Real-Time Systems

Lecture 02: Timed Behaviour

2014-05-06

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

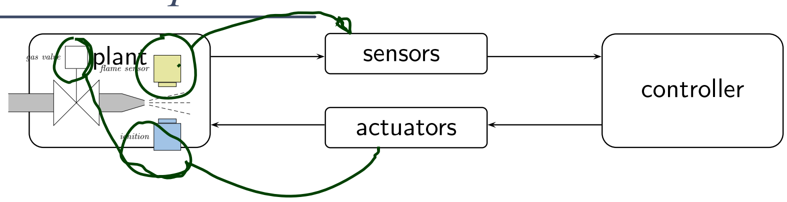
Last Lecture:

- Motivation, Overview

This Lecture:

- **Educational Objectives:**
  - Get acquainted with one (simple but powerful) formal model of timed behaviour.
  - See how first order predicate-logic can be used to state requirements.
- **Content:**
  - Time-dependent State Variables
  - Requirements and System Properities in first order predicate logic
  - Classes of Timed Properties

Recall: Prerequisites



To

design a (gas burner) controller that meets its requirements

we need

- a formal model of behaviour in quantitative time
- a language to concisely and conveniently specify requirements
- a language to describe controller behaviour
- a notion of "meet" — and a method to verify meeting

Real-Time Behaviour, More Formally...

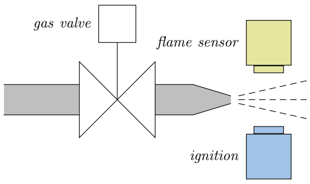
State Variables (or Observables)

- We assume that the real-time systems we consider is characterised by a finite set of **state variables** (or **observables**)

$$obs_1, \ldots, obs_n$$

each equipped with a **domain** $\mathcal{D}(obs_i)$, $1 \le i \le n$.

- **Example**: gas burner



- $G : \{0, 1\}$ — 0 iff valve closed
- $F : \{0, 1\}$ — 0 iff no flame
- $I : \{0, 1\}$ — 0 iff ignition off
- $H : \{0, 1\}$ — 0 iff no heating request

System Evolution over Time

- **One** possible evolution (or **behaviour**) of the considered system over time is represented as a function

$$\pi : \mathsf{Time} \to \mathcal{D}(obs_1) \times \cdots \times \mathcal{D}(obs_n).$$

- If (and only if) observable obs_i has value $d_i \in \mathcal{D}(obs_i)$ at time $t \in \mathsf{Time}$, $1 \le i \le n$, we set

$$\pi(t) = (d_1, \ldots, d_n).$$

- For convenience, we use

$$obs_i : \mathsf{Time} \to \mathcal{D}(obs_i)$$

to denote the projection of π onto the i-th component.

What's the time?

- There are two main choices for the time domain Time:
  - **discrete time**: Time $= \mathbb{N}_0$, the set of natural numbers.
  - **continuous or dense time**: Time $= \mathbb{R}_0^+$, the set of non-negative real numbers.

- Throughout the lecture we shall use the **continuous** time model and consider **discrete** time as a special case.

  Because
  - plant models usually live in **continuous** time,
  - we avoid too early introduction introduction of hardware considerations,

- Interesting view: continous-time is a well-suited **abstraction** from the discrete-time realms induced by clock-cycles etc.

Example: Gas Burner

One possible evolution of considered system over time is represented as function

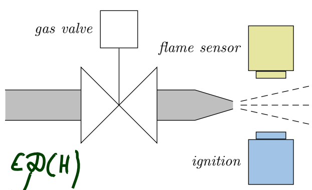
$$\pi : \text{Time} \to \mathcal{D}(obs_1) \times \cdots \times \mathcal{D}(obs_n)$$

with

$$\pi(t) = (d_1, \ldots, d_n)$$

if (and only if) observable obs_i has value $d_i \in \mathcal{D}(obs_i)$ at time $t \in$ Time.

For convenience: use obs_i : Time $\to \mathcal{D}(obs_i)$.

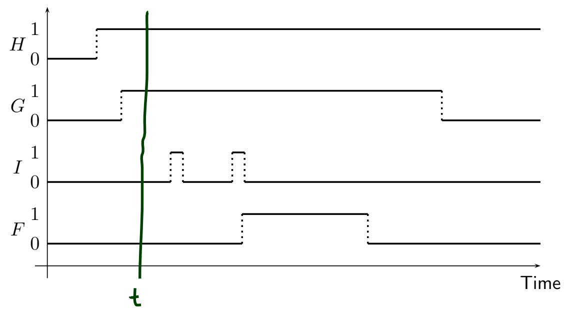


$\in \mathcal{D}(H)$

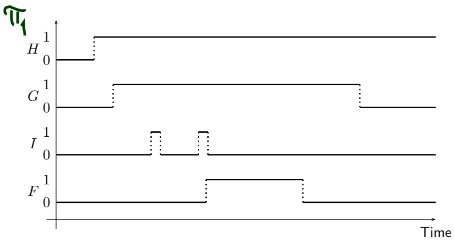
$\pi(t) = (1, 1, 0, 0)$

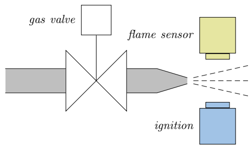
$I(t) = 0$

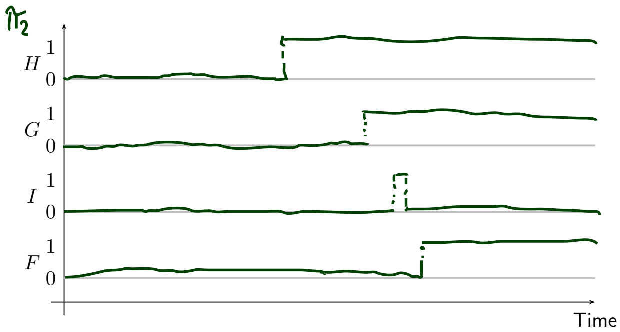
π :



Example: Gas Burner







Levels of Detail

Note:
Depending on the **choice of observables** we can describe a real-time system at various **levels of detail**.

For instance,

- if the gas valve has different positions, use

$$G : \mathsf{Time} \to \{0, 1, 2, 3\}$$

  ($\mathcal{D}(G)$ is never continuous in the lecture, otherwise it's a hybrid system!)

- if the thermostat and the controller are connected via a bus and exchange messages, use

$$B : \mathsf{Time} \to Msg^*$$

  to model the receive buffer as a finite sequence of messages from Msg.
- etc.

System Properties: A First Approach

Predicate Logic

$$\varphi ::= obs(t) = d \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \implies \varphi_2 \mid \varphi_1 \iff \varphi_2$$
$$\mid \forall\, t \in \mathsf{Time} \bullet \varphi \mid \forall\, t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi$$

obs an observable, $d \in \mathcal{D}(obs)$, $t \in \mathsf{Var}$ logical variable, $c_1, c_2 \in \mathbb{R}_0^+$ constants.

We assume the **standard semantics** interpreted over system evolutions

$$obs_i : \mathsf{Time} \to \mathcal{D}(obs), 1 \leq i \leq n.$$

That is, given a particular system evolution π and a formula φ, we can tell whether π satisfies φ under a given valuation β, denoted by $\pi, \beta \models \varphi$.

Recall: Predicate Logic, Standard Semantics

Evolution of system over time:	$\pi : \mathsf{Time} \to \mathcal{D}(obs_1) \times \cdots \times \mathcal{D}(obs_n)$.
Iff obs_i has value $d_i \in \mathcal{D}(obs_i)$ at $t \in \mathsf{Time}$, set:	$\pi(t) = (d_1, \ldots, d_n)$.
For convenience: use	$obs_i : \mathsf{Time} \to \mathcal{D}(obs_i)$.

$$\varphi ::= obs(t) = d \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \implies \varphi_2 \mid \varphi_1 \iff \varphi_2$$
$$\mid \forall\, t \in \mathsf{Time} \bullet \varphi \mid \forall\, t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi$$

- Let $\beta : \mathsf{Var} \to \mathsf{Time}$ be a **valuation** of the logical variables.

- $\pi, \beta \models obs_i(t) = d$ iff $obs_i(\beta(t)) = d$
- $\pi, \beta \models \neg\varphi$ iff not $\pi, \beta \models \varphi$
- $\pi, \beta \models \varphi_1 \vee \varphi_2$ iff ...
- ...
- $\pi, \beta \models \forall\, t \in \mathsf{Time} \bullet \varphi$ iff for all $t_0 \in \mathsf{Time}$, $\pi, \beta[t \mapsto t_0] \models \varphi$
- $\pi, \beta \models \forall\, t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi$ iff
  for all $t_0 \in [\beta(t_1) + c_1, \beta(t_2) + c_2]$, $\pi, \beta[t \mapsto t_0] \models \varphi$

Predicate Logic

all logical variables appear quantified

Note: we can view a closed predicate logic formula φ as a **concise description** of

$$\{\pi : \mathsf{Time} \to \mathcal{D}(obs_1) \times \cdots \times \mathcal{D}(obs_n) \mid \pi, \emptyset \models \varphi\},$$

the set of all system evolutions satisfying φ.

For example,

$$\forall\, t \in \mathsf{Time} \bullet \neg(I(t) \wedge \neg G(t))$$

describes all evolutions where there is no ignition with closed gas valve.

Requirements and System Properties

- So we can use first-order predicate logic to formally specify requirements.

  A **requirement** 'Req' is a set of system behaviours with the pragmatics that, whatever the behaviours of the final **implementation** are, they shall lie within this set.

  For instance,

$$\mathsf{Req} \;:\!\iff\; \forall\, t \in \mathsf{Time} \bullet \neg(I(t) \wedge \neg G(t))$$

  says: "an implementation is fine as long as it doesn't ignite without gas in any of its evolutions".

- We can also use first-order predicate logic to formally describe properties of the **implementation** or **design decisions**.

  For instance,

$$\mathsf{Des} \;:\!\iff\; \forall\, t \in \mathsf{Time} \bullet I(t) \implies \forall\, t' \in [t-1, t+1] \bullet G(t'))$$

  says that our controller opens the gas valve at least 1 time unit before ignition and keeps it open.

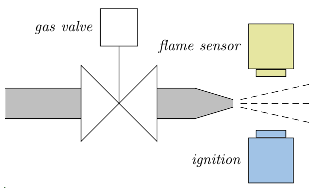
Example: Gas Burner

Req. is an abbrev. for formula

$\mathsf{Req} \;:\!\iff\; \forall\, t \in \mathsf{Time} \bullet \neg(I(t) \wedge \neg G(t))$

$\mathsf{Des} \;:\!\iff\; \forall\, t \in \mathsf{Time} \bullet$
$\quad I(t) \implies \forall\, t' \in [t-1, t+1] \bullet G(t'))$

$\pi \in \mathsf{Req}$?
$\pi \in \mathsf{Des}$?



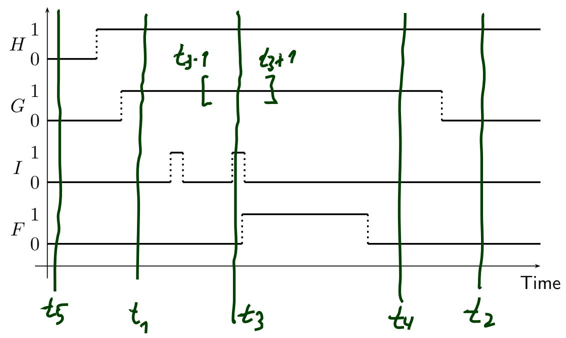
- $I(t_1) = 0$
- $G(t_1) = 1$

↳ t_1 ok

- $I(t_2) = 0$
- $G(t_2) = 0$

↳ t_2 ok

π :



- $I(t_3) = 1$
- $G(t') = 1$ for $t' \in [t_3-1, t_3+1]$

↳ ok for t_3

- $I(t_4) = 0$

↳ ok for t_4

Correctness

- Let 'Req' be a **requirement**,
- 'Des' be a **design**, and
- 'Impl' be an **implementation**.

Recall: each is a set of evolutions, i.e. a subset of $\big(\mathsf{Time} \to \times_{i=1}^{n} \mathcal{D}(obs_i)\big)$, described in any form.
We say

- 'Des' is a **correct design** (wrt. 'Req') if and only if

$$\mathsf{Des} \subseteq \mathsf{Req}.$$

- 'Impl' is a **correct implementation** (wrt. 'Des' (or 'Req')) if and only if

$$\mathsf{Impl} \subseteq \mathsf{Des} \quad (\text{or } \mathsf{Impl} \subseteq \mathsf{Req})$$

If 'Req' and 'Des' are described by formulae of first-oder predicate logic, proving the design correct amounts to proving that 'Des $\implies$ Req' is valid.

Classes of Timed Properties

Safety Properties

- A **safety property** states that

  something bad must never happen [Lamport].

- Example: train inside level crossing with gates open.

- More general, assume observable $C : \cancel{\mathsf{Time} \to} \{0, 1\}$ where $C(t) = 1$ represents a critical system state at time t. "sth. bad"

  Then

  $$\forall\, t \in \mathsf{Time} \bullet \neg C(t)$$

  is a safety property.

- In general, a safety property is characterised as a property that can be **falsified** in bounded time.

- But safety is not everything...

Liveness Properties

- The simplest form of a **liveness property** states that

  something good eventually does happen.

- Example: gates open for road traffic.

- More general, assume observable $G : \cancel{\mathsf{Time} \to} \{0, 1\}$ where $G(t) = 1$ represents a good system state at time t.

  Then

  $$\exists\, t \in \mathsf{Time} \bullet G(t)$$

  is a liveness property.

- Note: not falsified in finite time.

- With real-time, liveness is too weak...

Bounded Response Properties

- A **bounded response property** states that

  the desired reaction on an input occurs in time interval $[b, e]$.

- Example: from request to secure level crossing to gates closed.

- More general, re-consider good thing $G : \cancel{\text{Time} \to} \{0,1\}$ and request $R : \cancel{\text{Time} \to} \{0,1\}$.

  Then

  $$\forall\, t_1 \in \text{Time} \bullet (R(t_1) \implies \exists\, t_2 \in [t_1 + \overset{b}{\cancel{10}}, t_1 + \overset{e}{\cancel{15}}] \bullet G(t_2))$$

  is a bounded liveness property.

- This property can again be falsified in finite time.

- With gas burners, this is still not everything...

Duration Properties

$A(b,e) := (e-b) \geq 60$

- length bigger-equal 60 s
- at most 5% of the time leakage
- $C \leftarrow$ $G(t) \wedge \neg F(t)$

- A **duration property** states that

  for observation interval $[b, e]$ characterised by a condition $A(b, e)$
  the **accumulated time** in which the system is in a certain critical state has an upper bound $u(b, e)$.

  $u(b,e) := 0.05 \cdot (e-b)$

- Example: leakage in gas burner.

- More general, re-consider critical thing $C : \cancel{\text{Time} \to} \{0,1\}$

  Riemann-Integral

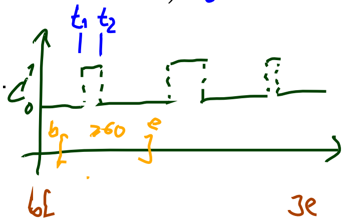
  Then

  $$\forall\, b, e \in \text{Time} \bullet \left(A(b, e) \implies \int_b^e C(t)\, dt \leq u(b, e) \right)$$

  $\int_b^e C(t)\,dt = (t_2 - t_1)$

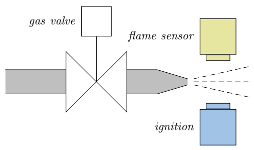
  is a duration property.

- This property can again be falsified in finite time.



Duration Calculus

Duration Calculus: Preview



- $G, F, I, H : \{0, 1\}$
- Define $L : \{0, 1\}$ as $G \wedge \neg F$.

- Duration Calculus is an **interval logic**.
- Formulae are evaluated in an (**implicitly given**) interval.

almost

Strangest operators:

- **everywhere** — Example: $\lceil G \rceil$

  (Holds in a given interval $[b, e]$ iff the gas valve is open almost everywhere.)

- **chop** — Example: $(\lceil \neg I \rceil \,;\, \lceil I \rceil \,;\, \lceil \neg I \rceil) \implies \ell \geq 1$

  (Ignition phases last at least one time unit.)

- **integral** — Example: $\ell \geq 60 \implies \int L \leq \frac{\ell}{20}$

  (At most 5% leakage time within intervals of at least 60 time units.)

Duration Calculus: Overview

We will introduce three (or five) syntactical "levels":

(i) **Symbols:**

$$f, g, \quad \mathit{true}, \mathit{false}, =, <, >, \leq, \geq, \quad x, y, z, \quad X, Y, Z, \quad d$$

(ii) **State Assertions:**

$$P ::= 0 \mid 1 \mid X = d \mid \neg P_1 \mid P_1 \wedge P_2$$

evaluate to 0,1

(iii) **Terms:**

$$\theta ::= x \mid \ell \mid \int P \mid f(\theta_1, \dots, \theta_n)$$

yield $\mathbb{R}$

(iv) **Formulae:**

$$F ::= p(\theta_1, \dots, \theta_n) \mid \neg F_1 \mid F_1 \wedge F_2 \mid \forall x \bullet F_1 \mid F_1 \,;\, F_2$$

yield tt, ff

(v) **Abbreviations:**

$$\lceil\,\rceil, \quad \lceil P \rceil, \quad \lceil P \rceil^t, \quad \lceil P \rceil^{\leq t}, \quad \Diamond F, \quad \Box F$$

Symbols: Syntax

- f, g: **function symbols**, each with arity $n \in \mathbb{N}_0$.
  Called **constant** if $n = 0$.
  Assume: constants $0, 1, \dots \in \mathbb{N}_0$; binary '$+$' and '$\cdot$'.
- p, q: **predicate symbols**, also with arity.
  Assume: constants $\mathit{true}, \mathit{false}$; binary $=, <, >, \leq, \geq$.
- $x, y, z \in$ GVar: **global variables**.
- $X, Y, Z \in$ Obs: **state variables** or **observables**, each of a data type $\mathcal{D}$ (or $\mathcal{D}(X), \mathcal{D}(Y), \mathcal{D}(Z)$ to be precise).
  Called **boolean observable** if data type is $\{0, 1\}$.
- d: **elements** taken from data types $\mathcal{D}$ of observables.

Symbols: Semantics

- **Semantical domains** are
  - the **truth values** $\mathbb{B} = \{\mathrm{tt}, \mathrm{ff}\}$,
  - the **real numbers** $\mathbb{R}$,
  - **time** Time,
    (mostly Time $= \mathbb{R}_0^+$ (continuous), exception Time $= \mathbb{N}_0$ (discrete time))
  - and **data types** $\mathcal{D}$.

- The semantics of an n-ary **function symbol** f
  is a (mathematical) function from $\mathbb{R}^n$ to $\mathbb{R}$, denoted $\hat{f}$, i.e.

$$\hat{f} : \mathbb{R}^n \to \mathbb{R}.$$

- The semantics of an n-ary **predicate symbol** p
  is a function from $\mathbb{R}^n$ to $\mathbb{B}$, denoted $\hat{p}$, i.e.

$$\hat{p} : \mathbb{R}^n \to \mathbb{B}.$$

Symbols: Examples

- The **semantics** of the function and predicate symbols **assumed above**
  is fixed throughout the lecture:

  - $\hat{true} = \mathrm{tt}$, $\hat{false} = \mathrm{ff}$
  - $\hat{0} \in \mathbb{R}$ is the (real) number **zero**, etc.
  - $\hat{+} : \mathbb{R}^2 \to \mathbb{R}$ is the **addition** of real numbers, etc.
  - $\hat{=} : \mathbb{R}^2 \to \mathbb{B}$ is the **equality** relation on real numbers,
  - $\hat{<} : \mathbb{R}^2 \to \mathbb{B}$ is the **less-than** relation on real numbers, etc.

- "Since the semantics is the expected one, we shall often simply use the
  symbols $0, 1, +, \cdot, =, <$ when we mean their semantics $\hat{0}, \hat{1}, \hat{+}, \hat{\cdot}, \hat{=}, \hat{<}$."

Symbols: Semantics

- The semantics of a **global variable** is not fixed (throughout the lecture) but given by a **valuation**, i.e. a mapping

$$\mathcal{V} : \mathsf{GVar} \to \mathbb{R}$$

assigning each global variable $x \in \mathsf{GVar}$ a real number $\mathcal{V}(x) \in \mathbb{R}$.

We use Val to denote the set of all valuations, i.e. $\mathsf{Val} = (\mathsf{GVar} \to \mathbb{R})$.

Global variables are though **fixed over time** in system evolutions.

Symbols: Semantics

- The semantics of a **global variable** is not fixed (throughout the lecture) but given by a **valuation**, i.e. a mapping

$$\mathcal{V} : \mathsf{GVar} \to \mathbb{R}$$

assigning each global variable $x \in \mathsf{GVar}$ a real number $\mathcal{V}(x) \in \mathbb{R}$.

We use Val to denote the set of all valuations, i.e. $\mathsf{Val} = (\mathsf{GVar} \to \mathbb{R})$.

Global variables are though **fixed over time** in system evolutions.

- The semantics of a **state variable** is **time-dependent**. It is given by an interpretation $\mathcal{I}$, i.e. a mapping

$$\mathcal{I} : \mathsf{Obs} \to (\mathsf{Time} \to \mathcal{D})$$

assigning each state variable $X \in \mathsf{Obs}$ a function

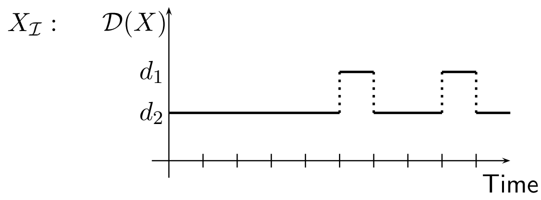
$$\mathcal{I}(X) : \mathsf{Time} \to \mathcal{D}(X)$$

such that $\mathcal{I}(X)(t) \in \mathcal{D}(X)$ denotes the value that X has at time $t \in \mathsf{Time}$.

Symbols: Representing State Variables

- For convenience, we shall abbreviate $\mathcal{I}(X)$ to $X_\mathcal{I}$.
- An **interpretation** (of a state variable) can be displayed in form of a **timing diagram**.

  For instance,



with $\mathcal{D}(X) = \{d_1, d_2\}$.