# Real-Time Systems

## Lecture 02: Timed Behaviour

2013-04-17


Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany




## Contents & Goals

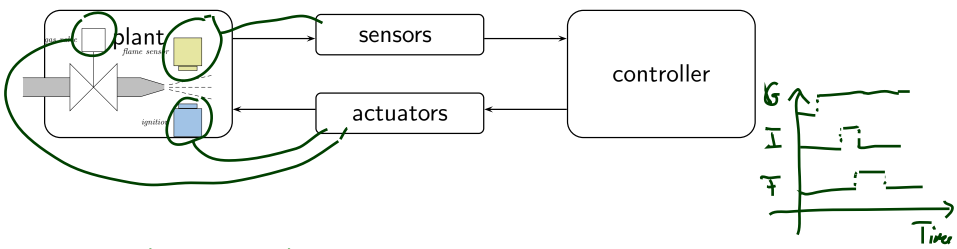
**Last Lecture:**

- 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 timed behaviour
- a language to specify behaviour of controllers
- a notion of "meet" and a methodology to verify (prove) meeting



## *Real-Time Behaviour, More Formally...*

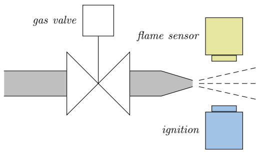
## *State Variables (or Observables)*

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

$$obs_1, \dots, obs_n$$

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

- **Example**: gas burner




- "gas valve open/closed" $G$, $\mathcal{D}(G) = \{0,1\}$, 0 iff valve closes
- "flame yes/no" $F$, $\mathcal{D}(F) = \{0,1\}$, 0 iff no flame
- "ignition going on yes/no" $I$, $\mathcal{D}(I) = \{0,1\}$, 0 iff no ignition
- "heating need yes/no" $H$, $\mathcal{D}(H) = \{0,1\}$. 0 iff no need



## *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 $t \in \mathsf{Time}$, $1 \leq i \leq n$, 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 $\pi$ 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.

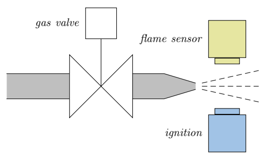
  Because
  - plant models usually live in **continuous** time,
  - we avoid too early introduction introduction of hardware considerations,

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

– 02 – 2013-04-17 – Smodel –



# Example: Gas Burner



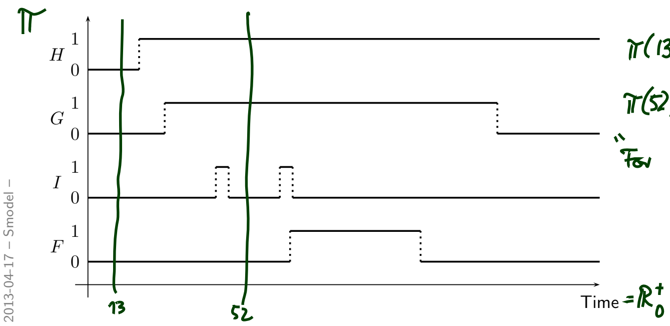

One possible evolution of considered system over time is represented as function

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

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

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

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




H G I F

$\pi(13) = (0,0,0,0)$

$\pi(52) = (1,1,0,0)$

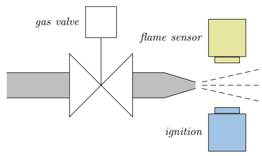
"For convenience"

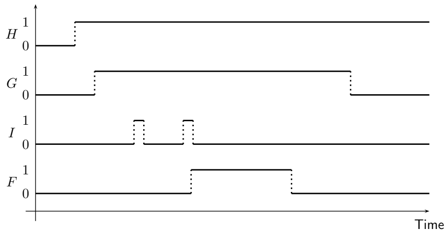
$H(13) = 0$

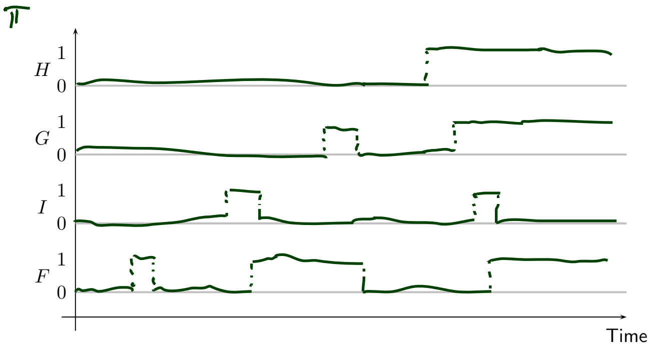
$H(52) = 1$

– 02 – 2013-04-17 – Smodel –

# 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

    $\mathcal{D}(G) = \{0,1,2,3\}$

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

    $\mathcal{D}(G) = \{(0,0),(1,0),(0,1),(1,1)\}$

    (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

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

    finite sequences of elements from Msg

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

## *System Properties*



## *Predicate Logic*

an observable

a logical variable from Var

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

$\in Var$

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

$\in Var$

$\in \mathbb{Z}$, constants

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

Example:

$\forall\, t \in Time \bullet \neg G(t) \Rightarrow \neg F(t)$

$\forall\, t \in Time \bullet H(t) \Rightarrow \exists\, t' \in [t, t+100] \bullet \cancel{F(t')}\ I(t')$

"$t_1 + 0$" "$t_2 + 100$"

we can't control the flame

so if this a requirement on the controller, we use I

# Predicate Logic

choice A: $Var = \{x, y, z\}$
choice B: $Var = \{a, b, c, \dots\}$
choice C: $Var = \{ \dots \}$

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

$obs$ 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

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

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



# Recall: Predicate Logic, Standard Semantics

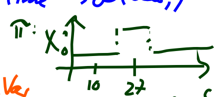
$\beta : \{ \dots \mapsto 27\}$

| | |
|---|---|
| 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 $t \in$ Time, set: | $\pi(t) = (d_1, \dots, d_n)$. |
| For convenience: use | $obs_i : \mathsf{Time} \to \mathcal{D}(obs_i)$. |

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

$\beta : Var \to Time$

- Let $\beta : \mathsf{Var} \to \mathsf{Time}$ be a **valuation** of the logical variables. $\in Var$
- $\pi, \beta \models \boxed{obs_i(t) = d}$ iff $obs_i(\beta(t)) = d$ $\quad \in \mathcal{D}(obs_i)$ $\quad obs_i : Time \to \mathcal{D}(obs_i)$
- $\pi, \beta \models \neg\varphi$ iff not $\pi, \beta \models \varphi$ $\quad$ projection of $\pi$ onto $obs_i$
- $\pi, \beta \models \varphi_1 \vee \varphi_2$ iff ...
- ...
- $\pi, \beta \models \forall t \in \mathsf{Time} \bullet \varphi$ iff for all $t_0 \in Time$, $\pi, \beta[t \mapsto t_0] \models \varphi$ $\quad$ ($t \in Var$, $t_0 \in \mathbb{R}_0^+$; $\beta[t \mapsto t_0]$: modification of $\beta$, s.t. $t$ is mapped to $t_0$, rest unchanged)
- $\pi, \beta \models \forall t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi$ iff for all $t_0 \in [\beta(t_1) + c_1, \beta(t_2) + c_2]$, $\pi, \beta[t \mapsto t_0] \models \varphi$




$\beta = \{t \mapsto 27\}$

$\pi, \beta \models \boxed{X(t) = 1}$ because $X(\beta(t)) = X(27) = 1$

$\pi, \beta' \not\models X(t) = 1$ if $\beta' = \{t \mapsto 10\}$

## Predicate Logic

all logical variables are quantified

Note: we can view a closed predicate logic formula $\varphi$ as a **concise description** of

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

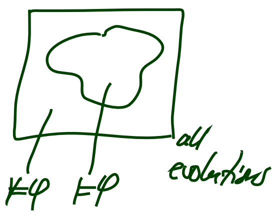
a set of evolutions

the set of all system evolutions satisfying $\varphi$.

For example,

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

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,

  defining Req as abbreviation for (*)

  $$\mathsf{Req} \;:\Longleftrightarrow\; \overbrace{\forall\, t \in \mathsf{Time} \bullet \neg(I(t) \land \neg G(t))}^{(*)}$$

  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,

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

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

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

We say

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

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

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

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

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

  Then

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

  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.

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

  Then

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

  is a duration property.

- This property can again be falsified in finite time.

*References*





## References

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