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 the items above

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

(Network-based) 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

$$\boxed{\text{Timed Automata [Alur \& Dill, 90]}}$$

Associated tool

- UPPAAL [Behrmann, David, Larsen, 04]

Uppaal

$\boxed{\text{UPPAAL} = (\text{Uppsala University} + \text{Aalborg University})\ [1995]}$

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

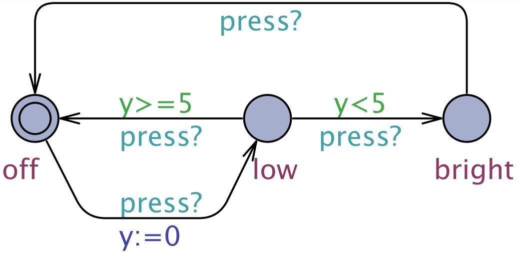
Table of Contents

Timed automata

Finite-state machines equipped with real-valued clocks

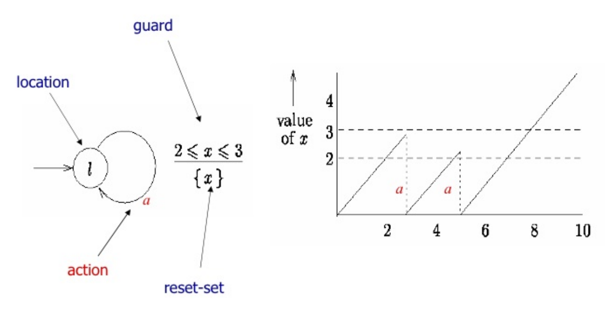
- clocks can only be read or
- reset to zero (after which they start increasing their value again as time progresses)
- a clock's value corresponds to time elapsed since its last reset
- all clocks proceed synchronously (*i.e.* at the same rate)

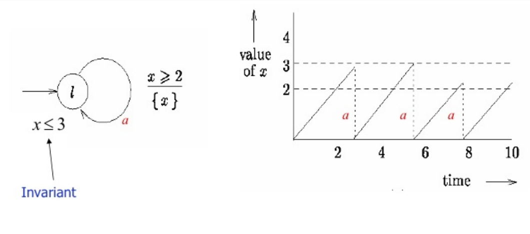
Example: the annoying lamp



(extracted from UPPAAL)

Guards, updates, and invariants





Timed automata

A timed automaton is a tuple $\langle \mathbf{L}, \mathbf{L_0}, \mathbf{Act}, \mathbf{C}, \mathbf{Tr}, \mathbf{Inv} \rangle$

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

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

  transition from location ℓ_1 to ℓ_2, labelled by a, enabled if guard g holds; when performed resets the set U of clocks
- $\mathbf{Inv} : \mathbf{L} \longrightarrow \mathcal{C}(\mathbf{C})$ assignment of invariants to locations

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

Clock constraints

Each constraint is formed according to

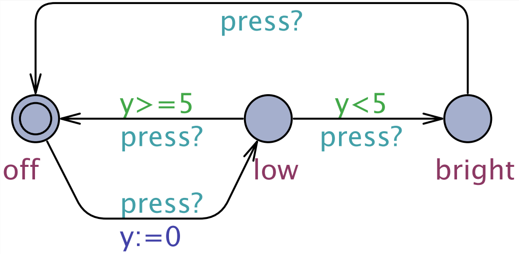
$$g \ ::= \ x \,\square\, n \ \mid \ x - y \,\square\, n \ \mid \ g \wedge g \ \mid \ \mathit{true}$$

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

This is used in

- transitions (enabling conditions) – a transition cannot occur if its guard is false
- locations (safety conditions) – a location must be left before its invariant becomes false

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

Action labels as channels

Communication mechanism analogous to CCS

Shared clocks

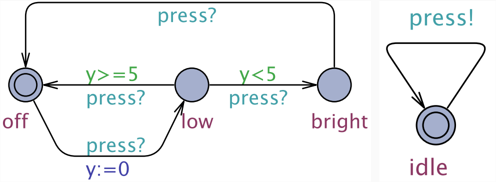
Definition

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

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

- $\mathbf{Act} = ((Act_1 \cup Act_2) - H) \cup \{\tau\}$
- $\mathbf{Inv}(\ell_1, \ell_2) = Inv_1(\ell_1) \wedge Inv_2(\ell_2)$
- $\mathbf{Tr}$ 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



In Uppaal

- Complementary actions marked by ? and ! annotations

Exercise: worker, hammer, nail

Write down the parallel composition of the following automata

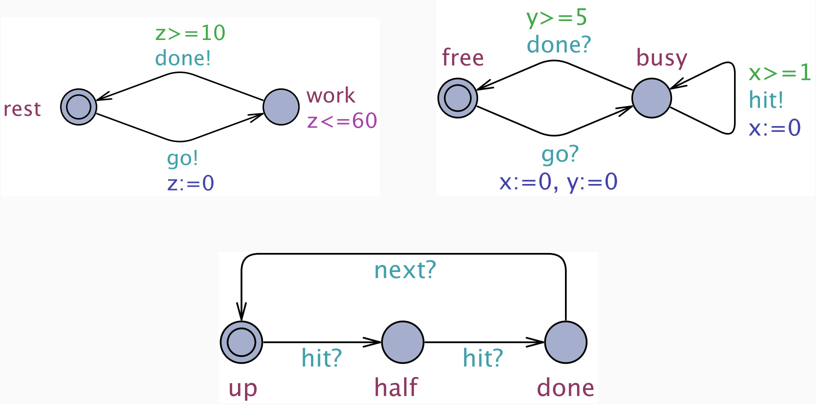


Table of Contents

Semantics of timed automata

Syntax	Semantics
How to write	*How to execute*
CCS	LTS
Timed Automaton	TLTS (Timed LTS)

Timed LTS

Introduce delay transitions to capture the passage of time

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

$s \xrightarrow{d} s'$ for $d \in \mathbb{R}_{\geq 0}$, are delay transitions

subject to sanity constraints ...

Timed LTS pt. II

1. Time additivity

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

2. Delay transitions are deterministic

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

The semantics

Every TA *ta* defines a TLTS

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

whose states are pairs

$$\langle \text{location}, \text{clocks valuations} \rangle$$

Clock valuation

Definition

A clock valuation η for a set of clocks C is a function

$$\eta : C \longrightarrow \mathbb{R}_{\geq 0}$$

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

Satisfaction of Clock Constraints

$$\eta \models x \,\square\, n \;\Leftrightarrow\; \eta\, x \,\square\, n$$

$$\eta \models x - y \,\square\, n \;\Leftrightarrow\; (\eta\, x - \eta\, y) \,\square\, n$$

$$\eta \models g_1 \wedge g_2 \;\Leftrightarrow\; \eta \models g_1 \wedge \eta \models g_2$$

Operations on clock valuations

Delay

For each $d \in \mathbb{R}_{\geq 0}$, valuation $\eta + d$ is given by

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

Reset

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

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

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

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

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

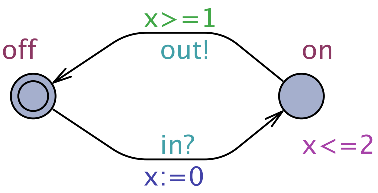
where

- $\mathbf{S} = \{(l,\eta) \in L \times (\mathbb{R}_{\geq 0})^C \mid \eta \models Inv(l)\}$
- $\mathbf{S_0} = \{(\ell_0,\eta) \mid \ell_0 \in L_0 \;\wedge\; \eta\, x = 0 \text{ for all } x \in C\}$
- $\mathbf{N} = Act + \mathbb{R}_{\geq 0}$ (i.e., transitions can be labelled by actions or delays)
- $\mathbf{T} \subseteq S \times N \times S$ is given by

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

$$(l,\eta) \xrightarrow{d} (l,\eta + d) \quad \text{if} \quad \eta + d \models Inv(l)$$

Example: the simple switch pt. I

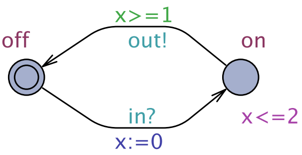


$\mathcal{T}$(Simple switch)

$$S = \{(\mathit{off}, t) \mid t \in \mathbb{R}_{\geq 0}\} \cup \{(\mathit{on}, t) \mid 0 \leq t \leq 2\}$$

where t is a shorthand for η such that $\eta\, x = t$

Example: the simple switch pt. II



$\mathcal{T}$(Simple switch)

$$(\mathit{off}, t) \xrightarrow{d} (\mathit{off}, t + d) \text{ for all } t, d \geq 0$$

$$(\mathit{off}, t) \xrightarrow{in?} (on, 0) \text{ for all } t \geq 0$$

$$(on, t) \xrightarrow{d} (on, t + d) \text{ for all } t, d \geq 0 \text{ and } t + d \leq 2$$

$$(on, t) \xrightarrow{out!} (\mathit{off}, t) \text{ for all } 1 \leq t \leq 2$$