The Future of Formal Software Verification in Avionics

Yannick Moy
Formal Methods 2012 Industry Day
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)
Programming Contracts

{P}C{Q}  Hoare logic (1969)

logic contracts for proofs  executable contracts for tests


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));
Example of contract

**Example:**
- A list of event detection statuses
- Request to reset the detection status for Event

```plaintext
procedure Reset_Event_Status (Event : in T_Event) with
Post =>

not Event_Status(Event).Detection and
(for all Other_Event in T_Event =>
 (if Other_Event /= Event then
   Event_Status( Other_Event ) = Event_Status'Old( Other_Event ) ));

The detection status is unchanged
```

<table>
<thead>
<tr>
<th>Event1</th>
<th>Event</th>
<th>Event3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not detected</td>
<td>Not detected</td>
<td>Detected</td>
</tr>
</tbody>
</table>
GNAT compiler support for Ada 2012

- Run-time checking of new assertions
- New aspect to formally specify test cases
- New switch to choose semantics of integers in assertions, e.g. unbounded integers or largest machine integers
- New library of containers adapted to formal verification (lists, sets, maps, vectors)
- New run-time checks for integrating tests and proofs
Testing vs. Formal Verification

**Local** exhaustivity argument:
- each function covered
  → **enough** behaviors explored

**Global** soundness argument:
- all functions proved
  → **all** assumptions justified

- use Q code cover P constructs
- prove pre of Q assume post of Q
- assume pre of Q prove post of Q
- P calls Q
- actual body of Q or stub…

P calls Q
Combining tests and proofs should be AT LEAST AS GOOD AS verification based on tests only.
**HI-LITE**  Combination 1: tested calls proved

*During testing:*
check that
precondition of Q
is respected

**assumption for proof:**
precondition of Q
is respected
HI-LITE  Combination 2: proved calls tested

during testing: check that postcondition of P is respected

P is tested
Q calls P
Q is proved

assumption for proof: postcondition of P is respected
HI-LITE  Caution: contracts are not only pre/post!

```haskell
1  procedure Open
2    (Customer : in Identity.Name;
3     Id : in Identity.Id;
4     Cur : in Money.CUR;
5     Account : out Account_Num)
6    with
7      Pre => not Max_Account_Reached,
8      Post => Existing (Account)...
```

strong typing
parameters not aliased
parameters initialized

Data dependences

Testing must check additional properties
Done by compiler instrumentation
**HI-LITE** Testing + Formal Verification

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

**global** soundness argument:
- proof: assumptions proved
- test: assumptions tested
HI-LITE GNAT toolsuite

- GNAT compiler
- GNAT tests unit testing
- GNAT prove unit proof
- GNAT merge aggregate results

 executable
Hi-Lite GNATmerge result

Locations

Fold Builder results (4 items)

Fold segway.ads (3 items)

16:4 goal not reached at Speed_Is_Valid (TEST FAILED)

22:4 info: goal reached at State_Update (TEST FAILED, PROVED)

28:4 info: goal reached at Execute (TEST PASSED)

Fold segway.adb (1 item)
Airbus 5 “must-have” of formal methods

- Soundness
- Applicability to the code
- Usability by normal engineers on normal computers
- Improve on classical methods
current work
- Certifiability