Safety-Critical Real-Time Systems

ARTES PhD course

Lecture 3: Formal Analysis of Safety-related Properties

Simin Nadjm-Tehrani ©

Real-time Systems Laboratory

simin@ida.liu.se

Recall from earlier…

- Increased reliability does not necessarily increase safety
- What is the role of formal techniques in enhancing safety?

Fault to Accident

- Fault
- Error
- Failure
- Hazard
- Accident

Removing some faults enhances safety!

Why formal techniques?

- An engineering discipline:

  Using mathematics can never be wrong!

But what about the tools?

Many practitioners are interested in new solutions. They are prepared to listen to you and to try. However, the key to their problems delivered by researchers usually does not fit, and when the practitioner comes back complaining, he is told that it is not the key which is wrong, but the lock,... and the door, and the wall...

- Gerald Holzmann (ATT research)

Formal techniques

- The least we can do:
  - **Avoid/remove** (design) faults that lead to obvious bad things
  - **Analyse behaviour** in presence of selected combinations of potential external faults

Analysing digital designs

Show that the design specification M (the set of computations for M) is a model for the temporal logic formula expressing the requirements S:

$$M \models S$$

State space search

- To check that M satisfies S, we must check that no possible state in M contradicts S.
- Consider a model M with n Boolean state variables. This leads to a potential state space of 2^n.

State space search

- With 55 variables, at 1 MHz, it would take (in the worst case) over 1 billion years to visit every state!

Advanced techniques

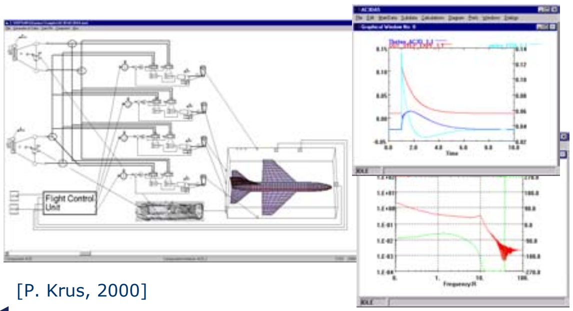
- Smart data structures for efficient representation of state space
- Smart deduction engines (satisfiability checkers) that find proofs fast
- Smart abstractions of the design to capture the essential properties
  e.g. synchronous languages

Engineering practice (trends)

- Detailed models and simulations for **non-digital hardware**
- Design models for **digital components** and functional analysis by
  - Simulations
  - (Formal verification)
  - (Semi-automatic code generation)
- Separate analysis for safety/reliability, selected models for FTA, FMEA

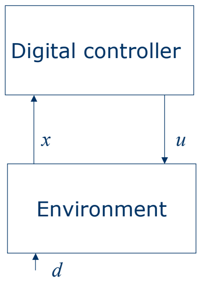
Non-digital hardware

Extensive simulations of coupled aircraft flight dynamics and actuator dynamics



[P. Krus, 2000]

System level properties



Design models, HW/SW

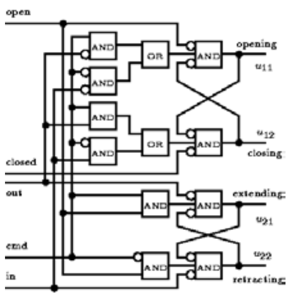
$$\dot{x} = f(x,u,d)$$

Example: Landing gear

- *Hazard ← failure ← … ← fault*
  - ***R_1*** : The door and gear do not collide under movement
  - ***R_2*** : Whenever the landing command is issued, the gear is extended and the doors closed within T seconds

Example: functional model

A landing gear controller:



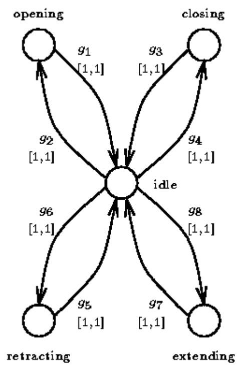
Avoiding bad things

In absence of external faults:

- Compositional verification
- To prove property R, find $R_1,\ldots, R_n$ such that $R_1 \& \ldots \& R_n \Rightarrow R$

- Prove R_i
  - in the controller (by logical analysis)
  - in the plant (by continuous analysis)
  - by further decomposition into $R_{i1},\ldots, R_{im}$

Adding delays: behavioural model



Transition conditions

Where the guards are:

$$\begin{aligned}
g_1 &\equiv \mathit{open} \vee [7] \vee [1] \\
g_2 &\equiv (\neg \mathit{cmd} \wedge ([8] \vee [2])) \vee (\mathit{cmd} \wedge ([9] \vee [3])) \\
g_3 &\equiv \mathit{closed} \vee [7] \vee [4] \\
g_4 &\equiv (\mathit{cmd} \wedge ([8] \vee [5])) \vee (\neg \mathit{cmd} \wedge ([9] \vee [6])) \\
g_5 &\equiv \mathit{out} \vee [9] \vee [7] \\
g_6 &\equiv (\mathit{cmd} \wedge ([4] \vee [6])) \vee [1] \\
g_7 &\equiv \mathit{in} \vee [1] \vee [7] \\
g_8 &\equiv \neg \mathit{cmd} \wedge ([4] \vee [5])
\end{aligned}$$

And ...

where the conditions $[k]$ are given by

$$\begin{aligned}
[1] &\equiv \neg \mathit{open} \wedge \mathit{closed} \wedge \neg \mathit{in} \wedge \neg \mathit{out} \\
[2] &\equiv \neg \mathit{open} \wedge \mathit{closed} \wedge \neg \mathit{in} \wedge \mathit{out} \\
[3] &\equiv \neg \mathit{open} \wedge \mathit{closed} \wedge \mathit{in} \wedge \neg \mathit{out} \\
[4] &\equiv \mathit{open} \wedge \neg \mathit{closed} \wedge \neg \mathit{in} \wedge \neg \mathit{out} \\
[5] &\equiv \mathit{open} \wedge \neg \mathit{closed} \wedge \neg \mathit{in} \wedge \mathit{out} \\
[6] &\equiv \mathit{open} \wedge \neg \mathit{closed} \wedge \mathit{in} \wedge \neg \mathit{out} \\
[7] &\equiv \neg \mathit{open} \wedge \neg \mathit{closed} \wedge \neg \mathit{in} \wedge \neg \mathit{out} \\
[8] &\equiv \neg \mathit{open} \wedge \neg \mathit{closed} \wedge \neg \mathit{in} \wedge \mathit{out} \\
[9] &\equiv \neg \mathit{open} \wedge \neg \mathit{closed} \wedge \mathit{in} \wedge \neg \mathit{out}
\end{aligned}$$

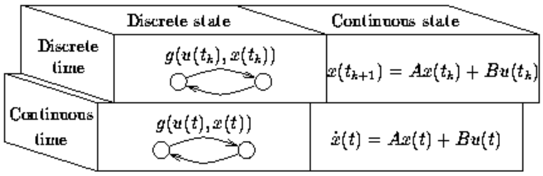
9th Oct 2003

Risk Forum:

- Flight International, 21-27 October 2003, reported a braking problem occurring on a Eurofighter Typhoon aircraft that led to the suspension of all flights. A cockpit warning light came on during landing, the pilot deployed the braking parachute, but the brakes could be used to bring the aircraft to a halt.

- The furlough lasted three weeks, and the aircraft were to return to flight operations last week. Apparently 15 days have been lost from the flight test program. The braking problem centered on a faulty microchip in the landing gear computer.

Modelling the environment



Mathematical approximations

Specification languages

- Large choice depending on
  - chosen level of abstraction
  - property of interest
  - nature of design models

Analysis approaches

- **Variation I**: Timed and hybrid logics to represent and reason about safety and timeliness properties at design level

- **Variation II**: Analysis at “program” level, code of digital controller

Deductive methods

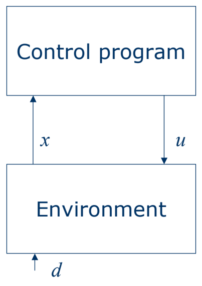
Variation I:
Formalising the properties in
Duration Calculus

$$R_1 : \quad \Box \neg \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil$$
$$R_2 : \quad \Box \neg ((\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \geq 1 \wedge x_d \leq 0) \rceil)$$

Correctness is proved using rules in the calculus.

SAT prover

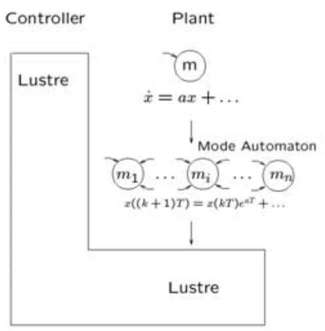
Variation II:



Synchronous Languages:
Lustre (Mealy automata)

$$\dot{x} = f(x,u,d)$$

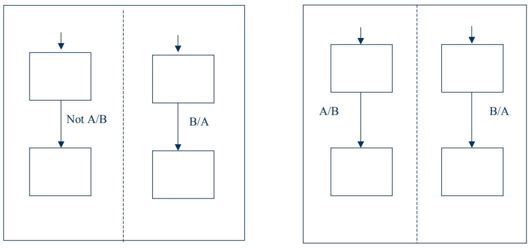
Timeliness: steps & intervals



Some desirable properties

- **Abstraction**: do not fix time granularity until very final stages of implementation
- **Causality**: an event should not be able to trigger itself
- **Consistency**: transitions taken in a step are not disabled by parallel transitions

What is desirable here?



The bright side

- ✪ Causality, determinism, consistency: dealt with by the compiler
- ✪ Connections to code optimisation and verification tools
  - efficient code generation in C, Ada or VHDL
  - Prover plugin or SMV format

State-of-art

- Are formal techniques used in development of real safety-critical systems?
- Yes!
  - Mainly (digital) HW verification
  - SW, some examples:
    - Lustre runs in new Airbus models
    - SPARK Ada in military applications
    - Recently: C code verification for Airbus!

A ”success” story

- C130J Hercules safety-critical software
- At selling time - after all certification :-(
- Combination of inspections, static analysis (formal verification)
- 70 man-years, 11590 anomalies
- 3% of anomalies safety-critical