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

Real-Time Systems
Lecture 02: Timed Behaviour

2013-04-17

Dr. Bernd Westphal
Albert-Ludwigs-Universität Freiburg, Germany
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 (part) 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)
  \[ \text{obs}_1, \ldots, \text{obs}_n \]
  each equipped with a domain \( \mathcal{D}(\text{obs}_i) \), \( 1 \leq i \leq n \).

- Example: gas burner

\[ \begin{array}{ll}
\text{gas valve} & G, \quad \mathcal{D}(G) = \{0,1\}, \quad 0 \text{ if valve closed} \\
\text{flame sensor} & F, \quad \mathcal{D}(F) = \{0,1\}, \quad 0 \text{ if no flame} \\
\text{ignition} & I, \quad \mathcal{D}(I) = \{0,1\}, \quad 0 \text{ if no ignition} \\
\text{burner heat yes/\neg} & H, \quad \mathcal{D}(H) = \{0,1\}, \quad 0 \text{ if no heat} \\
\end{array} \]

System Evolution over Time

- One possible evolution (or behaviour) of the considered system over time is represented as a function
  \[ \pi : \text{Time} \rightarrow \mathcal{D}(\text{obs}_1) \times \cdots \times \mathcal{D}(\text{obs}_n) \].

- If (and only if) observable \( \text{obs}_i \) has value \( d_i \in \mathcal{D}(\text{obs}_i) \) at time \( t \in \text{Time} \), \( 1 \leq i \leq n \), we set
  \[ \pi(t) = (d_1, \ldots, d_n) \].

- For convenience, we use
  \[ \text{obs}_i : \text{Time} \rightarrow \mathcal{D}(\text{obs}_i) \]
  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**: $\text{Time} = \mathbb{N}_0$, the set of natural numbers.
  - **Continuous or dense time**: $\text{Time} = \mathbb{R}_0^+$, 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 $\pi : \text{Time} \rightarrow \mathcal{D}(\text{obs}_1) \times \cdots \times \mathcal{D}(\text{obs}_n)$.

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

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

For convenience: use $\text{obs}_i : \text{Time} \rightarrow \mathcal{D}(\text{obs}_i)$.
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
  \[ D(G) = \{0, 1, 2, 3\} \quad G : \text{Time} \to \{0, 1, 2, 3\} \]
  (But: \( D(G) \) is never continuous in the lecture, otherwise we had a hybrid system.)

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

  to model the receive buffer as a finite sequence of messages from \( \text{Msg} \).
  - etc.
Predicate Logic

\[ \varphi ::= \text{obs}(t) = d | \lnot \varphi | \varphi_1 \lor \varphi_2 | \varphi_1 \land \varphi_2 | \varphi_1 \Rightarrow \varphi_2 | \varphi_1 \iff \varphi_2 \]

| \forall t \in \text{Time} \bullet \varphi | \forall t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi

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

Example:

\[ \forall t \in \text{Time} \bullet \gamma(t) \Rightarrow \gamma(t) \]

\[ \forall t \in \text{Time} \bullet \phi(t) \Rightarrow \exists \varepsilon \in [\varepsilon, \varepsilon + 100] \bullet \psi(t) \]

\text{We need control the term so if this a requirement in the controller, we use I}
**Predicate Logic**

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

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

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

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

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**

\[ \beta : \text{Var} \to \{0,1\} \]

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.
- \[ \pi, \beta \models \text{obs}_i(t) = d \iff \text{obs}_i(\beta(t)) = d \]
- \[ \pi, \beta \models \neg \varphi \iff \pi, \beta \not\models \varphi \]
- \[ \pi, \beta \models \varphi_1 \lor \varphi_2 \iff \cdots \]
- \[ \pi, \beta \models \forall t \in \text{Time} \bullet \varphi \iff \forall t \in \text{Time}, \pi, \beta \models \varphi \]
- \[ \pi, \beta \models \forall t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi \iff \forall t \in [t_1 + c_1, t_2 + c_2], \pi, \beta \models \varphi \]

Projection of \( \pi \) onto \( \text{obs}_i \):

- \( \pi, \beta \models \forall t \in \text{Time} \bullet \varphi \iff \forall t \in \text{Time}, \pi, \beta \models \varphi \)
- \( \pi, \beta \models \forall t \in [t_1 + c_1, t_2 + c_2] \bullet \varphi \iff \forall t \in [t_1 + c_1, t_2 + c_2], \pi, \beta \models \varphi \)
- Because \( X(\pi(t)) = X(\beta(t)) + 1 \)
Predicate Logic

Note: we can view a closed predicate logic formula \( \varphi \) as a concise description of
\[
\{ \pi : \text{Time} \rightarrow \mathcal{D}(\text{obs}_1) \times \cdots \times \mathcal{D}(\text{obs}_n) \mid \pi, \emptyset \models \varphi \},
\]
the set of all system evolutions satisfying \( \varphi \).

For example,
\[
\forall t \in \text{Time} \cdot \neg (I(t) \land \neg G(t))
\]
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,
  \[
  \text{Req} : \iff \forall t \in \text{Time} \cdot \neg (I(t) \land \neg G(t))
  \]
  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,
  \[
  \text{Des} : \iff \forall t \in \text{Time} \cdot I(t) \implies \forall t' \in [t - 1, t + 1] \cdot G(t')
  \]
  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 \((\text{Time} \to \times_{i=1}^n D(\text{obs}_i))\), 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
  \[ \forall t \in \text{Time} \cdot \neg C(t) \]

  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
  \[ \exists t \in \text{Time} \cdot G(t) \]

  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

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

  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

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

  is a duration property.

- This property can again be falsified in finite time.
References