨ on, 0 ⟩⟨ on, 1 ⟩⟨ off, 1 ⟩⟨ off, 2 ⟩⟨ on, 0 ⟩⟨ on, 1 ⟩⟨ off, 1 ⟩ …

Time convergent path: ⟨ off, 0 ⟩⟨ off, 1/2 ⟩⟨ off, 3/4 ⟩⟨ off, 7/8 ⟩⟨ off, 15/16 ⟩ …

The path is time convergent because

$$\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^{i+1} = 1$$

Timelocks

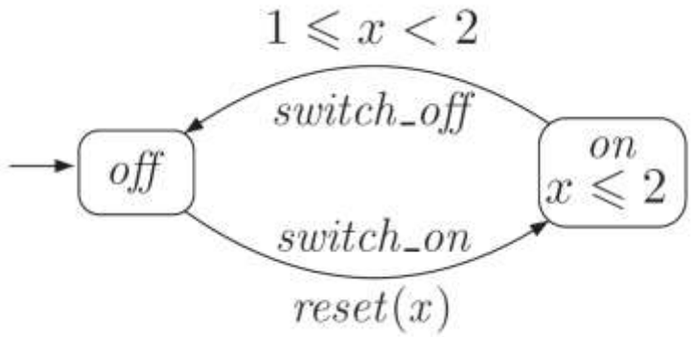
- For a state σ in a timed-automaton, there must be some way for time to progress.
  - If no way is possible, then "σ" has a *timelock.*

> Let $Paths_{div}(\sigma)$ be the set of time-divergent paths starting in σ. A state s contains a timelock iff $Paths_{div}(\sigma) = \Phi$

A timed automaton is timelock-free iff none of its reachable states contains a timelock.

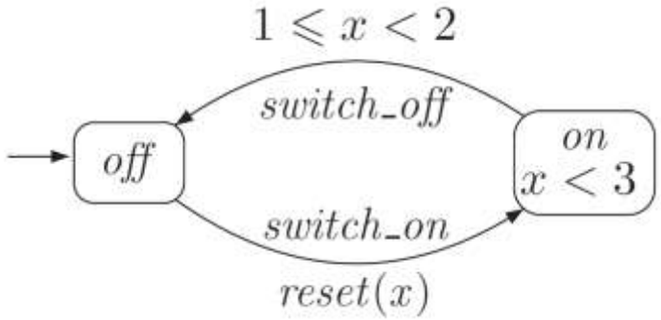
- A timelock is a modeling flaw – should be avoided.

Timelocks



$\langle$ off, 0 $\rangle$ $\xrightarrow{switch\_on}$ $\langle$ on, 0 $\rangle$ $\xrightarrow{2}$ $\langle$ on, 2 $\rangle$

Here the state, $\langle$ on, 2 $\rangle$, is a terminal state and is time locked. Time cannot progress.



$\langle$ off, 0 $\rangle$ $\langle$ on, 2 $\rangle$ $\langle$ on, 2.9 $\rangle$ $\langle$ on, 2.99 $\rangle$ $\langle$ on, 2.999 $\rangle$ $\langle$ on, 2.9999 $\rangle$

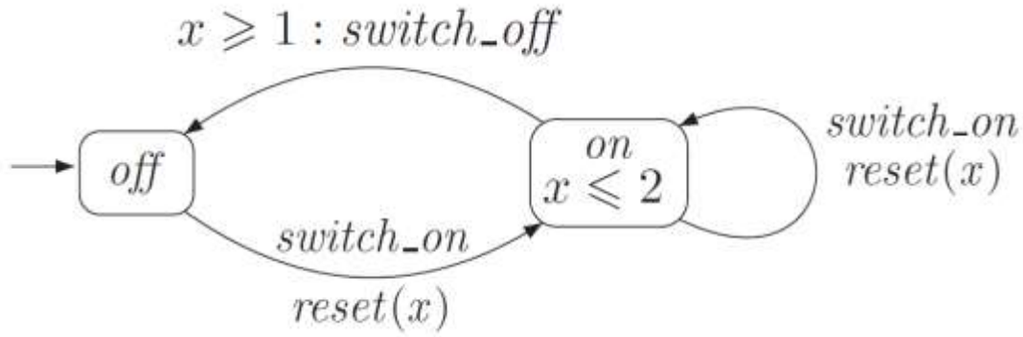
Here also we have reached a time lock. Time can progress, but only along a time convergent path, and therefore up to a finite value.

Zenoness

An infinite path fragment π is zeno if and only if it is time-convergent and infinitely many discrete actions are executed within π.

- Zeno paths represent non-realizable behaviour
  - since their execution would require infinitely fast processors.
- Thus zeno paths are modelling flows and should be avoided.
- To check whether a timed automaton is non-zeno is algorithmically difficult.
- Instead, sufficient conditions are considered that are simple to check, e.g., by static analysis.

Zeno Behaviors

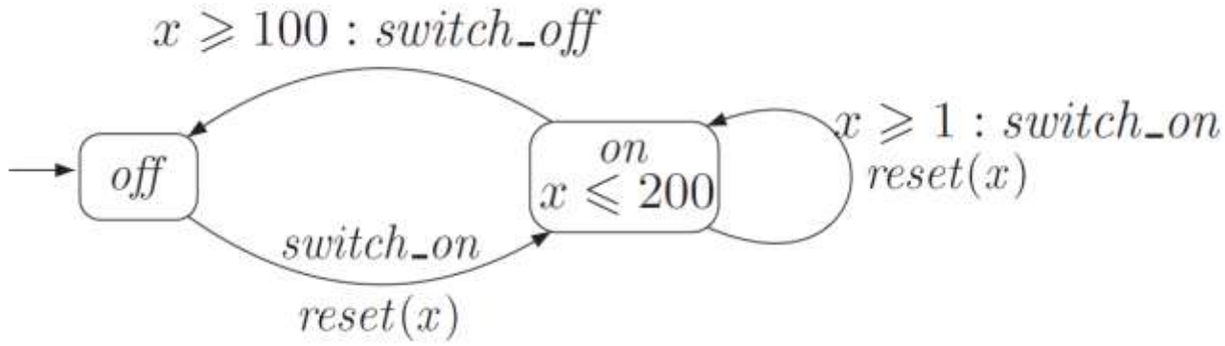


Xeno behaviors

⟨ off, 0 ⟩ —switch_on→ ⟨ on, 0 ⟩ —switch_on→ ⟨ on, 0 ⟩ —switch_on→ ⟨ on, 0 ⟩ —switch_on→

⟨ off, 0 ⟩ —switch_on→ ⟨ on, 0 ⟩ —0.5→ ⟨ on, 0.5 ⟩ —switch_on→ ⟨ on, 0 ⟩ —0.25→ ⟨ on, 0.25 ⟩ —switch_on→

Revised Timed Automaton:



Modeling Exercise-1 *An Elevator*

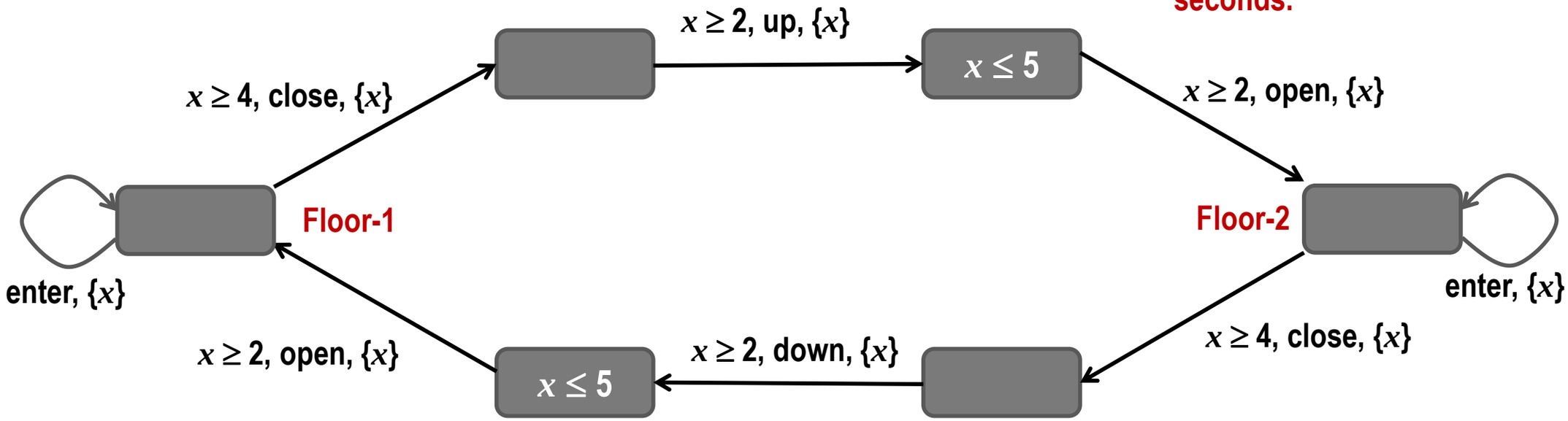
Consider an autonomous elevator which operates between two floors. The requested behavior of the elevator is as follows:

- The elevator can stop either at the ground floor or the first floor.
- When the elevator arrives at a certain floor, its door automatically opens. It takes at least 2 seconds from its arrival before the door opens but the door must definitely open within 5 seconds.
- Whenever the elevator's door is open, passengers can enter. They enter one by one and we (optimistically) assume that the elevator has a sufficient capacity to accommodate any number of passengers waiting outside.
- The door can close only 4 seconds after the last passenger entered. After the door closes, the elevator waits at least 2 seconds and then travels up or down to the other floor.

Suggest a timed automaton model of the elevator. Use the actions up and down to model the movement of the elevator, open and close to describe the door operation and the action enter which means that a passenger is entering the elevator.

Elevator Model

It takes at least 2 seconds from its arrival before the door opens but the door must definitely open within 5 seconds.

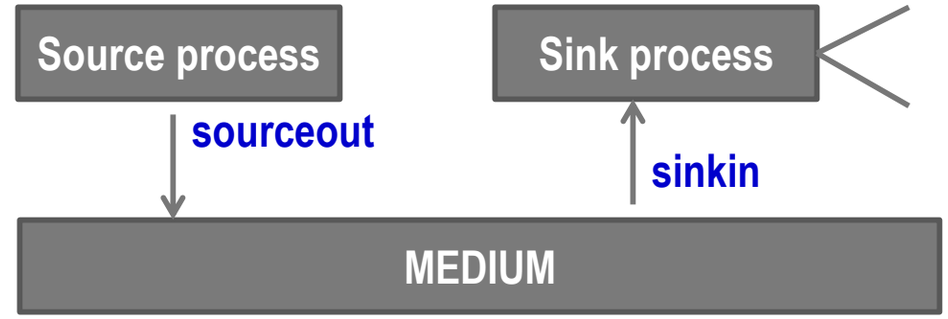


The door can close only 4 seconds after the last passenger entered. After the door closes, the elevator waits at least 2 seconds and then travels up or down to the other floor.

Modeling Exercise-2 *Quality of Service (QoS) of a Media Stream*

Consider the following requirements for a media stream channel and model a possible timed automaton representation.

- **Source emits a message every 50ms (that is, 20 messages per second)**
- **Channel latency is between 80 ms and 90 ms**
- **Channel may loose messages (no more than 20%)**
- **A message is considered lost if it does not arrive within 90 ms**
- **Sink end receives messages and takes 5ms to process each one**
- **An error should be generated if less than 15 messages per second arrive at the sink end**



Modeling Exercise-3 *Refrigerator Door*

When the door of a refrigerator is opened, the light inside the refrigerator turns ON. If the door is kept open for more than one minute, then a beeper is activated. The beeper, when activated produces a beep, and then repeats the beep at least once in every 30 seconds. When the door is closed, the light goes OFF and the beeper (if active) is deactivated.

- List the sensors your timed automaton will use. For each sensor, specify the label you will use to communicate each event sensed by the sensor.
- Draw a timed automaton for each object that is interacting in this system. Draw timed automata for the door controller which controls the light and the beeper. Use appropriate symbols indicating labels used as inputs and outputs. You also need to draw automata for the beeper and the refrigerator light. In addition to the automata mentioned, you may choose to draw supporting automata if required. You DO NOT need to draw the product of the automata.
  - For each automaton, clearly indicate which automaton is being used for each component.
  - Clearly indicate the location names, the timers used in each automaton, the invariants, resets, transition guards, and synchronization labels (inputs/outputs).

  [Note: For simplicity, on a transition you may receive an input event and you may also generate one or more output events on the same transition]

Verification of Timed Systems

- System modeled as a product of timed automata
- Verification problem reduced to reachability or to temporal logic model checking
- Applications
  - Real-time controllers
  - Asynchronous timed circuits
  - Scheduling
  - Distributed timing-based algorithms

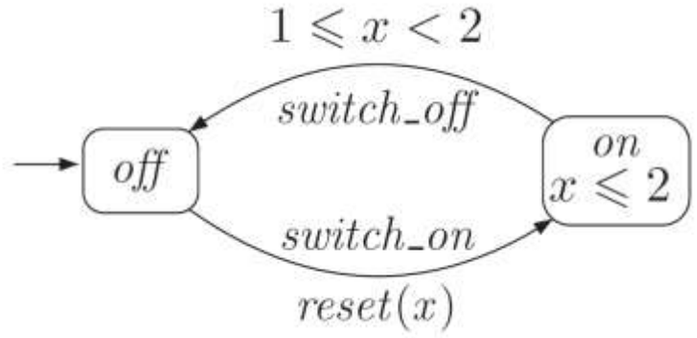
From time to time regions

- The states of a timed automaton: ⟨ loc, v ⟩
  - Where loc refers to the location
  - The k-dimensional vector of values of the k clock variables is denoted by v
  - There are infinite states since the clock variables have *dense* real domains
  - Important question: *Can we discretize the domains of the clock variables into time regions so that all values in a time region can be treated as equivalent?*

In the timed automaton shown here:

- We are concerned with the following intervals for x:

  ⟨ 0, 0 ⟩ , ⟨ 0, 1 ⟩, ⟨ 1, 1 ⟩, ⟨ 1, 2 ⟩, ⟨ 2, 2 ⟩, ⟨ 2, ∞ ⟩
- We can build a *region automaton* which is an abstraction of the timed automaton with states: ⟨ loc, region ⟩
- This is a type of *predicate abstraction* using predicates over the clock variables



Timed Automata and Reachability Abstractions

The *REGION GRAPH*: (for a constraint set C)

- Set of Regions:
  - $\mathcal{R}$ (a finite partition of $\mathbb{R}_{+}^{\mathcal{X}}$, the set of valuations) is a set of regions (for C) if the following all hold:
    1. For every g ∈ C and for every R ∈ $\mathcal{R}$: R ⊆ $[\![g]\!]$ OR $[\![g]\!]$ ∩ R = Φ. In other words, R either belongs fully inside the region satisfying g or totally outside the regions satisfying g.
    2. For all R,R' ∈ $\mathcal{R}$, if there exists v ∈ R and t ∈ $\mathbb{R}$ with v + t ∈ R' then for every v' ∈ R there exists a t' ∈ $\mathbb{R}$ with v' + t' ∈ R'. In other words, every point in a region reaches the same next region as time progresses.
    3. For all R,R' ∈ $\mathcal{R}$, for every Y ⊆ $\mathcal{X}$ if $R_{[Y\leftarrow 0]}$ ∩ R' ≠ Φ, then $R_{[Y\leftarrow 0]}$ ⊆ R'. In other words, every point in a region reaches the same next region when the clocks in $\mathcal{X}$ are reset.

Standard Regions

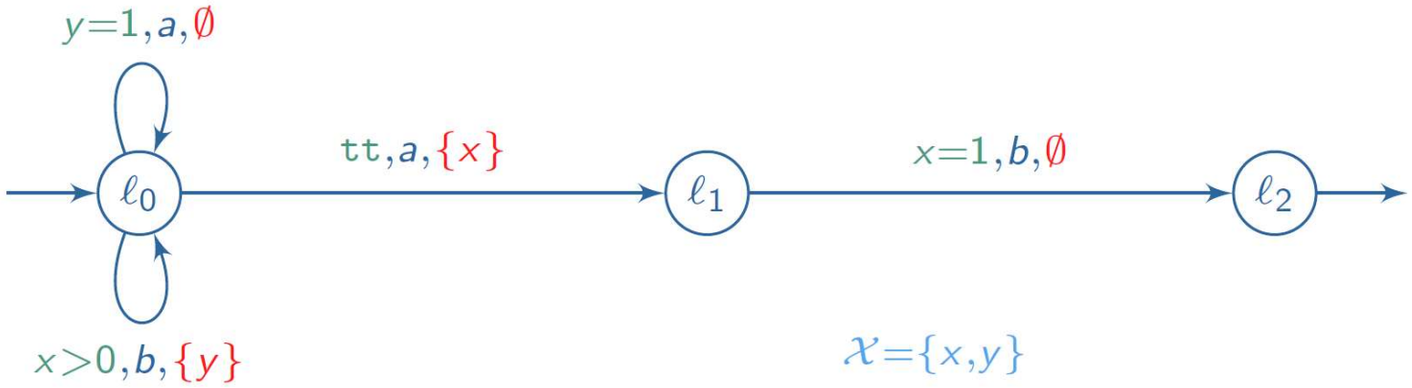
- Let M be the maximum constant in A.

  The following equivalence relation yields the set of *standard regions*:

  $$v \equiv^{M} v' \text{ if for every } x,y \in \mathcal{X}$$

  - $v(x) > M \Leftrightarrow v'(x) > M$
  - $v(x) \leq M \Longrightarrow (\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor)$ and $(\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0)$ where $\{v(x)\}$ refers to the fractional part of $v(x)$
  - $(v(x) \leq M$ and $v(y) \leq M) \Longrightarrow (\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\})$

Standard Regions (what do the rules mean?)



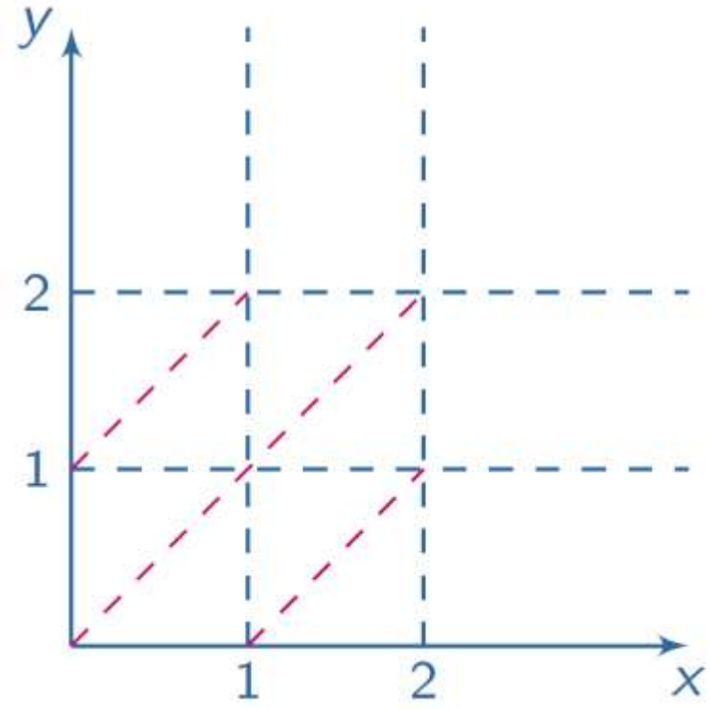
2 Clocks = 2 dimensions

The partition is compatible with constraints, time elapsing and resets.

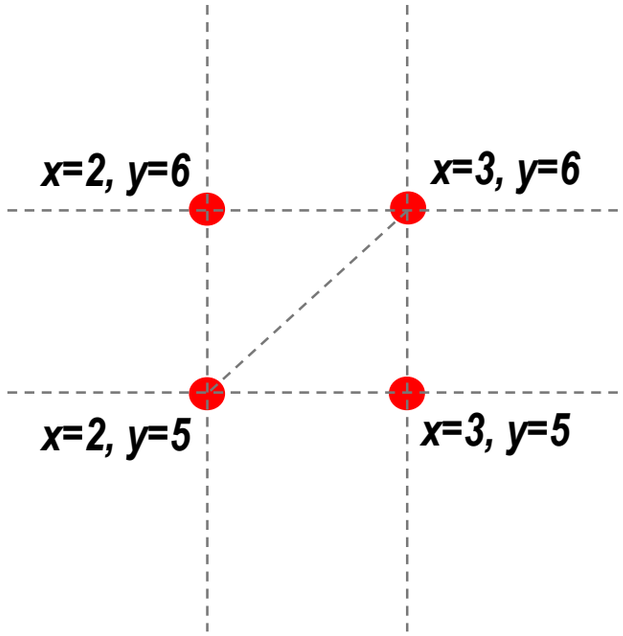
$v \equiv^{M} v'$ **if for every x,y** $\in \mathcal{X}$

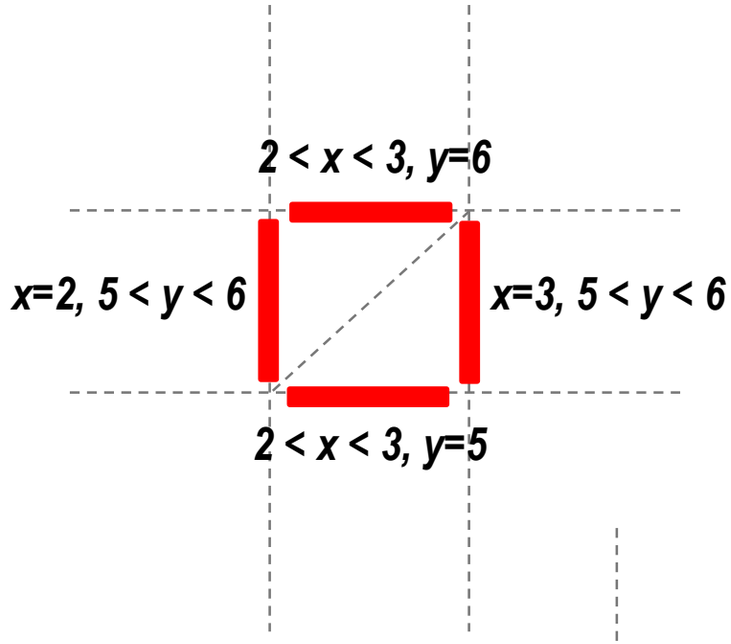
- $v(x) > M \Leftrightarrow v'(x) > M$
- $v(x) \leq M \Longrightarrow (\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor)$ and $(\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0)$
- $(v(x) \leq M$ and $v(y) \leq M) \Longrightarrow (\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\})$

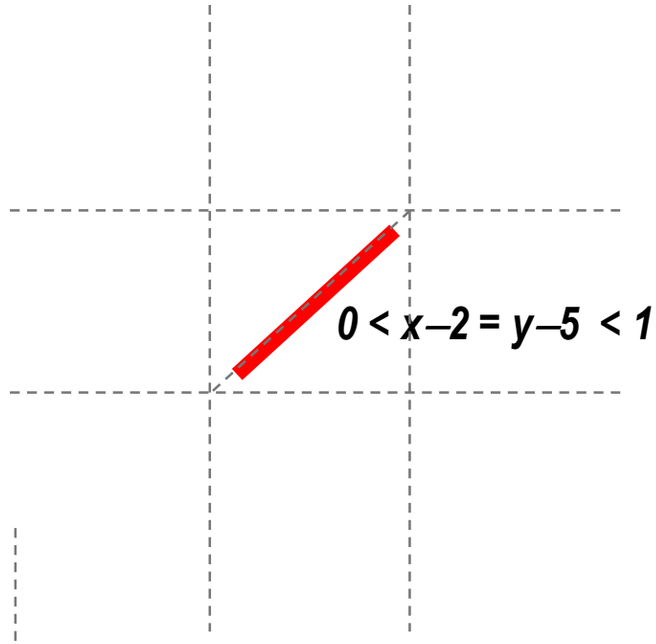
where {v(x)} refers to the fractional part of v(x)

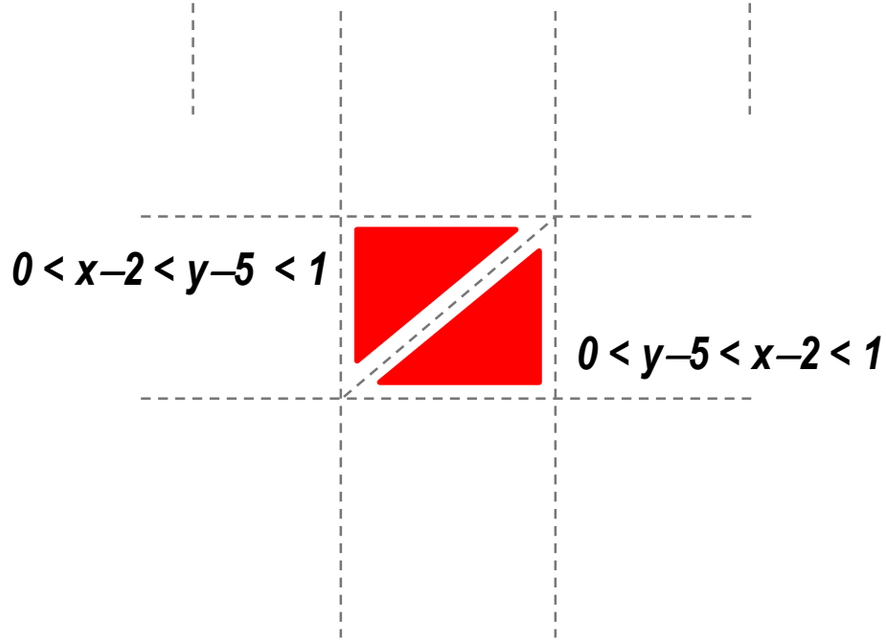


Examples of Standard Regions





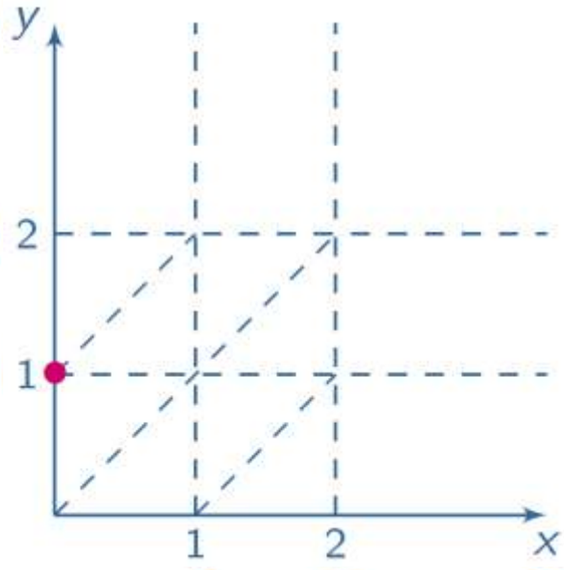




Operations on Regions

For two clocks, the (bounded) regions have the following shapes:

- $R_{[Y \leftarrow 0]}$ denotes the region obtained from R by resetting clocks in $Y \subseteq \mathcal{X}$.
- R' is a time-successor of R if there exists $v' \in R'$, $v \in R$, $t \in \mathbb{R}_+$ with $v' = v + t$



$(x=0,y=0) \xrightarrow{delay} (0<x=y<1) \xrightarrow{y:=0} (0<x<1,y=0)$

$\xrightarrow{delay} (0<y<x<1) \xrightarrow{delay} (0<y<1=x) \xrightarrow{delay}$

$(1<x<2,0<y<1,\{x\}<\{y\}) \xrightarrow{delay} (y=1<x<2)$

$\xrightarrow{x:=0} (x=0,y=1)$

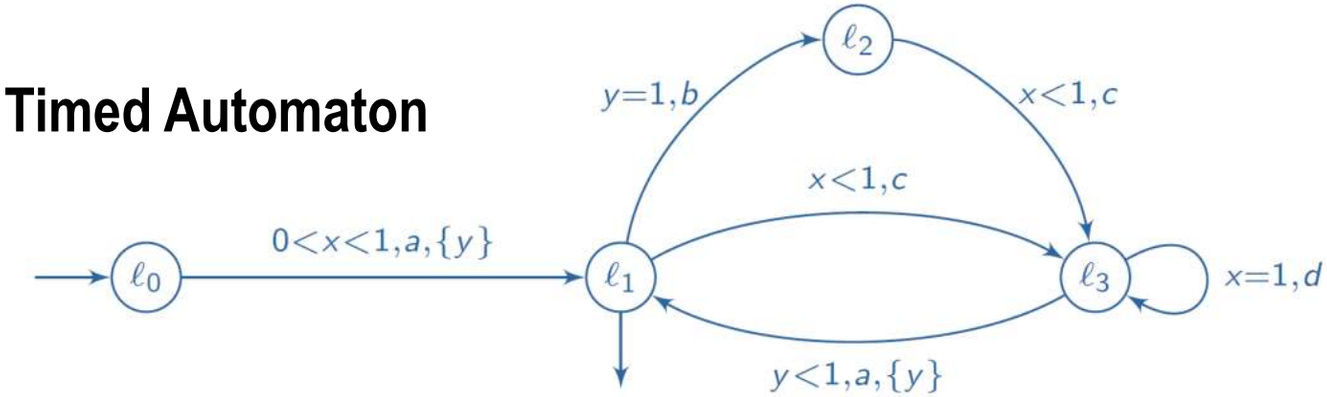
Region Automaton

From a timed automaton A we build a finite region automaton $\alpha(A)$ as follows:

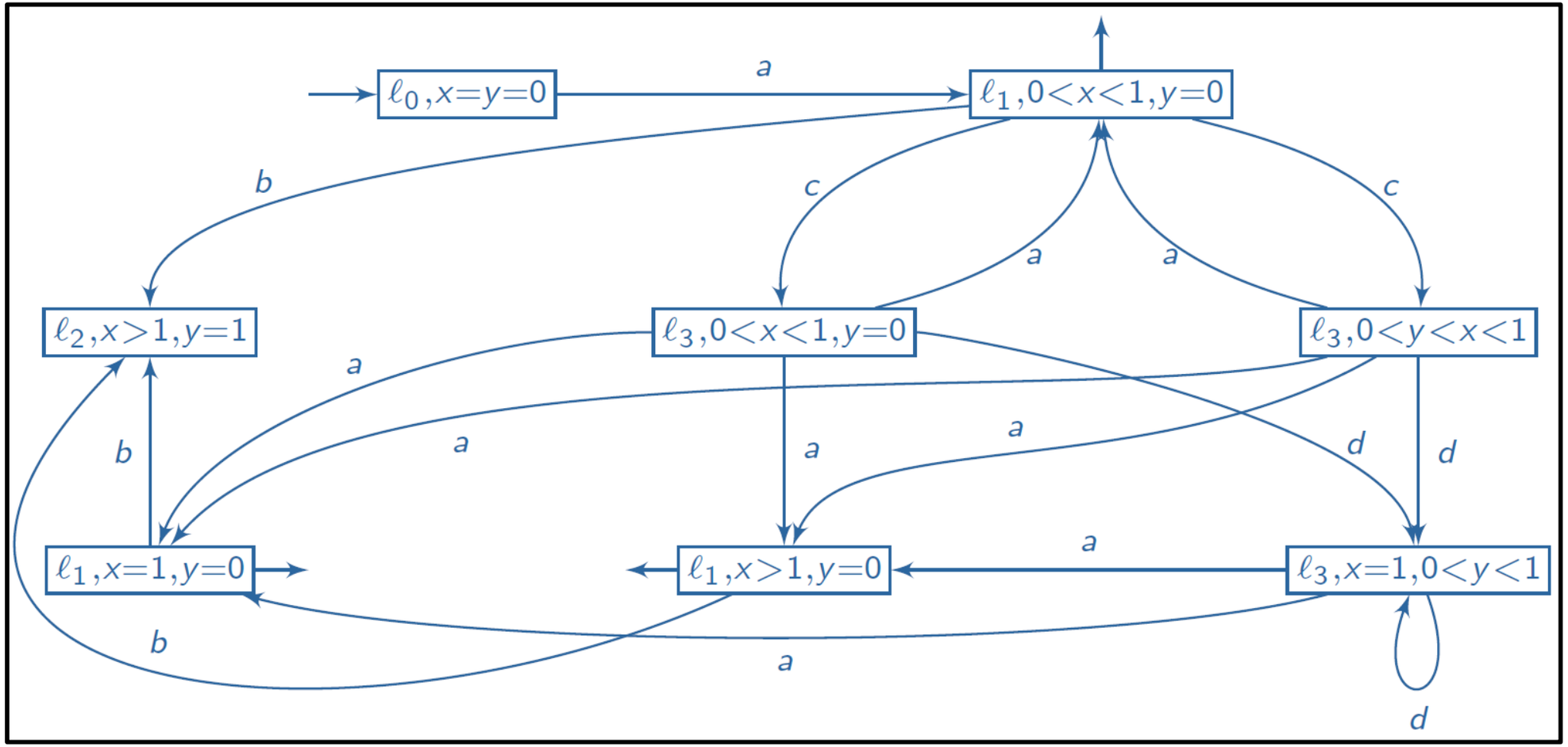
- States: $L \times \mathcal{R}$
- Initial Set: $L_0 \times \mathcal{R}$
- Final Set : $L_{acc} \times \mathcal{R}$
- Transitions: $(l, R) \xrightarrow{a} (l' , R')$ if there exists
  - $l \xrightarrow{g,a,Y} l'$ in A,
  - R" time-successor of R with R" $\subseteq [\![g]\!]$ and R' = R"$_{[Y \leftarrow 0]}$.

Example

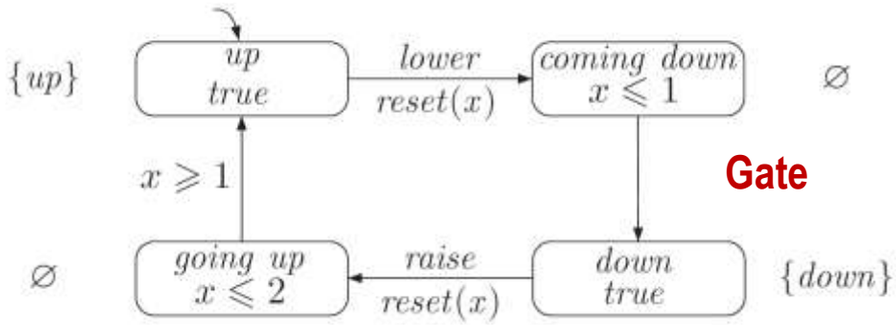
Timed Automaton



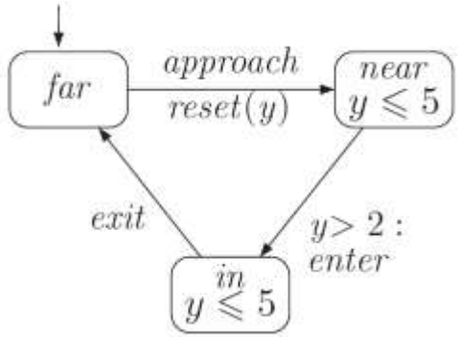
Region Automaton



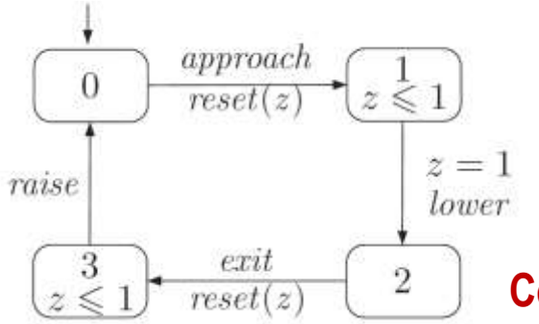
Product Construction: Train || Controller || Gate



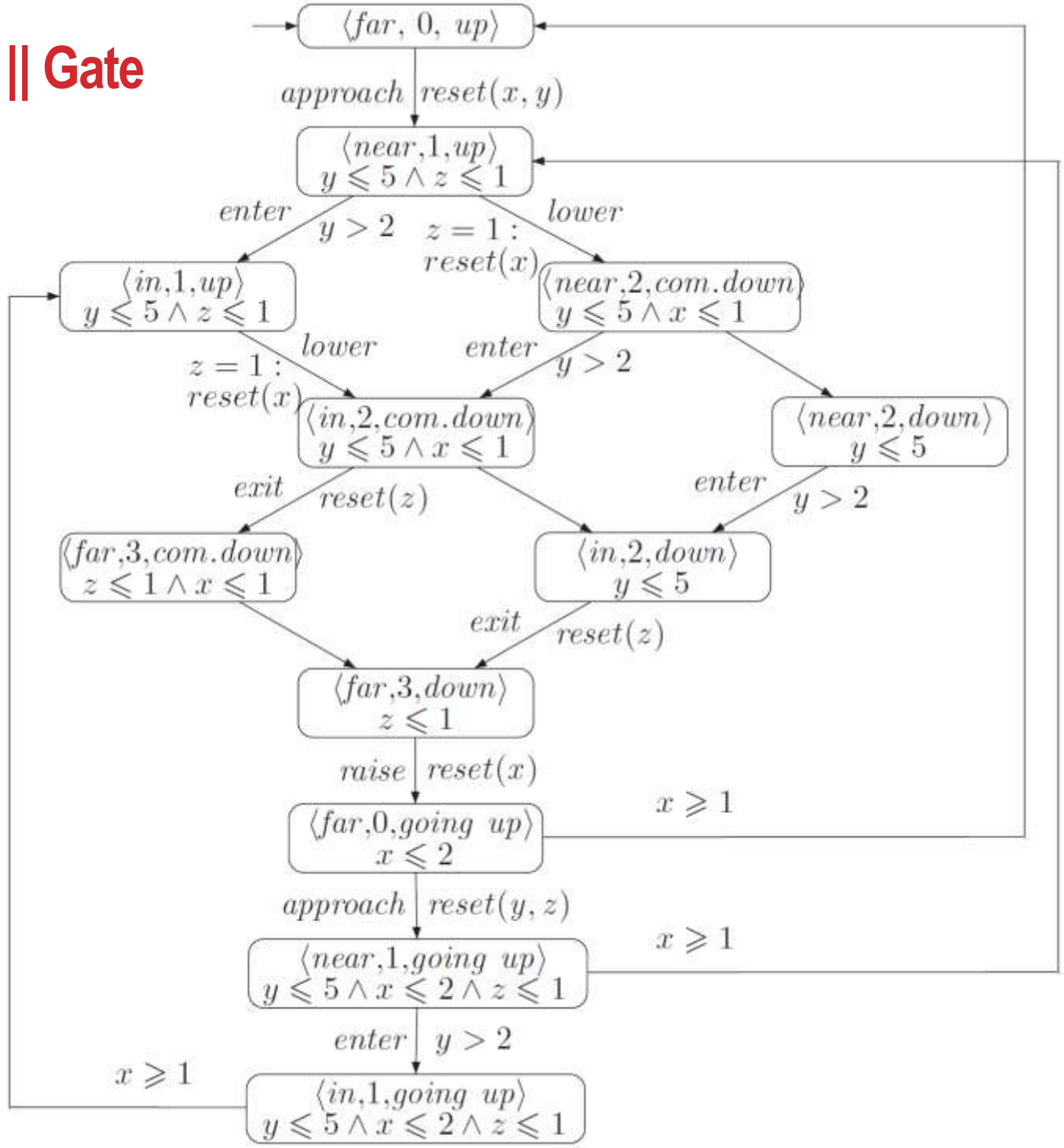
Gate



Train



Controller



A Logic for Timed Automata – Timed CTL

TCTL = CTL + Time

A TCTL formula uses one or more *formula clocks*.

Syntax: $\emptyset ::= p \mid \alpha \mid \neg\emptyset \mid \emptyset \vee \emptyset \mid z\ in\ \emptyset \mid E[\emptyset\ U\ \emptyset] \mid A[\emptyset\ U\ \emptyset]$

- **p ∈ AP, atomic propositions**
- **z ∈ D, formula clocks**
- **α – constraints over formula clocks and automata clocks**
- **z in ∅ - "freeze operator" introduces a new formula clock z**
- **$E[\emptyset\ U\ \emptyset]$ | $A[\emptyset\ U\ \emptyset]$ - As in CTL**
- **No [E X ∅]**

Derived Operators

$A [\phi\ U_{\leq 7}\ \psi]$ = z in A $[(\phi \wedge z \leq 7)\ U\ \psi]$

Along any path, ϕ hold continuously until within 7 time units ψ becomes valid.

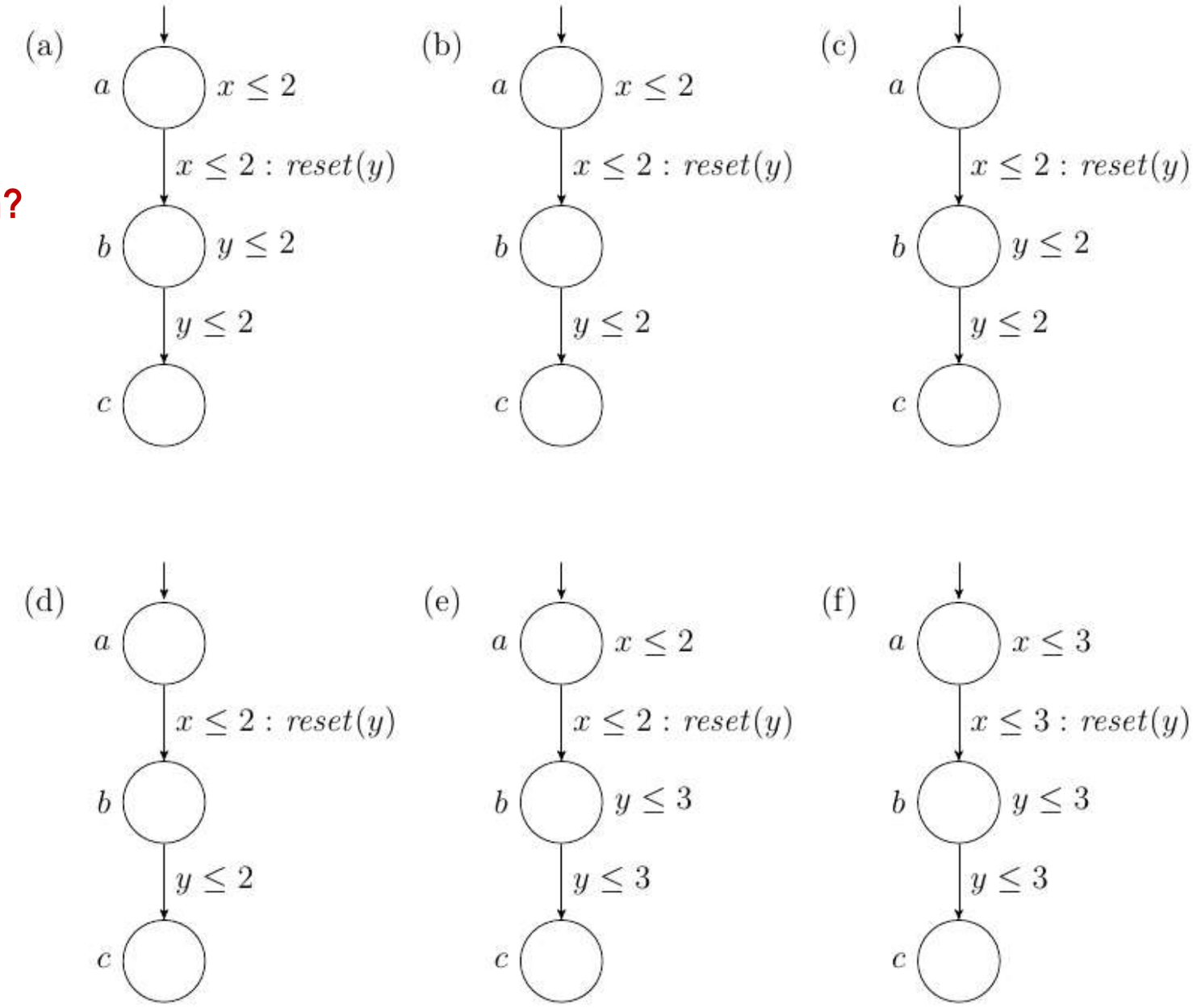
$E [F_{\geq 5}\ \phi]$ = z in EF $[(z \geq 5) \wedge \phi]$

There is a path on which the property ϕ becomes valid at or after 5 time units.

TCTL

Which formula is true in which automaton?

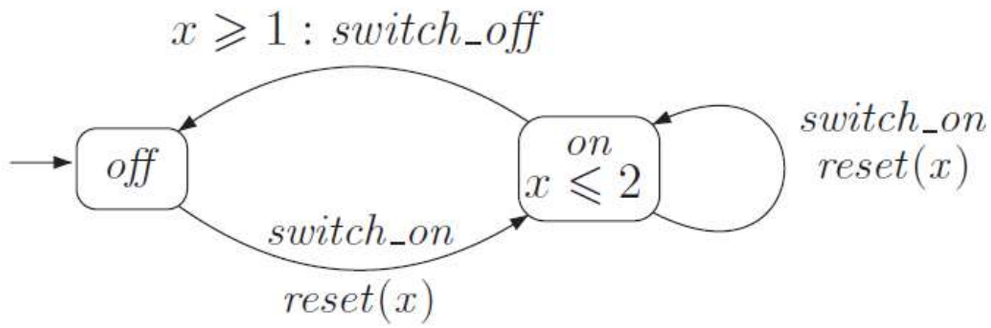
1. $AF_{\leq 4}\, c$
2. $AFEG\ b$
3. $(AF_{\leq 5}\, c) \wedge (EG_{\leq 5} \neg c)$
4. $(EG\ a) \wedge (EFEG\ b)$
5. $(EG\ a) \wedge (\neg\ EFEG\ b)$
6. $(AF_{\leq 6}\, c) \wedge (EG_{\leq 6} \neg c)$



Answer:

1-(a) 2-(b) 3-(e) 4-(d) 5-(c) 6-(f)

Difference between CTL and TCTL



Consider the property: **AG(on => AF off)**

- **If we treat this as a CTL property, then the automaton above does not satisfy it, because it can stay in the *on*-state forever**
- **If we treat this as a TCTL property, then the automaton satisfies it since TCTL properties are interpreted only over time divergent paths, and all time divergent paths must return to the *off*-state**

Variants of timed automata

Many variants of TA exist:

- **Diagonal constraints:**

  Guards are conjunctions of constraints of the form $x \bowtie c$ and $x - y \bowtie c$.

- **Additive clock constraints:**

  constraints of the form $x \bowtie c$ and $x + y \bowtie c$.

- **Epsilon transitions:**

  Actions from the alphabet $\Sigma \cup \{\epsilon\}$

- **Updatable timed automata:**

  Clocks updates of the form: $x: \bowtie c$ and $x: \bowtie y + c$

Moving forward from Timed Automata

- Regular - Finite (Deterministic / Non-Deterministic Finite Automata)
  - Locations = States, Memory is finite – states are finite.
  - Discrete actions only
- Transition Systems
  - Finite location systems : possibly infinite states (with variables)
  - Discrete actions only
- Timed Automata
  - Finite location systems – timers : possibly infinite states (When would a TA have finite states?)
  - Discrete and Delay actions
- Hybrid Automata
  - Finite location systems – beyond timers : possibly infinite states
  - Discrete actions and Custom activities