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 Properties in first order predicate logic
  - Classes of Timed Properties
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 on behaviour,
- a language to describe controller behaviour,
- a notion of “meet” — and a method to verify meeting.
Real-Time Behaviour, More Formally...
We assume that the real-time systems we consider is characterised by a finite set of **state variables** (or **observables**)

\[ \text{obs}_1, \ldots, \text{obs}_n \]

each equipped with a domain \( D(\text{obs}_i) \), \( 1 \leq i \leq n \).

**Example**: gas burner

- \( G : \{0, 1\} \rightarrow 0 \) iff valve closed
- \( F : \{0, 1\} \rightarrow 0 \) iff no flame
- \( I : \{0, 1\} \rightarrow 0 \) iff ignition off
- \( H : \{0, 1\} \rightarrow 0 \) iff no heating request
One possible evolution (or behaviour) of the considered system over time is represented as a function

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

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

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

For convenience, we use

$$obs_i : \text{Time} \rightarrow \mathcal{D}(obs_i)$$

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 of hardware considerations,
  - interesting view: continuous-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

\[ \pi : \text{Time} \to \mathcal{D}(\text{obs}_1) \times \cdots \times \mathcal{D}(\text{obs}_n) \]

with

\[ \pi(t) = (d_1, \ldots, d_n) \]

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

For convenience: use \( \text{obs}_i : \text{Time} \to \mathcal{D}(\text{obs}_i) \).
Example: Gas Burner

\[
\begin{align*}
&H \quad G \quad I \quad F \\
&0 \quad 0 \quad 0 \quad 0
\end{align*}
\]
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

  \[ G : \text{Time} \rightarrow \{0, 1, 2, 3\} \]

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

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

  \[ B : \text{Time} \rightarrow \text{Msg}^* \]

  to model the receive buffer as a finite sequence of messages from \text{Msg}.

- etc.
System Properties: A First Approach
\[ \varphi ::= \text{obs}(t) = d \mid \neg \varphi \mid \varphi_1 \lor \varphi_2 \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \implies \varphi_2 \mid \varphi_1 \iff \varphi_2 \mid \forall t \in \text{Time} \bullet \varphi \mid \forall t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi \]

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

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

\[ \text{obs}_i : \text{Time} \to \mathcal{D}(\text{obs}), 1 \leq i \leq n. \]

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

Evolution of system over time: $\pi : \text{Time} \to D(\text{obs}_1) \times \cdots \times D(\text{obs}_n)$. If $\text{obs}_i$ has value $d_i \in D(\text{obs}_i)$ at $t \in \text{Time}$, set: $\pi(t) = (d_1, \ldots, d_n)$. For convenience: use $\text{obs}_i : \text{Time} \to D(\text{obs}_i)$.

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

- Let $\beta : \text{Var} \to \text{Time}$ be a valuation of the logical variables.
- $\pi, \beta \models \text{obs}_i(t) = d$ iff $\text{obs}_i(\beta(t)) = d$
- $\pi, \beta \models \neg \varphi$ iff not $\pi, \beta \models \varphi$
- $\pi, \beta \models \varphi_1 \lor \varphi_2$ iff ...
- …
- $\pi, \beta \models \forall t \in \text{Time} \bullet \varphi$ iff for all $t_0 \in \text{Time}$, $\pi, \beta[t \mapsto t_0] \models \varphi$
- $\pi, \beta \models \forall t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi$ iff for all $t_0 \in [\beta(t_1) + c_1, \beta(t_2) + c_2]$, $\pi, \beta[t \mapsto t_0] \models \varphi$
Note: we can view a closed predicate logic formula $\varphi$ as a **concise description** of

$$\{ \pi : \text{Time} \rightarrow D(\text{obs}_1) \times \cdots \times D(\text{obs}_n) \mid \pi, \emptyset \models \varphi \},$$

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

For example,

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

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

\[
\text{Req} : \iff \forall t \in \text{Time} \bullet \neg (I(t) \land \neg G(t))
\]

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,

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

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

Req :\iff \forall t \in \text{Time} \bullet \neg (I(t) \land \neg G(t))

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

\pi \in \text{Req?}
\pi \in \text{Des?}

\begin{itemize}
  \item I(t_1) = 0
  \item G(t_1) = 1
  \quad \Rightarrow t_1 \text{ ok}
  \item I(t_2) = 0
  \item G(t_2) = 0
  \quad \Rightarrow t_2 \text{ ok}
  \item I(t_3) = 1
  \item G(t_3') = 1
  \quad \text{for } t_3 \leq t_3' < t_4 \text{ ok for } t_3
  \item I(t_4) = 0
  \quad \Rightarrow \text{ ok for } t_4
\end{itemize}
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 \((\text{Time} \rightarrow \times_{i=1}^{n} \mathcal{D}(\text{obs}_i))\), 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
  \[
  \text{Impl} \subseteq \text{Des} \quad \text{or} \quad \text{Impl} \subseteq \text{Req} 
  \]

If ‘Req’ and ‘Des’ are described by formulae of first-order 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 \( C : \text{Time} \rightarrow \{0, 1\} \) where \( C(t) = 1 \) represents a critical system state at time \( t \).

Then

\[
\forall t \in \text{Time} \quad \neg C(t)
\]

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 $G : \text{Time} \rightarrow \{0, 1\}$ where $G(t) = 1$ represents a good system state at time $t$.

  Then

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

  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 \(G : \text{Time} \rightarrow \{0, 1\}\) and request \(R : \text{Time} \rightarrow \{0, 1\}\).

  Then

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

  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 \(C : \text{Time} \rightarrow \{0, 1\}\). Then

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

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.

Almost

Strangest operators:

- **everywhere** — Example: $[G]$
  (Holds in a given interval $[b, e]$ iff the gas valve is open almost everywhere.)

- **chop** — Example: $(\lceil \neg I \rceil ; I ; \lceil \neg I \rceil) \implies \ell \geq 1$
  (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.)
We will introduce three (or five) syntactical “levels”:

(i) **Symbols:**
\[ f, g, \text{true}, \text{false}, =, <, >, \leq, \geq, x, y, z, X, Y, Z, d \]

(ii) **State Assertions:**
\[ P ::= 0 \mid 1 \mid X = d \mid \neg P_1 \mid P_1 \land P_2 \]

(iii) **Terms:**
\[ \theta ::= x \mid \ell \mid \int P \mid f(\theta_1, \ldots, \theta_n) \]

(iv) **Formulae:**
\[ F ::= p(\theta_1, \ldots, \theta_n) \mid \neg F_1 \mid F_1 \land F_2 \mid \forall x \bullet F_1 \mid F_1 ; F_2 \]

(v) **Abbreviations:**
\[ [\ ], [P], [P]^t, [P]^{\leq t}, \Diamond F, \Box F \]
Symbols: Syntax

- \( f, g \): function symbols, each with arity \( n \in \mathbb{N}_0 \).
  Called constant if \( n = 0 \).
  Assume: constants \( 0, 1, \ldots \in \mathbb{N}_0 \); binary ‘+’ and ‘.’.

- \( p, q \): predicate symbols, also with arity.
  Assume: constants \( true, false \); binary =, <, >, ≤, ≥.

- \( x, y, z \in \text{GVar} \): global variables.

- \( X, Y, Z \in \text{Obs} \): 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** \( \mathbb{B} = \{tt, ff\} \),
  - the **real numbers** \( \mathbb{R} \),
  - **time** \( \text{Time} \),
    (mostly \( \text{Time} = \mathbb{R}_0^+ \) (continuous), exception \( \text{Time} = \mathbb{N}_0 \) (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.
  \[
  \hat{f} : \mathbb{R}^n \rightarrow \mathbb{R}.
  \]

- The semantics of an \( n \)-ary **predicate symbol** \( p \)
  is a function from \( \mathbb{R}^n \) to \( \mathbb{B} \), denoted \( \hat{p} \), i.e.
  \[
  \hat{p} : \mathbb{R}^n \rightarrow \mathbb{B}.
  \]
Symbols: Examples

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

  - \( \hat{\text{true}} = \text{tt}, \hat{\text{false}} = \text{ff} \)
  - \( \hat{0} \in \mathbb{R} \) 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 \( \hat{0}, \hat{1}, \hat{+}, \hat{\cdot}, \hat{=}, \hat{<} \).”
The semantics of a global variable is not fixed (throughout the lecture) but given by a valuation, i.e. a mapping

$$V : \text{GVar} \rightarrow \mathbb{R}$$

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

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

Global variables are though fixed over time in system evolutions.
The semantics of a **global variable** is not fixed (throughout the lecture) but given by a **valuation**, i.e. a mapping

\[ \mathcal{V} : \text{GVar} \to \mathbb{R} \]

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

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

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

\[ \mathcal{I} : \text{Obs} \to (\text{Time} \to \mathcal{D}) \]

assigning each state variable \( X \in \text{Obs} \) a function

\[ \mathcal{I}(X) : \text{Time} \to \mathcal{D}(X) \]

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

\[ X_{\mathcal{I}} : \quad \mathcal{D}(X) \]

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