# Timed Automata

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

# Last lecture

Visited syntax and semantics

Analysed central ideas in concurrency and synchronisation

# Last lecture

Visited syntax and semantics

Analysed central ideas in concurrency and synchronisation

We will now see how time fits in

# Table of Contents

# Motivation

Saying that an airbag in a car crash eventually inflates is insufficient – it would be better to say that …

… in a car crash the airbag 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

**Traffic lights**

Lights activate at specific time intervals

**(Re)transmission protocols**

Communication of large files between a remote unit and a video/audio equipment

**Many others**

Pacemakers, autonomous driving, electric grids ...

# This chapter

We will explore an automaton-based formalism with an explicit notion of a clock

Timed Automata [Alur & Dill, 90]

Associated tool

- UPPAAL [Behrmann, David, Larsen, 04]

# Uppaal

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

- Toolbox for modelling and analysis of timed systems
- Systems modelled as networks of timed automata with channel synchronisations
- Properties specified in a temporal logic

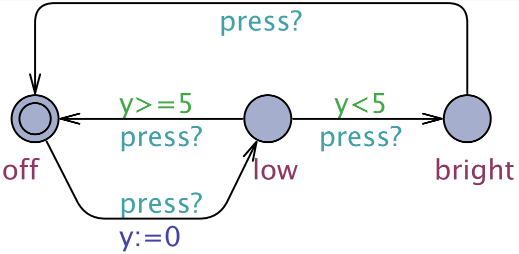
# Table of Contents

# Timed automata

Finite-state machines equipped with real-valued clocks

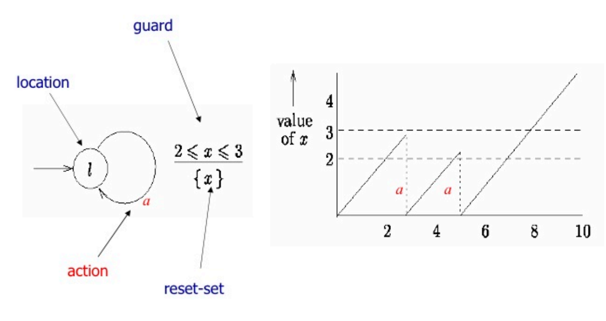
- clocks can only be read or
- reset to zero (after which they start increasing their value again as time progresses)
- a clock's value corresponds to time elapsed since its last reset
- all clocks proceed synchronously (*i.e.* at the same rate)

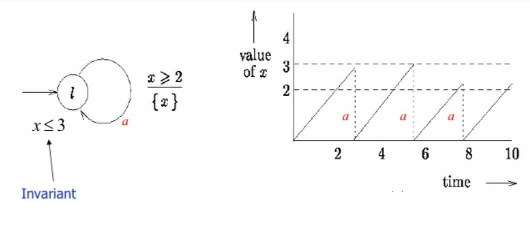
# Example: the annoying lamp




(extracted from UPPAAL)

# Guards, updates, and invariants



# Timed automata

A timed automaton is a tuple $\langle \mathsf{L}, \mathsf{L}_0, \mathsf{Act}, \mathsf{C}, \mathsf{Tr}, \mathsf{Inv} \rangle$

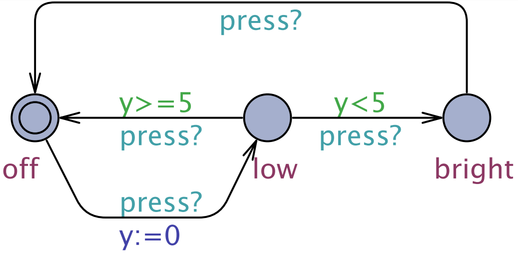
- $\mathsf{L}$ a set of locations and $\mathsf{L}_0 \subseteq \mathsf{L}$ set of initial locations
- $\mathsf{Act}$ set of actions (channels) and $\mathsf{C}$ set of clocks
- $\mathsf{Tr} \subseteq \mathsf{L} \times \mathcal{C}(\mathsf{C}) \times \mathsf{Act} \times \mathcal{P}(\mathsf{C}) \times \mathsf{L}$ is a transition relation

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

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

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

# A revisit of the annoying lamp




Exercise: define $\langle L, L_0, Act, C, Tr, Inv \rangle$

# Table of Contents

# Parallel composition of timed automata

Communication mechanism analogous to CCS

... but requires extra machinery

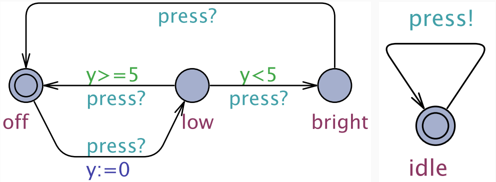
# Definition

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

$$ta_1 \parallel_H ta_2 := \langle L_1 \times L_2, L_{0,1} \times L_{0,2}, \textsf{Act}, C_1 \cup C_2, \textsf{Tr}, \textsf{Inv} \rangle$$

- $\textsf{Act} = ((Act_1 \cup Act_2) - H) \cup \{\tau\}$
- $\textsf{Inv}(\ell_1, \ell_2) = Inv_1(\ell_1) \wedge Inv_2(\ell_2)$
- $\textsf{Tr}$ is given by:
  - $(\ell_1, \ell_2) \xrightarrow{g,a,U} (\ell_1', \ell_2)$ if $a \notin H \wedge \ell_1 \xrightarrow{g,a,U} \ell_1'$
  - $(\ell_1, \ell_2) \xrightarrow{g,a,U} (\ell_1, \ell_2')$ if $a \notin H \wedge \ell_2 \xrightarrow{g,a,U} \ell_2'$
  - $(\ell_1, \ell_2) \xrightarrow{g,\tau,U} (\ell_1', \ell_2')$ if $a \in H \wedge \ell_1 \xrightarrow{g_1,a,U_1} \ell_1' \wedge \ell_2 \xrightarrow{g_2,\overline{a},U_2} \ell_2'$
    with $g = g_1 \wedge g_2$ and $U = U_1 \cup U_2$

# Example: (re)revisiting the lamp interrupt



Write down the parallel composition of the following automata