Tool Aided Proof of properties of avionics programs
Contents

- Formal verification techniques
- Formal verification tools for the A380
- Focus on Caveat
- Unit Proving
Formal verification techniques

- Abstract Interpretation Based Static Analysis of programs
- Deductive methods
- Model checking

- BUT ALL OF THEM ARE ABSTRACT INTERPRETATION (Cousot and Cousot)

- Since the subject is the proof of properties of real programs, we do not consider Model Checking
Formal verification tools for the A380 (1)

- aiT for WCET computation
  - Developed by AbsInt (www.absint.com)
  - Abstract Interpretation based; analysis of binary code; Model of the CPU (and chips impacting the timings)
  - First usage on A380
  - DO178B level A
  - Worst Case Execution Time computation of programs running on complex CPU

- Stack analyzers for stack consumption computation
  - Developed by AbsInt (www.absint.com)
  - Abstract Interpretation based; analysis of binary code
  - First usage on A380
  - DO178B level A, B and C
  - Tighter maximum stack usage computation
• RTE-ENS : proof of absence of Run Time Errors
  ‣ Developed by ENS (PR, Cousot’s team at Ecole normale supérieure de Paris)
  ‣ Abstract Interpretation based (of course); analysis of C code
  ‣ First usage on A380
  ‣ DO178B level A
  ‣ Proof of absence of RTE like division by zero, numerical overflow, out of bounds access to an array
Formal verification tools for the A380 (3)

- Fluctuat: precision of Floating-point calculus
  - Developed by CEA, the French nuclear research center
  - Abstract Interpretation based; analysis of C or assembly code
  - First usage on A380
  - DO178B level A
  - Safe computation of the numerical (rounding) errors introduced by basic operators or input filtering code

- Caveat for Unit Proving
  - Developed by CEA, the French nuclear research center
  - Deductive method; analysis of C code; Weakest Precondition
  - First usage on an A380 30.000 loc program
  - DO178B Level A
  - Verification of Low Level Requirements in replacement of Unit Testing
Caveat (1)

• 1996 – 1997 (EU funding): First – limited – prototype
• 1998: since 1998 Airbus is financially associated to the development (50% of dev. costs); the work being done by CEA
• 1998 – 2001:
  ‣ maturation of Caveat for a complete adaptation to the targeted application
  ‣ Experiments on real code from various applications
  ‣ Special work on the previous generation of the targeted application for being sure that:
    – a standard software engineer can use Caveat; answer: YES
    – Caveat runs on standard workstation; answer: YES
    – Caveat is suitable for the targeted application, i.e., Unit Proving on an A380 code; answer: YES
• Since 2001: the tool is used by the development team
  ‣ First, fine tuning of the tool/method
  ‣ Then: effective usage
Caveat (2)

- First Order Predicates (1)
  - Boolean operators
    - ¬, ∧, ∨, ⇒, ⇔
    - Examples: A ∧ B ⇒ C ; D ∨ ¬E ⇔ F ;
  - Relational operators
    - ≤, <, >, ≥, =, ≠
    - they act as Boolean functions
    - Exemples: a ≤ 8 ; b > c
Caveat (3)

- Predicates (2)
  - Arithmetic expressions
    - +, -, *, /, **;
    - Example: x+y-8*z;
  - Quantifiers
    - ∀, ∃
    - They act as Boolean functions
    - Examples: ∀x ∈ int. x ≥ 0 ∧ x < 10 ⇒ P(x);
      ∃ y ∈ int. y ≥ 0 ∧ y < 10 ∧ P(y);
Caveat (4)

• Predicates (3)

  ‣ Examples of predicates:
  
    • \(_{B\_ACS1}=1 \iff \_CPT\_ACS1 \geq 3 \land (VACS1'>-9.6 \lor VACS1'<-10.2)\);
    • \(a \leq x+y-8 \land b > 0 \Rightarrow z=9;\)
Caveat (5)

• Caveat’s Main function
  ▶ Proving that a property holds at a certain point of C function

• How?
  ▶ Caveat computes the condition which must hold at the beginning of the function for proving that the required property holds; it then tries to prove this condition
  ▶ Different kinds of properties to be proved:
    – Post-conditions, Asserts, loop invariants, loop variants
• Example

```c
int Max(int a, int b, int c) {
    int Max;
    if (a >= b)
        if (a >= c)
            Max=a;
        else
            Max=c;
    else
        Max=c;
    else if (b >=c)
        Max=b;
    else
        Max=c;
    return Max;
}
```

Condition of verification 1 : VRAI
Condition of verification 2 : VRAI

Post P1 : Max  =  a ∨ Max = b ∨ Max = c ;
Post P2 : Max  ≥ a ∧ Max ≥ b ∧ Max ≥ c ;
Caveat (8)

- Simplifier
  - Simplification of algebraic expressions

- Theorem prover
  - Notion of Inference rules
APPLICATION TO THE A380 COMPUTER

• Objectives
  ‣ Reduce the cost of software verification without compromising the effectiveness of the verification

• Solution selected
  ‣ Definition of low-level requirements (LLR) using formal properties (design)
  ‣ Unit proof verification supported by the CAVEAT tool, which replaces the unit test activity
APPLICATION TO THE A380 COMPUTER
Formal methods and development cycle

Subset Specification
- Specific Software Design Standard (SDS_CAV)
- Drafting of LLR using the properties
- Definition of the unit proof plan using properties
- Re-reading of the Design based on properties

Design
- Definition of the Unit Proof environment and complements to the properties (MRP)
- Verification of compliance of C-source to properties

Coding
- Designer / Coder
- Designer / Prover
- Independence

Unit Proofs
- Automatic verification of the data/control flow
- Verification of compliance of C-source to properties

Integration
APPLICATION TO THE A380 COMPUTER Certification aspects

- Assessment process of Caveat by Certification Authorities is currently running
- Issue: for the verification process, DO-178B is prescriptive in term of means (Testing)
- The caveat approach is an alternate means of compliance but C.A assess a process against DO-178B
- Need to go back to the underlying objectives and to show that our approach is compliant with them
- A dedicated Certification Review Item is open to record the ways of addressing the issue
- No adverse comment from C.A against our approach