Timed Systems

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Previously . . .

- We visited the concepts of syntax and semantics
- Explored a basic concurrent language and its semantics
- Analysed central ideas of Concurrency & Synchronisation
- Saw that (some) semantic aspects arise from functors, which provides basis for a uniform development of semantics

Previously . . .

- We visited the concepts of syntax and semantics
- Explored a basic concurrent language and its semantics
- Analysed central ideas of Concurrency & Synchronisation
- Saw that (some) semantic aspects arise from functors, which provides basis for a uniform development of semantics

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:

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 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 synchronisation delays
- time-out values

And many others ...

- medical instruments
- cruise controllers

Our Focus

We will explore an automaton-based formalism with an explicit notion of clock (stopwatch) to control availability of transitions

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

Emphasis on the reachability problem and corresponding practically efficient algorithms

Associated tool

- UPPAAL [Behrmann, David, Larsen, 04]

Uppaal

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

- A toolbox for modelling and analysis of timed systems
- Systems modelled as networks of timed automata enriched with integer variables and channel synchronisations
- Properties specified in a subset of CTL

Computation Tree Logic

https://uppaal.org/

Table of Contents

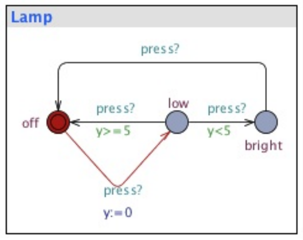
Timed automata

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 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 Lamp Interrupt



(extracted from UPPAAL)

Timed Automata

Definition

A timed automaton is a tuple $\langle L, L_0, Act, C, Tr, Inv\rangle$ such that

- 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

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

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

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

Clock Constraints

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

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

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

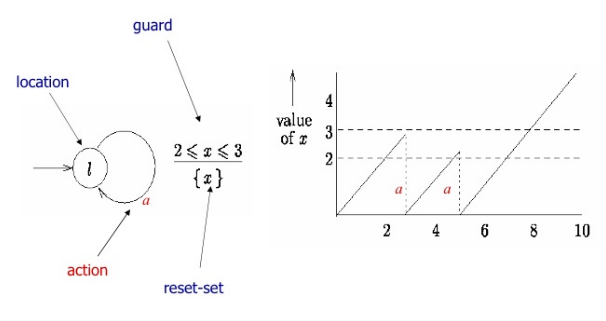
This is used in

- 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

Note

Invariants are the only way to force transitions to occur

Guards, Updates, & Invariants



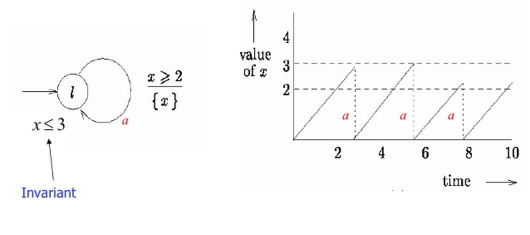


Table of Contents

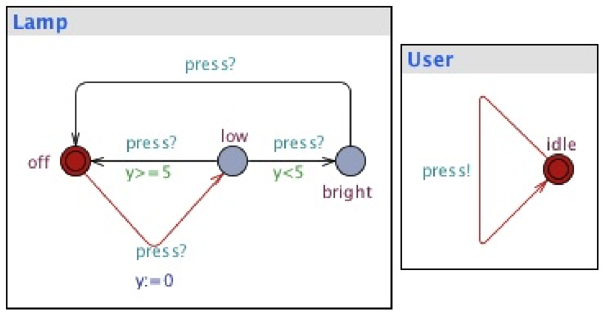
Parallel Composition of Timed Automata

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

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

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

Example: revisiting the Lamp Interrupt



Uppaal:

- communication occurs between complementary actions (marked by ? and ! annotations)
- only considers closed systems

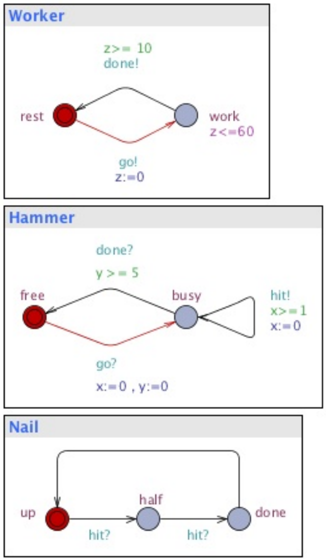

Worker
z>= 10
done!
rest
work
z<=60
go!
z:=0
Hammer
done?
y >= 5
free
busy
hit!
x>=1
x:=0
go?
x:=0 , y:=0
Nail
up
half
done
hit?
hit?

Table of Contents

Timed Labelled Transition Systems

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

Timed Labelled Transition Systems

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

Timed LTS pt. I

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

$s \xrightarrow{a} s'$ 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 a number of constraints

Timed LTS pt. II

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 Timed Automata

Every TA *ta* defines a TLTS

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

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

$$\begin{aligned}
\eta \models x \,\square\, n \;&\Leftrightarrow\; \eta\, x \,\square\, n \\
\eta \models x - y \,\square\, n \;&\Leftrightarrow\; (\eta\, x - \eta\, y) \,\square\, n \\
\eta \models g_1 \wedge g_2 \;&\Leftrightarrow\; \eta \models g_1 \wedge \eta \models g_2
\end{aligned}$$

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

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

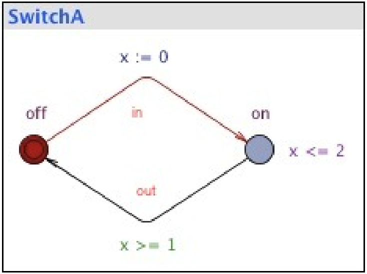
where

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

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

$$(l, \eta) \xrightarrow{d} (l, \eta + d) \quad \text{if} \quad \exists_{d \in \mathbb{R}_{\geq 0}} \;\; \eta + d \models Inv(l)$$

Example: the Simple Switch pt. I

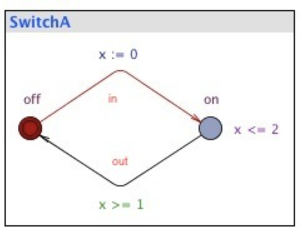


$\mathcal{T}$(SwitchA)

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

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

Example: the Simple Switch pt. II



$\mathcal{T}$(SwitchA)

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

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

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

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

Table of Contents

Execution Traces

Definition

A trace over a timed LTS is a (finite or infinite) sequence $(t_1, a_1), (t_2, a_2), \cdots$ in $\mathbb{R}_{\geq 0} \times \mathit{Act}$ such that there exist

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

respecting the equation

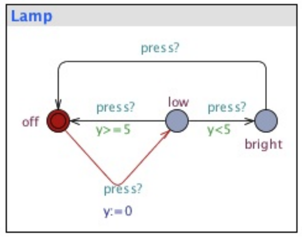
$$t_i = t_{i-1} + d_i$$

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

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

Exercise

Write possible traces



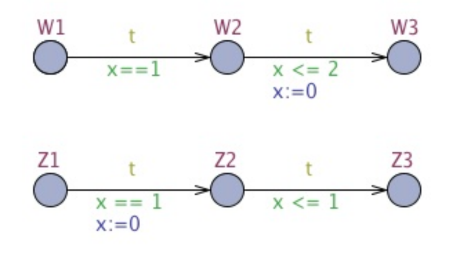
Observational Equivalence

Two states s_1 and s_2 of timed LTSs are equivalent iff for any action a and delay d

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

and *vice-versa*

Exercise



W1 equivalent to Z1?