CS 228 : Logic in Computer Science

Krishna. S

Some Real Life Stories

Therac-25(1987)

- The Therac-25 : radiation therapy machine produced by Atomic Energy of Canada Limited (AECL)
- Involved in at least six accidents, in which patients were given massive overdoses of radiation, approximately 100 times the intended dose.

Therac-25(1987)

- The Therac-25 : radiation therapy machine produced by Atomic Energy of Canada Limited (AECL)
- Involved in at least six accidents, in which patients were given massive overdoses of radiation, approximately 100 times the intended dose.
- Design error in the control software (race condition)

Intel Pentium Bug (1994)

- The Intel FDIV bug : Bug in the intel P5 floating point unit
- Discovered by a professor working on Brun's constant
- $(\frac{1}{3}+\frac{1}{5})+(\frac{1}{5}+\frac{1}{7})+(\frac{1}{11}+\frac{1}{13})+(\frac{1}{17}+\frac{1}{19})+\ldots$ converges to $B \cong 1.90216054$
- Intel offered to replace all flawed processors

Ariane 5 (1996)

- ESA (European Space Agency) Ariane 5 Launcher
  - Shown here in maiden flight on 4th June 1996
- Self destructs 37 secs later
  - uncaught exception: data conversion from 64-bit float to 16-bit signed int

Toyota Prius (2010)

- First mass produced hybrid vehicle
  - software “glitch” found in anti-lock braking system
  - Eventually fixed via software update in total 185,000 cars recalled, at huge cost

Nest Thermostat (2016)

- Nest Thermostat, the smart, learning thermostat from Nest Labs
  - software “glitch” led several homes to a frozen state, reported in NY times, Jan 13, 2016. May be, old fashioned mechanical thermostats better!

What do these stories have in common?

- Programmable computing devices
  - conventional computers and networks
  - software embedded in devices
- Programming error direct cause of failure
- Software critical
  - for safety
  - for business
  - for performance
- High costs incurred: financial, loss of life
- Failures avoidable

Formal Methods

Intuitive Description

"Applied Mathematics for modelling and analysing ICT systems"

Formal methods offer a large potential for:

- obtaining an early integration of verification in the design process
- providing more effective verification techniques (higher coverage)
- reducing the verification time

Simulation and Testing

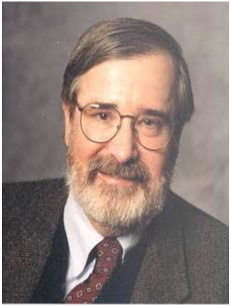
Basic procedure

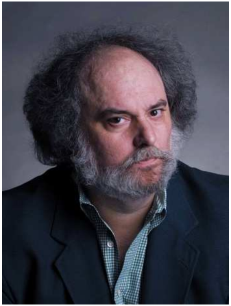
- Take a model
- Simulate it with certain inputs
- Observe what happens, and if this is desired

Important Drawbacks

- possible behaviours very large/infinite
- unexplored behaviours may contain fatal bug
- can show presence of errors, not their absence

Model Checking



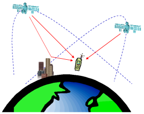


- Year 2008 : ACM confers the Turing Award to the pioneers of Model Checking: Ed Clarke, Allen Emerson, and Joseph Sifakis
- Why?

Model checking

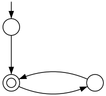
- Model checking has evolved in last 25 years into a widely used verification and debugging technique for software and hardware.
- Cost of *not* doing formal verification is high!
  - The France Telecom example
  - Ariane rocket: kaboom due to integer overflow!
  - Toyota/Ford recalls
- Model checking used (and further developed) by companies/institutes such as IBM, Intel, NASA, Cadence, Microsoft, and Siemens, and has culminated in many freely downloadable software tools that allow automated verification.

What is Model Checking?



System $\xrightarrow{\text{satisfy?}}$ specification

System Model $\xrightarrow[\models ?]{\text{model-checking}}$ Spec Model



ϕ

Model Checker as a Black Box

- Inputs to Model checker : A finite state system M, and a property P to be checked.
- Question : Does M satisfy P?
- Possible Outputs
  - Yes, M satisfies P
  - No, here is a counter example!.

What are Models?

Transition Systems

- States labeled with propositions
- Transition relation between states
- Action-labeled transitions to facilitate composition

Expressivity

- Programs are transition systems
- Multi-threading programs are transition systems
- Communicating processes are transition systems
- Hardware circuits are transition systems
- What else?

What are Properties?

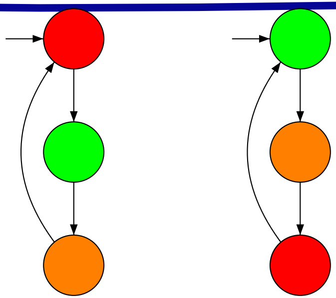
Example properties

- Can the system reach a deadlock?
- Can two processes ever be together in a critical section?
- On termination, does a program provide correct output?

Logics of Relevance

- Classical Logics
  - First Order Logic
  - Monadic Second Order Logic
- Temporal Logics
  - Propositional Logic, enriched with modal operators such as $\square$ (always) and $\diamond$ (eventually)
  - Interpreted over state sequences (linear)
  - Or over infinite trees (branching)

Two Traffic Lights



1. The traffic lights are never green simultaneously
   $\forall x(\neg(green_1(x) \wedge green_2(x)))$ or $\Box(\neg(green_1 \wedge green_2))$
2. The first traffic light is infinitely often green
   $\forall x \exists y(x < y \wedge green_1(y))$ or $\Box\Diamond green_1$
3. Between every two occurrences of traffic light 1 becoming red, traffic light 2 becomes red once.

The Model Checking Process

- Modeling Phase
  - model the system under consideration
  - as a first sanity check, perform some simulations
  - formalise property to be checked
- Running Phase
  - run the model checker to check the validity of the property in the model
- Analysis Phase
  - property satisfied? $\rightarrow$ check next property (if any)
  - property violated? $\rightarrow$
    - analyse generated counter example by simulation
    - refine the model, design, property, ... and repeat entire procedure
  - out of memory? $\rightarrow$ try to reduce the model and try again

The Pros of Model Checking

- widely applicable (hardware, software...)
- allows for partial verification (only relevant properties)
- potential “push-button” technology (tools)
- rapidly increasing industrial interest
- in case of property violation, a counter example is provided
- sound mathematical foundations
- not biased to the most possible scenarios (like testing)

The Cons of Model Checking

- model checking is only as “good” as the system model
- no guarantee about completeness of results (incomplete specifications)

Neverthless:
Model Checking is an effective technique to expose potential design errors

Striking Model-Checking Examples

- Security : Needham-Schroeder encryption protocol
  - error that remained undiscovered for 17 years revealed (model checker SAL)
- Transportation Systems
  - Train model containing 10^{47} states (model checker UPPAAL)
- Model Checkers for C, JAVA, C++
  - used (and developed) by Microsoft, Intel, NASA
  - successful application area: device drivers (model checker SLAM)
- Dutch storm surge barrier in Nieuwe Waterweg
- Software in current/next generation of space missiles
  - NASA's
    - Java Pathfinder, Deep Space Habitat, Lab for Reliable Software

Relevant Topics

- What are appropriate models?
  - from programs, circuits, communication protocols to transition systems
- What are properties?
  - Safety, Liveness, fairness
- How to check regular properties?
  - finite state automata and regular safety properties
  - Buchi automata and ω-regular properties

Relevant Topics

- How to express properties succintly?
  - First Order Logic (FO) : syntax, semantics
  - Monadic Second Order Logic (MSO) : syntax, semantics
  - Linear-Temporal-Logic (LTL) : syntax, semantics
  - What can be expressed in each logic?
  - Satisfiability and Model checking : algorithms, complexity
- How to make models succint?
  - Equivalences and partial-orders on transition systems
  - Which properties are preserved?
  - Minimization algorithms