Software verification

Will it ever work?

Ofer Strichman, Technion
- Testing: does the program behave as expected for a given set of inputs?

- Formal Verification: does the program behave as specified for all possible inputs.
Two main challenges, then

- **Specify** – what do you expect?

- **Verify** – is your expectation satisfied?
Specification

- Layers of specification = layers of abstraction

Temporal Logic: $G (move \land \neg closed-door)$

- Frequently as hard as programming itself

Implementation

if (waiting-time > 10 & …)
    close-door();
    move(up);
…
if (stop-signal)
    open_door();

When moving, door should be closed
Verification = Confrontation
(of abstraction levels)

- Automatic software verification, in general, is impossible (‘undecidable’).
- Like many other ‘impossible’ problems, it is still a valuable area to research:
  - Restrict the problem
  - Restrict the expectations
Restrictions

- Restrict the problem: Finite (& small) state
  - Hardware
  - Protocols, Controllers, Embedded programs, etc.

- Restrict the expectations:
  - Incomplete tools
  - Not fully automatic
By the way,

- Since this is the **Israel Innovation Summit** and we are in **Haifa**…

When it comes to formal hardware verification…

- Haifa happens to be the city with the **highest concentration of formal verification research in the world.**
- Thank you IBM, Intel, Motorola, Marvel, …
Software automatic verification - status

□ Some success stories:
  ■ Microsoft
  ■ AT&T and others
  ■ Microsoft, NASA, GE, General-Dynamics,…
  ■ NASA
  ■ Many more…

□ Common to all: incomplete (inherent). The question is: *how incomplete they are*, and how *efficient* they are.
Tony Hoare’s grand challenge

- In a JACM 2003 paper, T. Hoare declared the construction of a ‘verifying compiler’ as a grand challenge to computer science.

“…revives an old challenge: the construction and application of a verifying compiler that guarantees correctness of a program before running it.”

“… We now have a good theoretical understanding of how to describe what programs do, how they do it, and why they work.”
Deliverables of the grand challenge

- **We want:**
  - Verification technology will be integrated into commercial tools
  - The costs associated with program error will be significantly reduced (~ $60 B. in the US)

- **For this we need:**
  - A comprehensive and unified theory of programming, incl. concurrency, OO, inheritance…
  - A coherent toolset based on the theory

- **The challenge:**
  - 1,000,000 lines of (specified &) verified code
Some of the believers:

Patrick Cousot

John Rushby

Gerard Holzman

Shankar

Tony Hoare

J Moore

Amir Pnueli...
If nothing else…

- Take all the articles, people, software, ideas that are currently available
- Integrate them
- Et Voila!
If nothing else…

- **Short-term goals:**
  - A repository of benchmarks
  - Cross-tools translators
  - Language standards

- **Several sub-committees:**
  - Tool Integration and Interoperability
  - Digital Repository and Challenge Codes
  - Correctness by Construction
  - System Certification
Will it work? A. Pnueli’s answer.

The unanticipated success of hardware verification has been a 15 year exciting diversion (and distraction).

- It gave us most valuable techniques such as BDD’s, SAT, and model checking.
- It also cultivated the false illusion that most verification tasks can be solved by a press-button methods.

and...

Now that we have honed many interesting techniques on the toy problems of hardware verification, it is time to go back to the main goal of software verification.

From which he concludes…

User Guidance is Unavoidable
A grand-challenge: program-equivalence

- At the Technion, we are now trying to prove equivalence of two related C programs

- From Regression Testing to Regression Verification
  - Saves the need for specification
  - Offers numerous opportunities for optimizations
Equivalence of C programs

- Some usage scenarios:
  - after manual re-factoring,
  - test equivalence with and without an optimization
  - side-effects of new code
  - …
The Grand Challenge


- Compare to:
  - ‘put a man on the moon in 10 years’
  - Map the Human Genome
  - Unify the four forces of Physics
  - …
The grand challenge

- Is modeled on the Human Genome project (no less!) and shares many of its properties
  - A clear set of deliverables
  - A planned route to application
  - In areas of great potential benefit
  - looks 15 years ahead
  - …and changed the mode of conduct of science
What is ‘a correct program’?

- Brings to mind the classical philosophical question about verification: how can anything be established as ‘the truth’?

- Is Relativity Theory correct?

- K. Popper: Science progresses by refutation of theories, not their verification.
A theory

Nature
(Nature got it right)

A program

Specification
(as good as it gets)

**Student**: “What would you have done if your prediction had been refuted”?

**Einstein**: “I would have been sorry for the dear Lord, because the theory is correct”