# 3. Real-time models: Timed Automata

Renato Neves     José Proença

CPC 2022/2023

Cyber Physical Computation

CISTER – ISEP, Porto, Portugal

U.Minho, Braga, Portugal

https://lmf.di.uminho.pt/CyPhyComp2223/

https://haslab.github.io/MFP/PCF/2223/

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

# Motivation

# Motivation

Specifying an airbag saying that in a car crash the airbag eventually inflates maybe not enough, but:

in a car crash the airbag eventually inflates within 20ms

> *Correctness in time-critical systems not only depends on the logical result of the computation, but also on the time at which the results are produced*
>
> [Baier & Katoen, 2008]

# Examples of time-critical systems

### Network-based traffic lights

Lights activate at very specific time intervals.

### Bounded retransmission protocol

Communication of large files between a remote control unit and a video/audio equipment.
Correctness relies on:

- transmission and synchronization delays
- time-out values

### And many others...

- medical instruments
- hybrid systems (e.g. for cruise controllers)

# Syllabus

- CSS: a simple language for concurrency
  - Syntax
  - Semantics
  - Equivalence
- Timed Automata
  - Syntax
  - Semantics (composition, Zeno)
  - Equivalence
  - UPPAAL tool
    - Specification
    - CTL and Verification
- A simple C-like language
  - Syntax
  - Semantics (operational)
- Hybrid-language: adding differential equations
  - Syntax
  - Semantics
  - Lince tool
    - Specification
    - Analysis
- Monads: semantics with computational effects

# Table of contents

# Motivation

- timed transition systems, timed Petri nets, timed IO automata, timed process algebras and other formalisms associate lower and upper bounds to transitions, but no time constraints to transverse the automaton.
- Expressive power is often somehow limited and infinite-state LTS (introduced to express dense time models) are difficult to handle in practice

# Motivation

**Example**

Typical process algebra tools are unable to express a system which has one action $a$ which can only occur at time point 5 with the effect of moving the system to its initial state.

This example has, however, a simple description in terms of time measured by a stopwatch:

1. Set the stopwatch to 0
2. When the stopwatch measures 5, action $a$ can occur. If $a$ occurs go to 1., if not idle forever.

# Motivation

This suggests resorting to an automaton-based formalism with an explicit notion of clock (stopwatch) to control availability of transitions.

Timed Automata [Alur & Dill, 90]

- emphasis on decidability of the reachability problem and corresponding practically efficient algorithms
- infinite underlying timed transition systems are converted to finitely large symbolic transition systems where reachability becomes decidable (region or zone graphs)

Associated tools

- UPPAAL [Behrmann, David, Larsen, 04]
- IMITATOR [André, 09]
- PRISM [Parker, Kwiatkowska, 00]
- KRONOS [Bozga, 98]

# Motivation

UPPAAL = (Uppsala University + Aalborg University) [1995]

- A toolbox for modeling, simulation and verification of real-time systems
- where systems are modeled as networks of timed automata enriched with integer variables, structured data types, channel syncronisations and urgency annotations
- Properties are specified in a subset of CTL

`www.uppaal.org`

# Timed Automata Definition

# Timed automata

Finite-state machine equipped with a finite set of real-valued clock variables (clocks)

Clocks

- clocks can only be read or
- reset to zero, after which they start increasing their value implicitly as time progresses
- the value of a clock corresponds to time elapsed since its last reset
- all clocks proceed synchronously (at the same rate)

# Timed automata Definition

$$\langle \mathbf{L}, \mathbf{L_0}, \mathbf{Act}, \mathbf{C}, \mathbf{Tr}, \mathbf{Inv} \rangle$$

where

- $L$ is a set of locations, and $L_0 \subseteq L$ the set of initial locations
- $Act$ is a set of actions and $C$ a set of clocks
- $Tr \subseteq L \times \mathcal{C}(C) \times Act \times \mathcal{P}(C) \times L$ is the transition relation

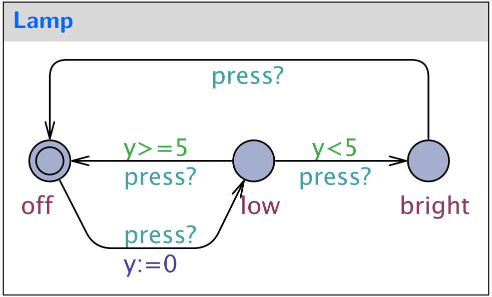
  $$\ell_1 \xrightarrow{g,a,U} \ell_2$$

  denotes a transition from location $\ell_1$ to $\ell_2$, labelled by $a$, enabled if guard $g$ is valid, which, when performed, resets the set $U$ of clocks
- $Inv : L \longrightarrow \mathcal{C}(C)$ is the assignment of invariants to locations

where $\mathcal{C}(C)$ denotes the set of clock constraints over a set $C$ of clock variables

# Example: the lamp interrupt

(extracted from UPPAAL)




**Ex. 3.1:** Define $\langle L, L_0, Act, C, Tr, Inv \rangle$.

# Clock constraints

$\mathcal{C}(C)$ denotes the set of clock constraints over a set $C$ of clock variables. Each constraint is formed according to

$$g \ ::= \ x \,\square\, n \ \mid \ x - y \,\square\, n \ \mid \ g \wedge g \ \mid \ \mathit{true}$$

where $x, y \in C$, $n \in \mathbb{N}$ and $\square \in \{<, \leq, >, \geq, =\}$

used in

- transitions as guards (enabling conditions)

  a transition cannot occur if its guard is invalid

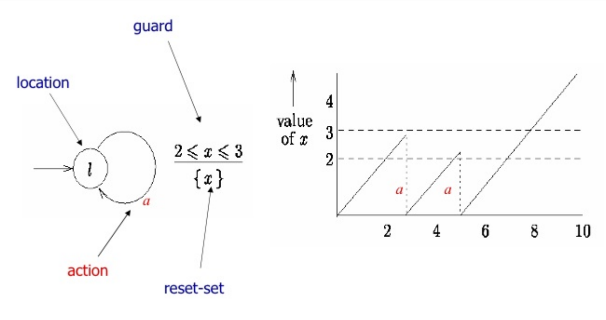
- locations as invariants (safety specifications)

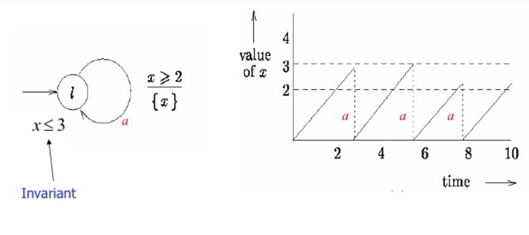
  a location must be left before its invariant becomes invalid

**Note**

Invariants are the only way to force transitions to occur

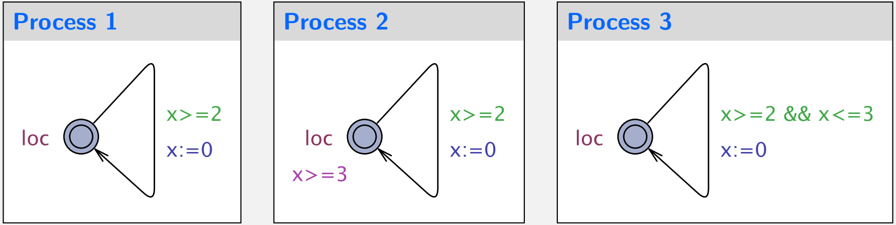
# Guards, updates & invariants



# Transition guards & location invariants

## Demo (in Uppaal)



# Parallel Composition

# Parallel composition of timed automata

- Action labels as channel identifiers
- Communication by forced handshacking over a subset of common actions
- Is defined as an automaton construction over a finite set of timed automata originating a so-called network of timed automata

# Parallel composition of timed automata

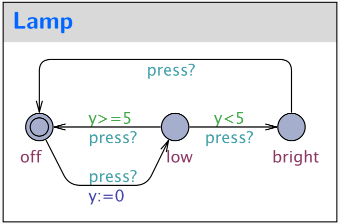
Let $H = Act_1 \cap Act_2$. The parallel composition of $ta_1$ and $ta_2$ synchronizing on $H$ is the timed automata

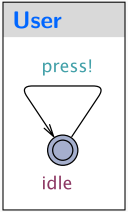
$$ta_1 \parallel_H ta_2 := \langle L_1 \times L_2, L_{0,1} \times L_{0,2}, Act_{\parallel_H}, C_1 \cup C_2, Tr_{\parallel_H}, Inv_{\parallel_H} \rangle$$

where

- $Act_{\parallel_H} = ((Act_1 \cup Act_2) - H) \cup \{\tau\}$
- $Inv_{\parallel_H} \langle \ell_1, \ell_2 \rangle = Inv_1(\ell_1) \wedge Inv_2(\ell_2)$
- $Tr_{\parallel_H}$ is given by:
  - $\langle \ell_1, \ell_2 \rangle \xrightarrow{g,a,U} \langle \ell_1', \ell_2 \rangle$ if $a \notin H \wedge \ell_1 \xrightarrow{g,a,U} \ell_1'$
  - $\langle \ell_1, \ell_2 \rangle \xrightarrow{g,a,U} \langle \ell_1, \ell_2' \rangle$ if $a \notin H \wedge \ell_2 \xrightarrow{g,a,U} \ell_2'$
  - $\langle \ell_1, \ell_2 \rangle \xrightarrow{g,\tau,U} \langle \ell_1', \ell_2' \rangle$ if $a \in H \wedge \ell_1 \xrightarrow{g_1,a,U_1} \ell_1' \wedge \ell_2 \xrightarrow{g_2,a,U_2} \ell_2'$
    with $g = g_1 \wedge g_2$ and $U = U_1 \cup U_2$

# Example: the lamp interrupt as a closed system



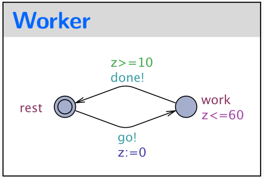



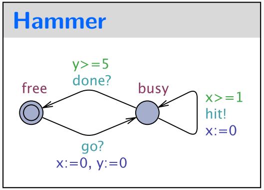

**Uppaal:**

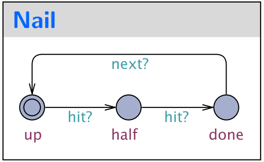
- takes $H = Act_1 \cap Act_2$ (actually as complementary actions denoted by the ? and ! annotations)
- only deals with closed systems

**Ex. 3.2:** Define the TA of the composition.

# Exercise: worker, hammer, nail




**Ex. 3.3: Define the TA of the composition.**

# Semantics

# Timed Labelled Transition Systems

| Syntax | Semantics |
|---|---|
| *How to write* | *How to execute* |
| Process Algebra | LTS (Labelled Transition Systems) |
| Timed Automaton | TLTS (Timed LTS) |

# Timed Labelled Transition Systems

| Syntax | Semantics |
|---|---|
| *How to write* | *How to execute* |
| Process Algebra | LTS (Labelled Transition Systems) |
| Timed Automaton | TLTS (Timed LTS) |

## Timed LTS

Introduce delay transitions to capture the passage of time within a LTS:

$$s \xrightarrow{a} s' \quad \text{for } a \in Act\text{, are ordinary transitions due to action occurrence}$$

$$s \xrightarrow{d} s' \quad \text{for } d \in \mathcal{R}_0^+\text{, are delay transitions}$$

subject to a number of constraints, eg,

# Dealing with time in system models

### Timed LTS

- time additivity

$$(s \xrightarrow{d} s' \wedge 0 \leq d' \leq d) \Rightarrow s \xrightarrow{d'} s'' \xrightarrow{d-d'} s' \text{ for some state } s''$$

- delay transitions are deterministic

$$(s \xrightarrow{d} s' \wedge s \xrightarrow{d} s'') \Rightarrow s' = s''$$

**Semantics of TA:**

Every TA *ta* defines a TLTS

$$\mathcal{T}(ta)$$

whose states are pairs

$$\langle \text{location}, \text{clock valuation} \rangle$$

with infinitely, even uncountably many states, and infinite branching

# Clock valuations

## Definition

A clock valuation $\eta$ for a set of clocks $C$ is a function

$$\eta : C \longrightarrow \mathcal{R}_0^+$$

assigning to each clock $x \in C$ its current value $\eta\, x$.

## Satisfaction of clock constraints

$$\eta \models x \,\Box\, n \;\Leftrightarrow\; \eta\, x \,\Box\, n$$
$$\eta \models x - y \,\Box\, n \;\Leftrightarrow\; (\eta\, x - \eta\, y) \,\Box\, n$$
$$\eta \models g_1 \wedge g_2 \;\Leftrightarrow\; \eta \models g_1 \wedge \eta \models g_2$$

# Operations on clock valuations

## Delay

For each $d \in \mathcal{R}_0^+$, valuation $\eta + d$ is given by

$$(\eta + d)\,x \;=\; \eta\,x \;+\; d$$

## Reset

For each $R \subseteq C$, valuation $\eta[R]$ is given by

$$\begin{cases} \eta[R]\,x \;=\; \eta\,x & \Leftarrow x \notin R \\ \eta[R]\,x \;=\; 0 & \Leftarrow x \in R \end{cases}$$

# From $ta$ to $\mathcal{T}(ta)$

Let $ta = \langle L, L_0, Act, C, Tr, Inv \rangle$

$$\mathcal{T}(ta) \;=\; \langle S, S_0 \subseteq S, N, T \rangle$$

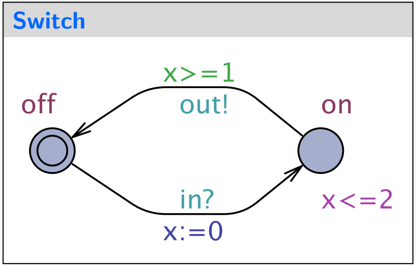
where

- $S = \{\langle l, \eta \rangle \in L \times (\mathcal{R}_0^+)^C \mid \eta \models Inv(l)\}$
- $S_0 = \{\langle \ell_0, \eta \rangle \mid \ell_0 \in L_0 \;\wedge\; \eta\, x = 0 \text{ for all } x \in C\}$
- $N = Act \cup \mathcal{R}_0^+$ (ie, transitions can be labelled by actions or delays)
- $T \subseteq S \times N \times S$ is given by:

$$\langle l, \eta \rangle \xrightarrow{a} \langle l', \eta' \rangle \quad \Leftarrow \quad \exists_{l \xrightarrow{g,a,U} l' \in Tr} \;\; \eta \models g \;\wedge\; \eta' = \eta[U] \;\wedge\; \eta' \models Inv(l')$$

$$\langle l, \eta \rangle \xrightarrow{d} \langle l, \eta + d \rangle \quad \Leftarrow \quad \exists_{d \in \mathcal{R}_0^+} \;\; \eta + d \models Inv(l)$$

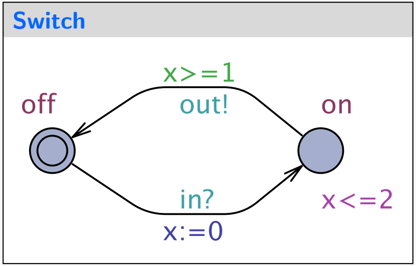
# Example: the simple switch




**Ex. 3.4: Define $\mathcal{T}$(SwitchA)**

$$S =$$

# Example: the simple switch

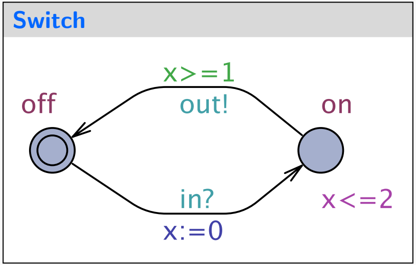



**Ex. 3.4: Define $\mathcal{T}$(SwitchA)**

$$S = \{\langle off, \bar{t}\rangle \mid t \in \mathcal{R}_0^+\} \cup \{\langle on, \bar{t}\rangle \mid 0 \leq t \leq 2\}$$

where $\bar{t}$ is a shorthand for $\eta$ such that $\eta\, x = t$

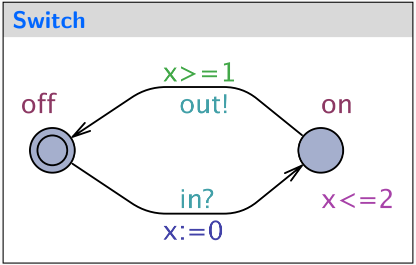
# Example: the simple switch




**Ex. 3.4: Define $\mathcal{T}$(SwitchA)**

$$T = \dots$$

# Example: the simple switch




**Ex. 3.4: Define $\mathcal{T}(\text{SwitchA})$**

$$\langle off, \overline{t}\rangle \xrightarrow{d} \langle off, \overline{t+d}\rangle \quad \text{for all } t, d \geq 0$$

$$\langle off, \overline{t}\rangle \xrightarrow{in} \langle on, \overline{0}\rangle \quad \text{for all } t \geq 0$$

$$\langle on, \overline{t}\rangle \xrightarrow{d} \langle on, \overline{t+d}\rangle \quad \text{for all } t, d \geq 0 \text{ and } t+d \leq 2$$

$$\langle on, \overline{t}\rangle \xrightarrow{out} \langle off, \overline{t}\rangle \quad \text{for all } 1 \leq t \leq 2$$

# Behavioural Equivalence

# Traces

**Definition**

A timed trace over a timed LTS is a (finite or infinite) sequence $\langle t_1, a_1 \rangle, \langle t_2, a_2 \rangle, \cdots$ in $\mathcal{R}_0^+ \times Act$ such that there exists a path

$$\langle \ell_0, \eta_0 \rangle \xrightarrow{d_1} \langle \ell_0, \eta_1 \rangle \xrightarrow{a_1} \langle \ell_1, \eta_2 \rangle \xrightarrow{d_2} \langle \ell_1, \eta_3 \rangle \xrightarrow{a_2} \cdots$$

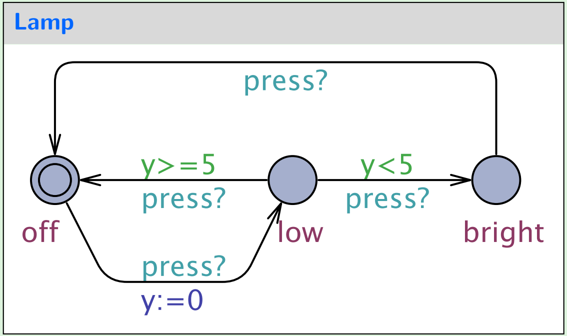
such that

$$t_i = t_{i-1} + d_i$$

with $t_0 = 0$ and, for all clock $x$, $\eta_0\, x = 0$.

Intuitively, each $t_i$ is an absolute time value acting as a time-stamp.

## Ex. 3.5: Write 4 possible timed traces

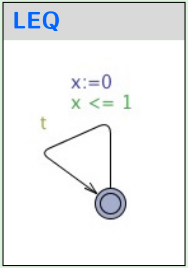


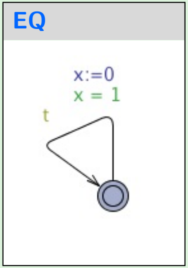
Given a timed trace *tc*, the corresponding untimed trace is $(\pi_2)^\omega\ tc$.

## Definition

- two states $s_1$ and $s_2$ of a timed LTS are timed-language equivalent if the set of finite timed traces of $s_1$ and $s_2$ coincide;
- ... similar definition for untimed-language equivalent ...

**Ex. 3.6: Why?**




are not timed-language equivalent

**Timed bisimulation (between states of timed LTS)**

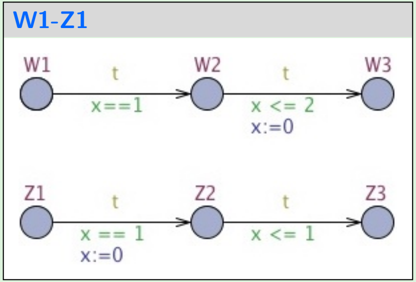
A relation $R$ is a timed simulation iff whenever $s_1 R s_2$, for any action $a$ and delay $d$,

$$s_1 \xrightarrow{a} s_1' \Rightarrow \text{there is a transition} \quad s_2 \xrightarrow{a} s_2' \wedge s_1' R s_2'$$

$$s_1 \xrightarrow{d} s_1' \Rightarrow \text{there is a transition} \quad s_2 \xrightarrow{d} s_2' \wedge s_1' R s_2'$$

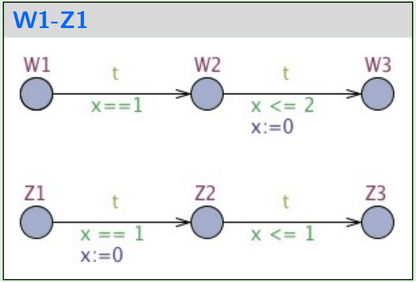
And a timed bisimulation if its converse is also a timed simulation.

## Example




W1 bisimilar to Z1?

## Example




W1 bisimilar to Z1?

$$\langle\langle W1, \{x \mapsto 0\}\rangle, \langle Z1, \{x \mapsto 0\}\rangle\rangle \in R$$

where

$$\begin{aligned} R \; = \quad & \{\langle\langle W1, \{x \mapsto d\}\rangle \;, \langle Z1, \{x \mapsto d\}\rangle\rangle \;\mid d \in \mathcal{R}_0^+\} \;\cup \\ & \{\langle\langle W2, \{x \mapsto d+1\}\rangle \;, \langle Z2, \{x \mapsto d\}\rangle\rangle \;\mid d \in \mathcal{R}_0^+\} \;\cup \\ & \{\langle\langle W3, \{x \mapsto d\}\rangle \;, \langle Z3, \{x \mapsto e\}\rangle\rangle \;\mid d, e \in \mathcal{R}_0^+\} \end{aligned}$$

# Untimed Bisimulation

**Untimed bisimulation**

A relation $R$ is an untimed simulation iff whenever $s_1 R s_2$, for any action $a$ and delay $t$,

$$s_1 \xrightarrow{a} s_1' \Rightarrow \text{there is a transition} \quad s_2 \xrightarrow{a} s_2' \wedge s_1' R s_2'$$

$$s_1 \xrightarrow{d} s_1' \Rightarrow \text{there is a transition} \quad s_2 \xrightarrow{d'} s_2' \wedge s_1' R s_2'$$

And it is an untimed bisimulation if its converse is also an untimed simulation.

Alternatively, it can be defined over a modified LTS in which all delays are abstracted on a unique, special transition labelled by $\epsilon$.

# Zeno (if there is time)

# Note

- The elapse of time in timed automata only takes place in locations:
- ... actions take place instantaneously
- Thus, several actions may take place at a single time unit

# Behaviours

- Paths in $\mathcal{T}(ta)$ are discrete representations of continuous-time behaviours in $ta$
- ... at least they indicate the states immediately before and after the execution of an action
- However, as interval delays may be realised in uncountably many different ways, different paths may represent the same behaviour

# Behaviours

- Paths in $\mathcal{T}(ta)$ are discrete representations of continuous-time behaviours in *ta*
- ... at least they indicate the states immediately before and after the execution of an action
- However, as interval delays may be realised in uncountably many different ways, different paths may represent the same behaviour
- ... but not all paths correspond to valid (realistic) behaviours:

**undesirable paths:**

- time-convergent paths
- timelock paths
- zeno paths

# Time-convergent paths

$$\langle l, \eta \rangle \xrightarrow{d_1} \langle l, \eta + d_1 \rangle \xrightarrow{d_2} \langle l, \eta + d_1 + d_2 \rangle \xrightarrow{d_3} \langle l, \eta + d_1 + d_2 + d_3 \rangle \xrightarrow{d_4} \cdots$$

such that

$$\forall_{i \in N}.\ d_i > 0 \ \wedge \ \sum_{i \in N} d_i = d$$

ie, the infinite sequence of delays converges toward $d$

- Time-convergent path are conterintuitive; as their existence cannot be avoided, they are simply ignored in the semantics of Timed Automata
- Time-divergent paths are the ones in which time always progresses

# Time-convergent paths

**Definition**

An infinite path fragment $\rho$ is time-divergent if $\mathrm{ExecTime}(\rho) = \infty$
Otherwise is time-convergent.

where

$$\mathrm{ExecTime}(\rho) = \sum_{i=0..\infty} \mathrm{ExecTime}(\delta)$$

$$\mathrm{ExecTime}(\delta) = \begin{cases} 0 & \Leftarrow \delta \in Act \\ \delta & \Leftarrow \delta \in \mathcal{R}_0^+ \end{cases}$$

for $\rho$ a path and $\delta$ a label in $\mathcal{T}(ta)$

# Timelock paths

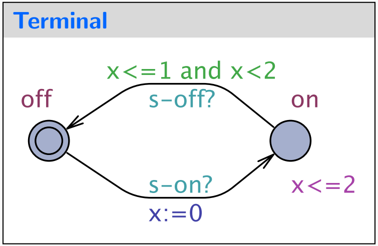
**Definition**

A path is timelock if it contains a state with a timelock, ie, a state from which there is not any time-divergent path

A timelock represents a situation that causes time progress to halt (e.g. when it is impossible to leave a location before its invariant becomes invalid)

- any teminal state ($\neq$ terminal location) in $\mathcal{T}(ta)$ contains a timelock
- ... but not all timelocks arise as terminal states in $\mathcal{T}(ta)$

# Timelock paths

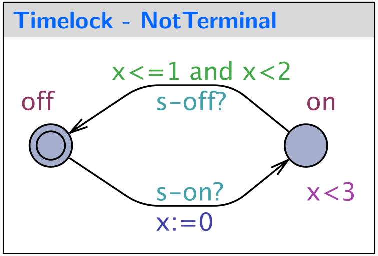



State $\langle on, 2\rangle$ is reachable through path

$$\langle off, 0\rangle \xrightarrow{s-on} \langle on, 0\rangle \xrightarrow{2} \langle on, 2\rangle$$

and is terminal

# Timelock paths




State $\langle on, 2\rangle$ is not terminal but has a convergent path:

$$\langle on, 2\rangle\langle on, 2.9\rangle\langle on, 2.99\rangle\langle on, 2.999\rangle\ldots$$

## In a Timed Automaton

- The elapse of time only takes place at locations
- Actions occur instantaneously: at a single time instant several actions may take place

... it may perform infinitely many actions in a finite time interval
(non realizable because it would require infinitely fast processors)

# Zeno

## In a Timed Automaton

- The elapse of time only takes place at locations
- Actions occur instantaneously: at a single time instant several actions may take place

... it may perform infinitely many actions in a finite time interval
(non realizable because it would require infinitely fast processors)

## Definition

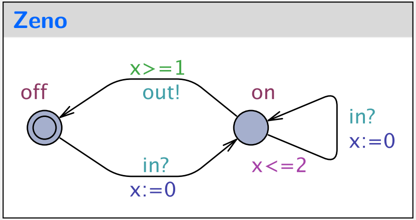
An infinite path fragment $\rho$ is zeno if it is time-convergent and infinitely many actions occur along it

A timed automaton *ta* is non-zeno if there is not an initial zeno path in $\mathcal{T}(ta)$

# Zeno

## Example

Suppose the user can press the *in* button when the light is *on* in




In doing so clock $x$ is reset to 0 and light stays *on* for more 2 time units (unless the button is pushed again ...)

# Zeno

### Example

Typical paths: The user presses *in* infinitely fast:

$$\langle \mathit{off}, 0\rangle \xrightarrow{in} \langle on, 0\rangle \xrightarrow{in} \langle on, 0\rangle \xrightarrow{in} \langle on, 0\rangle \xrightarrow{in} \langle on, 0\rangle \xrightarrow{in} \cdots$$

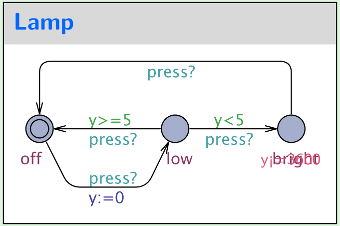
The user presses *in* faster and faster:

$$\langle \mathit{off}, 0\rangle \xrightarrow{in} \langle on, 0\rangle \xrightarrow{0.5} \langle on, 0.5\rangle \xrightarrow{in} \langle on, 0\rangle \xrightarrow{0.25} \langle on, 0.25\rangle \xrightarrow{in} \langle on, 0\rangle \xrightarrow{0.125} \cdots$$

How can this be fixed?

**Time shall pass!**

## Ex. 3.7: Recall our lamp




1. Describe a time-divergent path, if it exists.
2. Describe a time-convergent path, if it exists.
3. Describe a timelock path, if it exists.
4. Is this automata non-zeno? Justify.

### Sufficient criterion for nonzenoness

A timed automaton is nonzeno if on any of its control cycles time advances with at least some constant amount ($\geq 0$). Formally, if for every control cycle

$$\ell_0 \xrightarrow{g_0, a_0, U_0} \ell_1 \xrightarrow{g_1, a_1, U_1} \ldots \xrightarrow{g_n, a_n, U_n} \ell_0$$

there exists a clock $x \in C$ such that

1. $x \in U_i$ (for $0 \leq i \leq n$)
2. for all clock valuations $\eta$, there is a $c \in \mathbb{N}_{>0}$ such that

$$\eta(x) < c \;\Rightarrow\; ((\eta \not\models g_j) \;\vee\; \neg Inv(\ell_j)) \text{ for some } 0 \leq j \leq n$$

# Warning

Both

- timelocks
- zenoness

are modelling flaws and need to be avoided.

**Example**

In the example above, it is enough to impose a non zero minimal delay between successive button pushings.