Instead of a Motivation: a provocation. Test vs. Proof

Burkhart Wolff

Université Paris-Sud, LRI, CNRS

Part I

- Test vs. Proof: An old controversy
  - Can proofs guarantee the "Absence of Errors"
  - Are deductive verifiers "better" than testers?
  - Can we avoid Tests ? Or Reality ?
- HOL-TestGen: A verification and validation approach by Model-based Testing (MBT)
- HOL-TestGen: Achievements FOR Proofs
- The Future of (Model-based) Testing

Test vs. Proof: An old controversy

- "Dijkstra's Verdict" :

,   Program testing can be used to show the presence of bugs, but never to show their absence!

Test vs. Proof: An old controversy

- "Dijkstra's Verdict" :

  , Program testing can be used to show the presence of bugs, but never to show their absence!

- Well, Dijkstra was party; so can he be trusted ?

Test vs. Proof: An old controversy

- "Dijkstra's Verdict" :

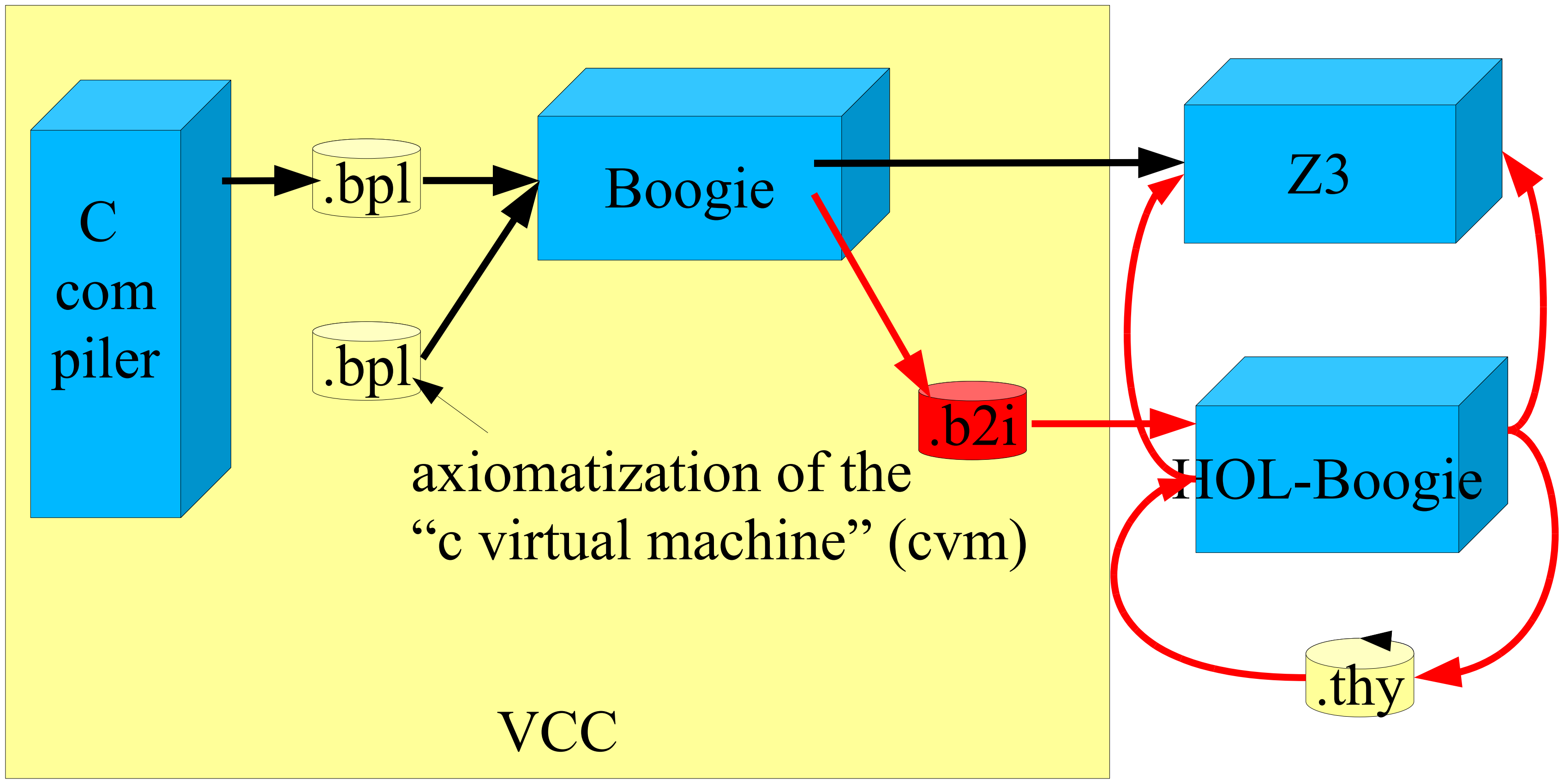
  , Program testing can be used to show the presence of bugs, but never to show their absence!

- So: can proof-based verifications guarantee the

  "abscence of bugs" ?

Test vs. Proof: An old controversy

- An Architecture of a Program Verifier (VCC) HOL-Boogie [Böhme, Wolff]



Test vs. Proof: An old controversy

- The ugly reality: deductive verification methods make a lot of assumptions *besides being costly in brain-power!
  - operational semantics should be faithfully executed
  - complex memory-machine model consistent (VCC: 800 axioms)
  - correctness of the vc generation (for concurrent C with "ownership", "locks", … !):
  - correctness of the vc generator and prover
  - abscence of an environment (= Operating System) that manipulates the underlying state.

Test vs. Proof: An old controversy

- Back to "Dijkstra's Verdict" :

  , Program testing can be used to show the presence of bugs, but never to show their absence!

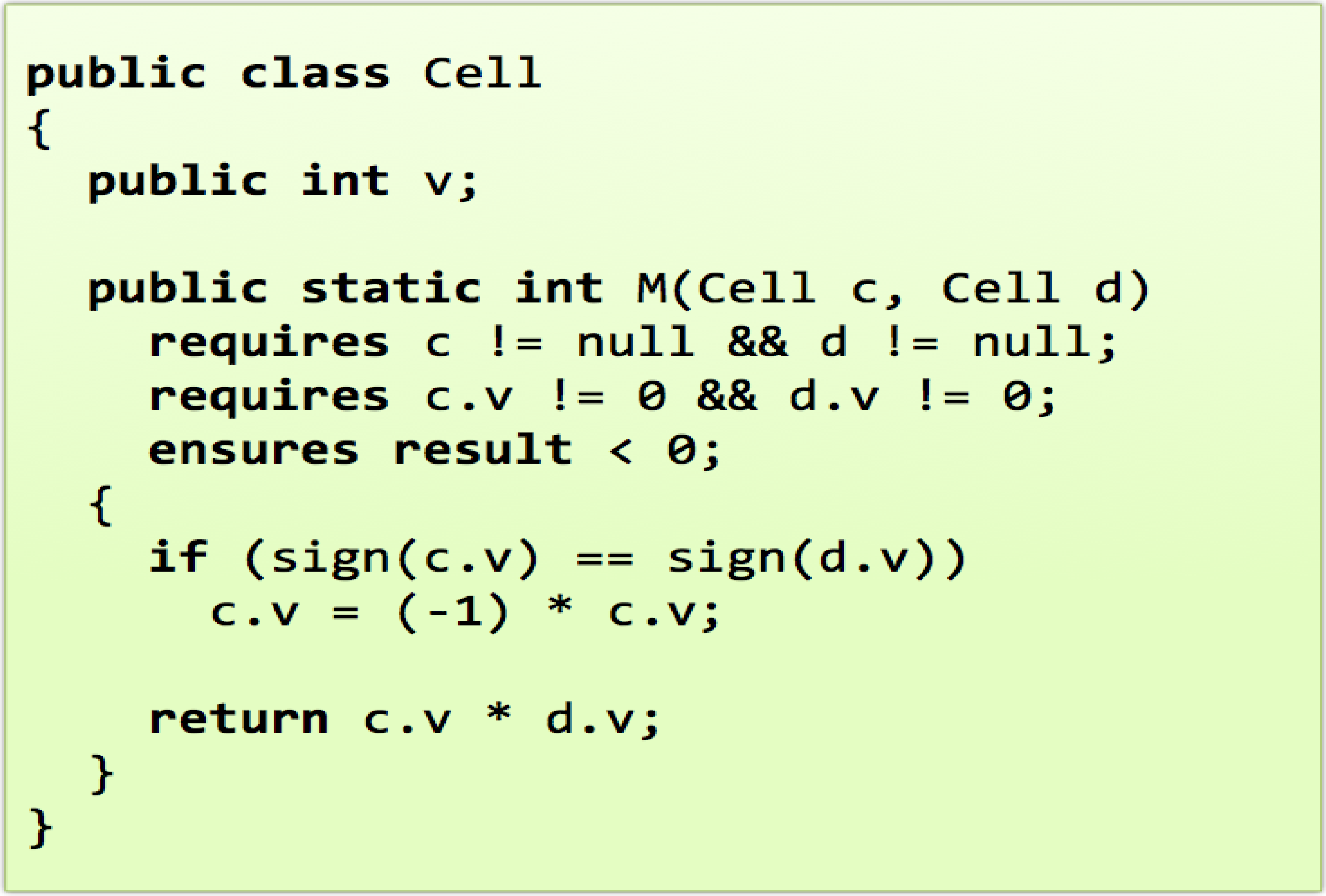
- Deductive Verification infers Properties on infinite sets of inputs; aren't they then

  "always better than tests" ?

Test vs. Proof: An old controversy

- Well, this depends on these assumptions … See the (very nice) example of Maria Christakis,

  where for a simple program:



```
public class Cell
{
  public int v;

  public static int M(Cell c, Cell d)
    requires c != null && d != null;
    requires c.v != 0 && d.v != 0;
    ensures result < 0;
  {
    if (sign(c.v) == sign(d.v))
      c.v = (-1) * c.v;

    return c.v * d.v;
  }
}
```

Test vs. Proof: An old controversy

- Well, this depends on these assumptions ...

  ... two different tools
  - Clousot (deductive based verification)
  - Pex (white-box tester)

  provide alltogether differently false results, since their underlying assumptions on arithmetics and memory model are simply different.
  Accidently, the Pex-Verdict is actually more correct than Clousots ...

Test vs. Proof: An old controversy

- "Dijkstra's Verdict" :

, Program testing can be used to show the presence of bugs, but never to show their absence!

Can we actually always avoid testing ?

Test vs. Proof: An old controversy

- "Dijkstra's Verdict" :

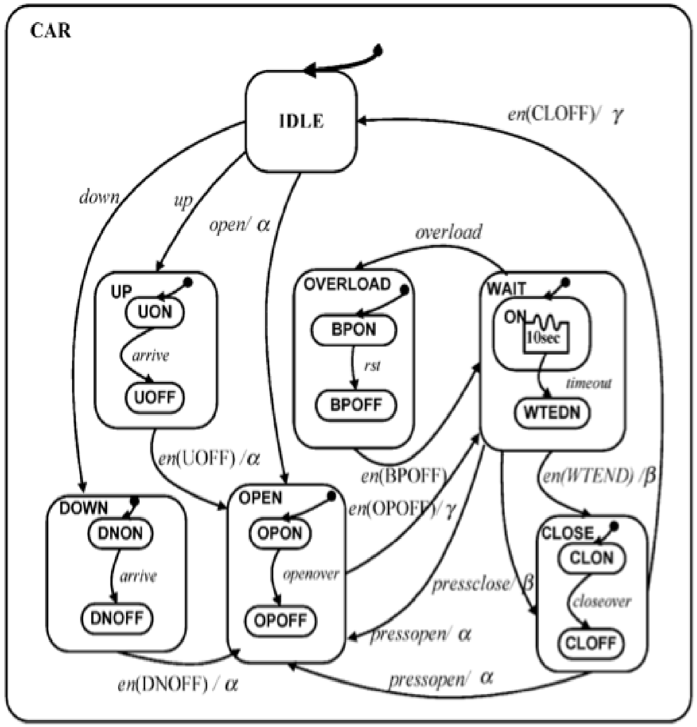
  , Program testing can be used to show the presence of bugs, but never to show their absence!

- "Einsteins scepticism":

  As far as the laws of mathematics refer to reality, they are not certain, as far as they are certain, they do not refer to reality.

Test vs. Proof: An old controversy

.



a posteriori

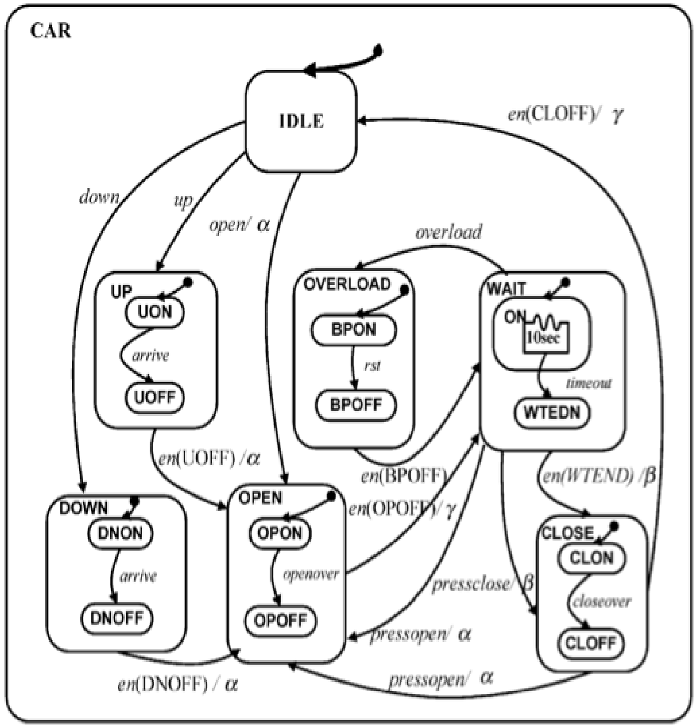
learning by experimenting

Model
(behaviour, and data !)

System
(hard + software)

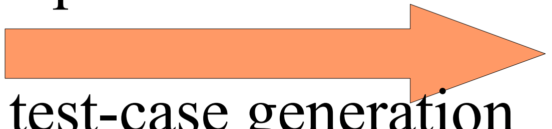
Test vs. Proof: An old controversy

.



a priori

test-case generation



a posteriori

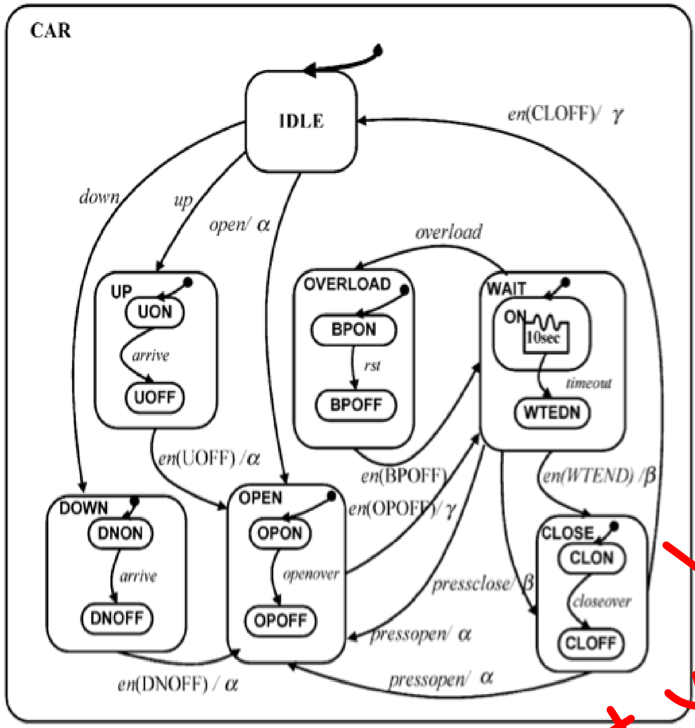
learning by experimenting

Model
(behaviour, and data !)

System
(hard + software)

Test vs. Proof:
An old controversy

.



a priori

test-case generation

Validation Problem:

What you can't do with Verification

a posteriori

learning by experimenting

Model
(behaviour, and data !)

System
(hard + software)

Verification by Model-based Testing …

- … can be done post-hoc; significant projects "reverse engineer" the model of a legacy system

- … attempts to find bugs in specifications EARLY (and can thus complement proof-based verification …)

- … can help system integration processes in a partly unknown environment ("embedded systems")

Nothing of this can be done by deductive verification methods !

Test vs. Proof:
Is it actually still a controversy?

- Dijkstra - Test :

  , Would Dijkstra fly with an aeroplane which is verified by deduct. methods alone ?

  ,

- Well, that's illegal.
  Certification bodies (CC, DO183) require tests, (and are very reluctant at proofs)

Test vs. Proof:
Is it actually still a controversy?

- Microsoft: Five major verification tools:
  Pex (Structural Test), SAGE(Fuzz Test) and
  Dafny, Spec#, VCC (VCG) use SMT solver Z3 !

- Test and Proofs, are they actually adversaries?
  (Tony Hoare, POPL2012, "says meanwhile no").

HOL-TestGen: A model-based approach to Verification

- Vision of HOL-Testgen
    - HOL-TestGen provides:
        - A formal testcase-generation method based on the solution of logical constraints

HOL-TestGen: A model-based approach to Verification

- HOL-TestGen provides:
  - A formal testcase-generation method based on the solution of logical constraints
  - Built-on top of an interactive theorem proving environment, it allows to combine automated provers with user intelligence

Conclusion: Test & Proof

- … can never ever establish the absense of "Bugs" in a system! Never ever. Both of them.
- … can, when combined, further increase confidence in verification results by using mutually independent assumptions.
- … can, when combined, offer new ways to tackle abstraction and state space explosion. (Normalization Theorems, Massage of Constraint Systems, …)

Conclusion

Conclusion: Test & Proof

- Is Testing actually a Verification Method ?

  Yes, when used to check that a program conforms to a specification (a "model").

  In the sense: did we get the program right ?

  It depends of the conformance notion.

Conclusion: Test & Proof

- … but Testing can actually be Validation Method:

  Yes, when used to check that a specification builds a useful "model" of a system.

  In the sense: experimenting.

  In the sense: did we get the right model?