Contents & Goals

---

Last Lecture:

- /

This Lecture:

- Educational Objectives:
  - Be able to decide whether you want to stay with us or not. (IOW: an advertisement for the lecture.)
  - Agree on formalia.

Content:

- Overview: content (and non-content) of the lecture.
- Definition reactive, real-time, hybrid system.
- Outlook on methodology for precise development of (provably) correct real-time systems.

Formalism:

- Dates/times, exercises, exam admission.

Literature

- A formal model of real-time behaviour.

---

Introduction

---

Subject of the Lecture

---

What is a Real-Time System?

---

Classic example: Airbag Controller

- Controller
- Crash
- Fire

Requirement: "When a crash is detected, fire the airbag."

- When firing too early: airbag ineffective.
- When firing too late: additional threat.
- Say, 300 ms (plus/minus small ε) after a crash is the right time to fire.
- Then the precise requirement is "When a crash is detected at time \( t \), fire the airbag at \( t + 300 \) ms ± \( \varepsilon \)."

Other example: Gas Burner

- Gas valve
- Flamesensor
- Ignition

- Leakage is practically unavoidable:
  - For ignition, first open valve
  - Then ignite the available gas
  - Ignition may fail. . .

- Leakage is safety-critical: Igniting large amounts of leaked gas may lead to a dangerous explosion.
No, Really, What is a Real-Time System?

• The examples have in common that it matters, when in time the output for a given input (sequence) takes place. For instance,
  • "fire" 300ms after "crash",
  • within any interval of at least 60s, leakage (= have the gas valve open without a flame) amounts to at most 5% of the time.

Note: quantitative (here) vs. qualitative notion of time (untimed).

• Often: There is a physical environment, which has a notion of time, and which evolves while our controller is computing.

(Contrast: vending machine for soft-drinks:
  • If the customer is really thirsty, she'll wait.
  • Neither using a really fast nor a really slow contemporary controller causes a violation of (timing) requirements.

(Real) Contrast: transformational systems, such as computing \( \pi \).

Other Definitions

[Douglass, 1999]

• "A real-time system is one that has performance deadlines on its computations and actions."

• Distinguish:
  • "Hard deadlines: performance requirements that absolutely must be met and every event or time mark."
  "(Latedata can be bad data.)"
  • "Soft deadlines: for instance about average response times."
  "(Latedata is still good.)"

• Design Goal: A timely system, i.e. one meeting its performance requirements.

• Note: performance can in general be any unit of quantities:
  • (discrete) number of steps or processor instructions,
  • (discrete or continuous) number of seconds,
  • etc.

Definitions: Reactive vs. Real-Time vs. Hybrid Systems

• Reactive Systems interact with their environment by reacting to inputs from the environment with certain outputs.

• A Real-Time System is a reactive system which, for certain inputs, has to compute the corresponding outputs withingiven time bounds.

• A Hybrid System is a real-time system consisting of continuous and discrete components. The continuous components are time-dependent (!) physical variables ranging over a continuous valueset.

• A system is called Safety-Critical if and only if a malfunction can cause loss of goods, money, or even life.

Reactive Systems

Real-Time Systems

Hybrid Systems

The Problem: Constructing Safety-Critical RT Systems

• Reactivesystems can be partitioned into:
  • plant
  • sensors
  • actuators
  • controller

• "In constructing a real-time system, the aim is to control a physically existing environment, the plant, in such a way that the controlled plant satisfies all desired (timing) requirements."

• The design of safety-critical (reactive) systems requires a high degree of precision: We want—at best—to be sure that a design meets its requirements.

• Real-time systems are often safety-critical.

• The lecture presents approaches for the precise development of real-time systems based on formal, mathematical methods.
Leakage is safety critical: Igniting large amounts of leaked gas may lead to a dangerous explosion.

Controller program for ignition is easy:

\[
\text{while (!flame) do} \quad \text{open valve();} \\
\text{wait(t);} \quad \text{ignite();} \\
\text{od}
\]

Is it correct? (Here: Is it avoiding dangerous explosions?)

Prerequisites for precise development of real-time systems:

- 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 (or prove) "meeting".

Then we can devise a methodology to get from requirements to a (correct) implementation — here: following [Olderog and Dierks, 2008].

Sketch of the methodology: Gas Burner Example

- Requirements
  - At most 5% of any at least 60 s long interval amounts to leakage.

- Reflective design
  - Time intervals with leakage last at most 1 s.
  - After each leakage, wait 30 s before opening valve again.

- Constructive design
  - PLC Automaton: (open valve for 0.5 s; ignite; if no flame after 0.1 s close valve)

- Implementation
  - IEC61131-3 program

Content overview:

- Introduction
  - First-order Logic
  - Duration Calculus (DC)
- Semantical correctness
  - Proof with DC
  - DC decidability
  - DC implementables
- PLC Automata
- Timed Automata, Uppaal
- Networks of Timed Automata
- Region/Zone-Abstraction
- Extended Timed Automata

Undecidability results

[Formal description language I]

\[
\text{obs} : \text{Time} \rightarrow \text{D} \left( \text{obs} \right)
\]

\[
\langle \text{obs}_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_0} \langle \text{obs}_1, \nu_1 \rangle, t_1
\]

... whether T satisfies DC formula, observer-based abstraction...
Recall over-simplified airbag controller:

\[
\begin{aligned}
\text{while (true) do} & \quad \text{poll sensors()}; \\
& \quad \text{if (crash) tmr.start(300ms);} \\
& \quad \text{if (tmr.elapsed()) fire := 1;} \\
& \quad \text{update actuators();} \\
\od
\end{aligned}
\]

The execution of `poll sensors()` and `update actuators()` also take time! (And we have to consider it!)

Maybe in lecture:

How to determine the WCET of, for instance, C code. (As a science of its own.)
Schedule/Submission:

- Recall: exercises online on Thursday before (or so) lecture, regular turnin on corresponding tutorial day until 10:00 local time
- Should work in groups of max. 3, clearly give names on submission
- Please submit electronically by mail to me (cf. homepage), some LaTeX styles on homepage; paper submissions are tolerated

Didactical aim:

- Deal more extensively with notions from lecture (easy)
- Explore corner cases or alternatives (medium)
- Evaluate/appreciate approaches (difficult)
- Additional difficulty: imprecise/unclear tasks—by intention

True aim: most complicated rating system ever, namely two ratings
- Good-will (“reasonable solution with knowledge before tutorial”)
- Evil/Exam (“reasonable solution with knowledge after tutorial”)

10% bonus for early submission.

Exam Admission:

- 50% of the maximum possible non-bonus good-will points in total are sufficient for admission to exam

Exam Form:

- (oral or written) not yet decided

Speaking of grading and examination...

Mid-term Evaluation:
- We will have a mid-term evaluation, but we’re always interested in comments/hints/proposals concerning form or content.
  - That is, students are asked to evaluate lecture, lecturer, and tutor...

Questions:

- “Online”: (i) ask immediately or in the break
- “Offline”: (i) try to solve yourself (ii) discuss with colleagues (iii)...

Exercises: contact tutor by mail (cf. homepage)

Rest: contact lecturer by mail (cf. homepage) or just drop by: Building 52, Room 00-020
We assume that the real-time system 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
- Gas valve
- Flamesensor
- Ignition
- \( G \): \( \{0, 1\} \) — 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 of the considered system over time is represented as a function \( \pi : \text{Time} \to D(\text{obs}_1) \times \cdots \times D(\text{obs}_n) \).

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

For convenience, we use \( \text{obs}_i : \text{Time} \to D(\text{obs}_i) \) to denote the projection of \( \pi \) onto the \( i \)-th component.

**Level 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 \( G : \text{Time} \to \{0, 1, 2, 3\} \). (Domain is never continuous in the lecture, otherwise it's a hybrid system.)
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 \( G: \text{Time} \rightarrow \{0, 1, 2, 3\} \)
  (\( D(G) \) 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 \( B: \text{Time} \rightarrow \text{Msg}^\ast \)
to model the receive buffer as a finite sequence of messages from \( \text{Msg} \).
- etc.

References