Real-Time Systems

Lecture 02: Timed Behaviour

2012-04-26

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

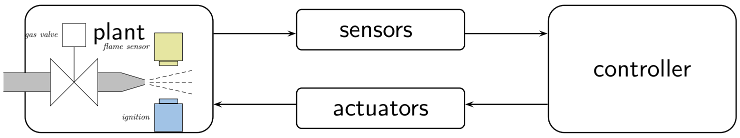
Last Lecture:

- 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, conveniently specify requirements on behaviour,
- a language to specify behaviour of controllers,
- a notion of "meet" and a methodology to verify meeting.

see Slide 14, Lecture 01

– 02 – 2012-04-26 – Srecall –

Real-Time Behaviour, More Formally...

– 02 – 2012-04-26 – main –

Recall

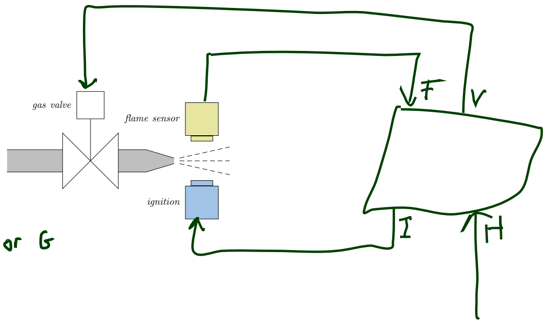
State Variables (or Observables)

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

$$obs_1, \dots, obs_n$$

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

- **Example**: gas burner



- V, $\mathcal{D}(V) = \{0, 1\}$ or G
- F, $\mathcal{D}(F) = \{0, 1\}$
- I, $\mathcal{D}(I) = \{0, 1\}$
- H, $\mathcal{D}(H) = \{0, 1\}$

– Smodel –

5/31

Recall: System Evolution over Time

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

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

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

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

- For convenience, we use

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

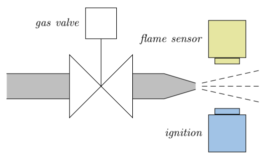
to denote the projection of π onto the i-th component.

Recall: 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



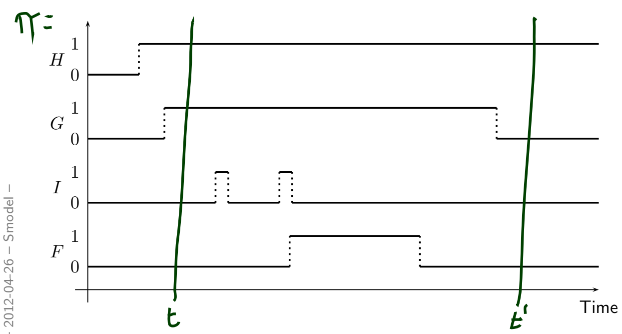
One possible evolution of considered system over time is represented as function

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

If (and only if) observable obs_i has value $d_i \in \mathcal{D}(obs_i)$ at time $t \in \mathsf{Time}$, set:

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

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



F I G H

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

$\pi(t') = |0, 0, 0, 1|$

$H(t) = 1$

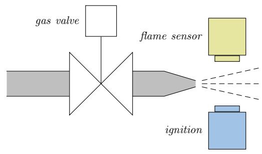
$H(t') = 1$

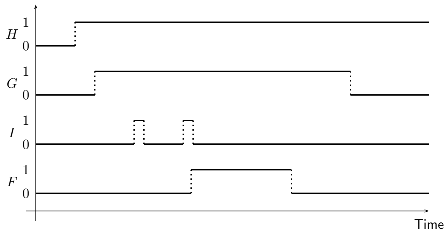
$\forall\, t \in \mathsf{Time} \bullet$

$I(t) \Rightarrow G(t) \quad \models \pi$

$I(t) \wedge G(t) \quad \not\models \pi$

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

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

    (But: $\mathcal{D}(G)$ is never continuous in the lecture, otherwise we had a hybrid system.)

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

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

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

System Properties

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, $d \in \mathcal{D}(obs)$, $t \in$ 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:	$\pi(t) = (d_1, \dots, d_n)$.
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 $obs_i(\beta(t)) = d$ or $\pi(\beta(t))(obs_i) = d$
- $\pi, \beta \models \neg\varphi$ 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 $\tilde{t} \in \mathsf{Time}$, $\pi, \beta[t \mapsto \tilde{t}] \models \varphi$
- $\pi, \beta \models \forall\, t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi$ iff for all $\tilde{t} \in [\beta(t_1)+c_1, \beta(t_2)+c_2]$, $\pi, \beta[t \mapsto \tilde{t}] \models \varphi$

We write $\pi, \beta \not\models \varphi$ iff not $\pi, \beta \models \varphi$.

– 02 – 2012-04-26 – Sprop –

Predicate Logic

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.

– 02 – 2012-04-26 – Sprop –

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,

  syntact. abbrev.

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

  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,

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

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

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 $\big(\text{Time} \to \times_{i=1}^{n} \mathcal{D}(obs_i)\big)$, described in any form.

We say

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

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

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

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

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].

Other Def.:
A finite prefix of an evolution π is sufficient for any counterexample.

- Example: train inside level crossing with gates open.

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

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

"Response": $\forall\, t \in \mathsf{Time} \bullet \overbrace{\;}^{R(t) \implies} \exists\, t' \in [t, t+\infty] \bullet G(t')$

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 $G : \mathsf{Time} \to \{0, 1\}$ and request $R : \mathsf{Time} \to \{0, 1\}$.

  Then

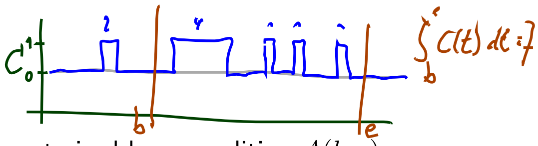
  $$\forall\, t_1 \in \mathsf{Time} \bullet (R(t_1) \implies \exists\, t_2 \in [t_1 + 10, t_1 + 15] \bullet G(t_2))$$

  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 **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 $u(b, e)$.

- Example: leakage in gas burner.

  e.g. $|b-e| \geq 60$

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

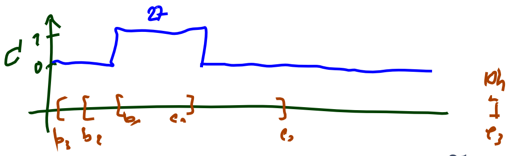
  $0{,}05 \cdot |b-e|$

  Then

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

  is a duration property.

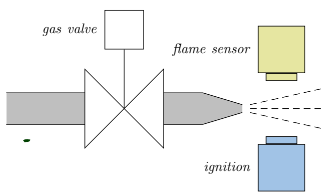
- This property can again be falsified in finite time.



Duration Calculus

Duration Calculus: Preview

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



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

Strangest operators:

- almost **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:**

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

(iii) **Terms:**

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

(iv) **Formulae:**

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

(v) **Abbreviations:**

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

References

References

[Olderog and Dierks, 2008] Olderog, E.-R. and Dierks, H. (2008). *Real-Time Systems - Formal Specification and Automatic Verification*. Cambridge University Press.