Integrating Formal Program Verification with Testing

Cyrille Comar, Johannes Kanig and Yannick Moy
Integrating Formal Program Verification with Testing

Cyrille Comar, Johannes Kanig and Yannick Moy

Adacore
The GNAT Pro Company

HI-LITE
Simplifying the use of formal methods

openDO
Integrating Formal Program Verification with Testing

Cyrille Comar, Johannes Kanig and Yannick Moy
HI-LITE | Motivation
Formal methods [...] might be the primary source of evidence for the satisfaction of many of the objectives concerned with development and verification.

2011: Formal Methods Supplement (DO-333)
Myths of formal methods

• Myth 4: Formal methods require highly trained mathematicians

• Myth 5: Formal methods increase the cost of development

• Myth 6: Formal methods are unacceptable to users

• Myth 7: Formal methods are not used on real, large-scale software

(Anthony Hall, Praxis Systems, 1990)
Since 2001, Airbus has been integrating several tool supported formal verification techniques into the development process of avionics software products.

2009: Formal Verification of Avionics Software Products (Souyris, Wiels, Delmas, Delseny)
**Programming Contracts**

- **Hoare logic (1969)**
- **Logic contracts for proofs**
  - SPARK (1987)
- **Executable contracts for tests**
  - Eiffel DbC (1986)
- **Hi-Lite: executable annotation language??**
function One_Of (V, X, Y : in Int) return Boolean is (V = X or else V = Y);

function Max (X, Y : in Int) return Int with
  Pre => X /= Y,
  Post => Max’Result >= X and then
          Max’Result >= Y and then
          One_Of (Max’Result, X, Y);

function Max (X : in Int_Array) return Int with
  Post => (for all J in X’Range =>
           Max’Result >= X(J)) and then
          (for some J in X’Range =>
           Max’Result = X(J));
Testing vs. Formal Verification

use Q code cover P constructs

P calls Q

P calls Q

prove pre of Q assume post of Q

prove post of Q

local exhaustivity argument:
each function covered
→ enough behaviors explored

global soundness argument:
all functions proved
→ all assumptions justified
Combining tests and proofs should be AT LEAST AS GOOD AS verification based on tests only.
Caution: contracts are not only pre/post!

```
procedure Open
  (Customer : in Identity.Name;
   Id       : in Identity.Id;
   Cur      : in Money.CUR;
   Account  : out Account_Num)
with
Pre => not Max_Account_Reached,
Post => Existing (Account)...
```
HI-LITE  Combination 1: tested calls proved

during testing:
check that precondition of Q is respected

assumption for proof:
precondition of Q is respected
Combination 2: proved calls tested

During testing:
- Check that postcondition of P is respected

Assumption for proof:
- Postcondition of P is respected

Q calls P

P is tested

Q is proved
Testing + Formal Verification

**local** exhaustivity argument:
- test: function covered
- proof: by nature of proof

**global** soundness argument:
- proof: assumptions proved
- test: assumptions tested

Testing must check additional properties
Done by compiler instrumentation
Airbus 5 “must-have” of formal methods

• Soundness
• Applicability to the code
• Usability by normal engineers on normal computers
• Improve on classical methods
• Certifiability

Current work
HI-LITE Benefits of openness

- announcements
- meeting slides
- articles / docs

public:
- meeting minutes
- technical work
- 69 members

private:
- management
- partner code

- all code
- dev docs
- user docs

→ external collaborations with industry and academia