# Software Systems Verification and Validation

## Lecture 07 - Correctness - Floyd, Hoare

Lect. dr. Andreea Vescan

Babeș-Bolyai University
Cluj-Napoca

2015-2016

# Software quality assessment

- Software quality: Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software. [Pre00]
  - correctness - the extent to which a program conforms to its specification.

# Program verification

- Program verification
  - proof-based, computer-assisted, program-verification approach, mainly used for programs which we expect to terminate and produce a result
  - model-based, automatic, property-verification approach, mainly used for concurrent, reactive systems (originally used in a post-development stage) - model checking

# Program verification methods

- Floyd Method - Inductive assertions
- Hoare - Semantics of Hoare triples
- Dijkstra's Language- Guarded commands, Nondeterminacy and Formal Derivation of Programs

# Floyd Method - Inductive assertions [Flo67]

- Input: The condition satisfied by the initial values of the program.
- Output: The condition to be satisfied by the output of the program.
- Method: Steps:
  1. Cut the loops
  2. Find an appropriate set of inductive assertions.
  3. Construct the verification/termination conditions.

# Partial correctness

- Method:
  1. Cut the loops.
  2. Find an appropriate set of inductive assertions.
  3. Construct the verification conditions.
- Theorem: If all verification conditions are true, then the program is partially correct, i.e., whenever it terminates the result is correct.
- The method is useful when it is combined with termination.

# Steps - Partial correctness

- Cutting points are chosen inside the algorithm
  1. 1 point at the beginning of the algorithm, 1 point at the end;
  2. At least 1 point for each *loop* statement
- For each cutting point an assertion (invariant predicate) is chosen.
  1. Entry point - $\varphi(X)$;
  2. Ending point - $\psi(X, Z)$.
- Construction of the verification conditions
  1. Path from $i$ to $j$ - $\alpha$;
  2. $P_i$ and $P_j$ are assertions in $i$ and $j$;
  3. $R_\alpha(X, Y)$ - predicate that gives the condition for path $\alpha$;
  4. $r_\alpha(X, Y)$ - function that gives the transformations of the variables $Y$ from path $\alpha$;
  5. $\forall X \forall Y (P_i(X, Y) \wedge R_\alpha(X, Y) \rightarrow P_j(X, r_\alpha(X, Y)))$.
- Theorem: If all the verification conditions are true then $P$ is

# Example - Partial correctness

- Algorithm for $z = x^y$

  $z := 1;\ \ u := x;\ \ v := y;$
  While $(v\ >\ 0)$ execute
  If ($v$ is even)
  then $u := u * u;\ \ v := v/2;$
  else $v := v - 1;\ \ z := z * u;$
  endIf
  endWhile
  endAlg;

# Example - Partial correctness

- Algorithm for $z = x^y$

  $z := 1;\ u := x;\ v := y;$ A: $\varphi(X) ::= (v > 0 \land (y \geq 0))$

  While $(v\ >\ 0)$ execute

  If ($v$ is even)

  then $u := u * u;\ v := v/2;$

  else $v := v - 1;\ z := z * u;$

  endIf

  endWhile

  endAlg; C: $\psi(X, Z) ::= z = x^y$

# Example - Partial correctness

- Algorithm for $z = x^y$

$z := 1;\ u := x;\ v := y;$ A: $\varphi(X) ::= (v > 0 \wedge (y \geq 0))$
While $(v > 0)$ execute B: $\eta(X, Y) ::= z * u^v = x^y$
If ($v$ is even)
then $u := u * u;\ v := v/2;$
else $v := v - 1;\ z := z * u;$
endIf
endWhile
endAlg; C: $\psi(X, Z) ::= z = x^y$

# Termination

- Method:
  1. Cut the loops and find "good" inductive assertions.
  2. Choose a well-formed set M (i.e., an ordered set without infinite strictly decreasing sequences) and a "good" partial function mapping program variables in M. ("Good" means, if the assertions in a state are true, then the function is defined.)
  3. Show the termination condition hold, i.e., the function strictly decreases at each loop.
- Theorem: If all termination conditions are true, then the program terminates.

# Steps - Termination

- Steps:
  1. Well-ordered set $M$ - partial ordered and doesn't have an infinite decreasing sequence.
  2. To demonstrate that some termination conditions hold: passing from one cutting point to another the values of some functions in the well-ordered set decrease.
  3. In point $i$ a function is chosen $u_i : D_X \times D_Y \to M$ and the termination condition on $\alpha$ is:
     $\forall X \forall Y (\varphi(X) \wedge R_\alpha(X,Y) \to (u_i(X,Y) \; > \; u_j(X, \; r_\alpha(X,Y)))).$
  4. If partial correctness was demonstrated then the termination condition can be:
     $\forall X \forall Y (P_i(X) \wedge R_\alpha(X,Y) \to (u_i(X,Y) \; > \; u_j(X, \; r_\alpha(X,Y)))).$
- Theorem: If all the termination conditions hold then the program $P$ terminates.

# Hoare triples [Hoa69]

- The meaning of a statement is described by a triple
  - $\{\varphi\}\ P\ \{\psi\}$, where $\varphi$ is called the precondition and $\psi$ is called the postcondition.
  - Informal Meaning: “If the program P is run in a state that satisfies $\varphi$, then the state after its execution will satisfy $\psi$ ”
- Note the caveat: “all terminating executions of”
- If P does not terminate, we make no guarantees.
- This is called the partial correctness property

# Semantics of Hoare triples

- Partial correctness
  - $\models_{par} \{\varphi\}P\{\psi\}$
  - only if P actually terminates.
- Total correctness
  - $\models_{tot} \{\varphi\}P\{\psi\}$
  - the program P is guaranteed to terminate.

# Hoare rules - Assignment

- Consider the triple $\{P\}X := Y + 2\{Q\}$
  - Given predicate Q, for what predicate P does this hold?
  - for any P such that $[P \Rightarrow \langle X \leftarrow Y + 2\rangle\,(Q)]$
- Examples
  - $\{P_0\}\ X := Y + 2\ \{X \leq Y + 2\}$
    $P_0 \equiv true$
  - $\{P_1\}\ X := Y + 2\ \{X < 0\}$
    $P_1 \equiv (Y + 2 < 0)$
  - $\{P_2\}\ X := Y + 2\ \{Y < 0\}$
    $P_2 \equiv (Y < 0)$
  - $\{P_3\}\ X := X + 2\ \{X\ is\ even\}$
    $P_3 \equiv (X\ is\ even)$
- General Form: for any expression E
  - $\{P\}\ X := E\{Q\}\ provided[P \Rightarrow \langle X \leftarrow E\rangle\,(Q)]$

# Hoare rules - Sequencing

- We can conclude
  $\{P\}\ S;T\{Q\}$
  if we can find a predicate R such that
  $\{P\}\ S\{R\}$ *and* $\{R\}T\{Q\}$
- Examples
  - $\{P_0\}\ X := 2 * X;\ \ X := X + 1\{X > 0\}$
    $P_0 \equiv (2 * X + 1 > 0)]$
  - $\{P_1\}\ X := Y;\ \ Y := 3\ \{X + Y < 5\}$
    $\{P_1 \equiv (Y + 3 < 5)]$

# Hoare rules - Conditionals

- We can conclude
  $\{P\}$ $IF\ (C)\ THEN\ S\ ELSE\ T\ END\{Q\}$
  provided we can show
  $\{P \wedge C\}$ $S\{Q\}$ *and* $\{P \wedge \neg C\}T\{Q\}$
- Examples
  - $\{?\}$ $\{((x > y) \Rightarrow Q_0) \wedge ((x \leq y) \Rightarrow Q_1)\}$
    $IF\ (x > y)\ THEN$ $\quad Q_0 : \{(m|x - y) \wedge (m|y)\}$
    $x := x - y$
    $ELSE$ $Q_1 : \{(m|x) \wedge (m|y - x)\}$
    $y := y - x$
    $END$
    $Q : \{(m|x) \wedge (m|y)\}$
  - So our final proof obligations are
    $[(x > y) \Rightarrow (m|x - y) \wedge (m|y)$ *and*
    $[(x \leq y) \Rightarrow (m|x) \wedge (m|y - x)]$

# Hoare rules - Example

- Mystery Program
  - ?

$$x := x + y;$$
$$y := x - y$$
$$x := x - y$$

# Hoare rules - Example

- Mystery Program
  - ?

$x := x + y;$
$y := x - y$
$x := x - y$
$(x = A) \wedge (y = B)$

# Hoare rules - Example

- Mystery Program
  - ?

$$x := x + y;$$
$$y := x - y$$
$$x := x - y \ \{(x - y = A) \wedge (y = B)\}$$
$$(x = A) \wedge (y = B)$$

# Hoare rules - Example

- Mystery Program
  - ?

$$
\begin{aligned}
&x := x + y;\\
&y := x - y \ \{(x - (x - y) = A) \wedge (x - y = B)\}\\
&x := x - y \ \{(x - y = A) \wedge (y = B)\}\\
&(x = A) \wedge (y = B)
\end{aligned}
$$

# Hoare rules - Example

- Mystery Program
  - ?

$$x := x + y;\ \{(y = A) \wedge (x - y = B)\}$$
$$y := x - y\ \{(x - (x - y) = A) \wedge (x - y = B)\}$$
$$x := x - y\ \{(x - y = A) \wedge (y = B)\}$$
$$(x = A) \wedge (y = B)$$

# Hoare rules - Example

- Mystery Program
  - ?
    $\{(y = A) \wedge ((x + y) - y = B)\}$

    $x := x + y;\ \{(y = A) \wedge (x - y = B)\}$
    $y := x - y\ \{(x - (x - y) = A) \wedge (x - y = B)\}$
    $x := x - y\ \{(x - y = A) \wedge (y = B)\}$
    $(x = A) \wedge (y = B)$

# Hoare rules - Example

- Mystery Program
  - ?

    $\{(y = A) \wedge ((x + y) - y = B)\}$

    $\{(y = A) \wedge (x = B)\}$

    $x := x + y;\ \{(y = A) \wedge (x - y = B)\}$

    $y := x - y\ \{(x - (x - y) = A) \wedge (x - y = B)\}$

    $x := x - y\ \{(x - y = A) \wedge (y = B)\}$

    $(x = A) \wedge (y = B)$

# Hoare rules - Example

- Mystery Program
  - ?
    $\{(y = A) \wedge ((x + y) - y = B)\}$
    $\{(y = A) \wedge (x = B)\}$
    $x := x + y;\ \{(y = A) \wedge (x - y = B)\}$
    $y := x - y\ \{(x - (x - y) = A) \wedge (x - y = B)\}$
    $x := x - y\ \{(x - y = A) \wedge (y = B)\}$
    $(x = A) \wedge (y = B)$
- Thus this program swaps the values of x, y.

# Hoare rules - Reasoning About Loops

- How can we conclude
  $\{P\}$ *WHILE* $(G)$ *DO S END* $\{Q\}$
  At the end of the loop (assuming it terminates), we know $\neg G$
  But in general we dont know how often S is executed...
- Suppose we have a predicate J that is preserved by S
  $\{J\}S\{J\}$ such a J is called a loop invariant
  Then, at the end of the loop, we can conclude
  $J \wedge \neg G$
  To establish the postcondition, we need J such that
  $[J \wedge \neg G \Rightarrow Q]$

# Hoare rules - Loops

- We can conclude
  $\{P\}$ *WHILE* $(G)$ *DO* $S$ *END* $\{Q\}$
  provided we can find a loop invariant J such that

| | |
|---|---|
| $[P \Rightarrow J]$ | J holds at loop entry |
| $[J \wedge \neg G \Rightarrow Q]$ | J establishes Q at loop exit |
| $\{G \wedge J\}S\{J\}$ | J is preserved by each iteration |

# Hoare rules - Loop Example

- Loop Example
  - $\{N \geq 0\}$

    $m := 0;\ \ y := 1;$

    $WHILE\ (m! = N)\ DO$
    $y := 2 * y;$
    $m := m + 1$
    $END$

# Hoare rules - Loop Example

- Loop Example
  - $\{N \geq 0\}$
    $m := 0; \;\; y := 1;$

    $WHILE \; (m! = N) \; DO$
    $y := 2 * y;$
    $m := m + 1$
    $END$
    $\{y = 2^N\}$

# Hoare rules - Loop Example

- Loop Example
  - $\{N \geq 0\}$
    $m := 0; \ y := 1;$

    $WHILE\ (m! = N)\ DO\ J : \{y = 2^m\}$
    $y := 2 * y;$
    $m := m + 1$
    $END$
    $\{y = 2^N\}$
- Need to show that invariant
  - holds initially
  - is preserved by loop body $\{J\}\ y := 2 * y;\ m := m + 1\ \{J\}$
  - establishes postcondition $[J \wedge (m = N) \Rightarrow (y = 2^N)]$.

# Hoare rules - Loop Example

- Loop Example
  - $\{N \geq 0\}$
    $m := 0;\ \ y := 1;$
    $\{y = 2^m\}$
    $WHILE\ (m! = N)\ DO\ J : \{y = 2^m\}$
    $y := 2 * y;$
    $m := m + 1$
    $END$
    $\{y = 2^N\}$
- Need to show that invariant
  - holds initially
  - is preserved by loop body $\{J\}\ y := 2 * y;\ \ m := m + 1\ \{J\}$
  - establishes postcondition $[J \wedge (m = N) \Rightarrow (y = 2^N)]$.

# Hoare rules - Loop Example

- Multiplication using addition
  - Precondition $B \geq 0$
  - Postcondition $R = A * B \Rightarrow \{B \geq 0\}\ S\{R = A * B\}$
  - Solution:

    $\{B \geq 0\}$
    "initialize R"
    WHILE (G) DO
    "update R"
    END
    $\{R = A * B\}$

# Hoare rules - Loop Example

- Multiplication using addition
  - Precondition $B \geq 0$
  - Postcondition $R = A * B \Rightarrow \{B \geq 0\}\ S\{R = A * B\}$
  - Solution:
    $\{B \geq 0\}$
    "initialize R"
    WHILE (G) DO
    "update R"
    END
    $\{R = A * B\}$
- Rule: replace constant with variable in postcondition Q to obtain a loop invariant J such that $[(J \wedge \neg G) \Rightarrow Q]$
  - Introduce variable b and add the invariant J defined as $R = A * b$
  - To ensure postcondition, choose G to be $(b \neq B)$ since

# Hoare rules - Loop Example (2)

- Multiplication using addition (2)
  - the loop invariant will be $(R = A * b)$ and the loop guard will be $(b \neq B)$
  - To ensure that invariant holds initially, we initialize with $R := 0; b := 0$

# Hoare rules - Loop Example (2)

- Multiplication using addition (2)
  - the loop invariant will be $(R = A * b)$ and the loop guard will be $(b \neq B)$
  - To ensure that invariant holds initially, we initialize with $R := 0; b := 0$
  - In each iteration, we increase b by 1 , giving

    $\{B \geq 0\}$
    $R := 0; b := 0;$
    WHILE $((b \neq B)$ DO $J: \ \{R = A * b\}$

    $R :=? \ \Rightarrow R := R + A$
    $b := b + 1$
    END
    $\{R = A * B\}$

# Hoare rules - Loop Example

- Exponentiation using multiplication
  - $\{(A > 0) \wedge (B \geq 0)\}$ S $\{R = A^B\}$

# Hoare rules - Loop Example

- Exponentiation using multiplication
  - $\{(A > 0) \wedge (B \geq 0)\}$ S $\{R = A^B\}$
  - Solution: Again, we replace a constant with a variable use loop invariant $J : R = A^b$
    $\{(A > 0) \wedge (B \geq 0)\}$
    $R$ :=?; $b := 0$ R:=1
    WHILE $(b \neq B)$ DO $J : R = A^b$
    $R$ :=?; $R := R * A$;
    $b := b + 1$
    END
    $\{R = A^B\}$

# Hoare - Partial and Total correctness

- Recall that we interpreted $\{P\}S\{Q\}$ as
  "when started in a state satisfying P, any terminating execution of S ends in a state satisfying Q "
- The "total correctness" interpretation also requires termination
  "when started in a state satisfying P, any execution of S must terminate in a state satisfying Q "
- Informally
  - proof of total correctness =
    proof of partial correctness
    + proof of termination

# Hoare Rules - Total correctness

- Assignment
  $\{P\}\ X := E\ \{Q\}$*provided*$[P \Rightarrow \langle X \leftarrow E\rangle(Q)]$
- Sequencing
  $\{P\}\ S; T\{Q\}$ provided
  $\{P\}\ S\ \{R\}$ *and* $\{R\}T\ \{Q\}$ for some R
- Conditional
  $\{P\}\ IF\ (G)\ THEN\ S\ ELSE\ T\ END\ \{Q\}$ provided
  $\{P \land G\}\ S\ \{Q\}$ *and* $\{P \land \neg G\}\ T\{Q\}$
- Note: Same as the rules for partial correctness!

# Hoare Rules - Total correctness

- Total correctness rule for loops
- Consider
  $\{P\}$ *WHILE* $(G)$ *DO* $S$ *END* $\{Q\}$
- How do we show that the loop terminates?
- One method
  find an integer expression V such that
  the value of V is nonnegative (that is $V \geq 0$ ), and
  the value of V (strictly) decreases in every iteration that is,
  $\{V = K\}$ $S$ $\{V < K\}$
- Such an expression is called a "loop variant"

# Hoare Rules - Total correctness - loop

- Exponentiation using multiplication
  - $\{(A > 0) \wedge (B \geq 0)\}$ S $\{R = A^B\}$
  - Recall loop invariant $J : R = A^b \ \wedge (B \geq b);$

    $\{(A > 0) \wedge (B \geq 0)\}$
    $R := 1; b := 0$
    WHILE $(b \neq B)$ DO $\quad J : \ R = A^b \wedge (B \geq b);$
    $R := R * A;$
    $b := b + 1$
    END
    $\{R = A^B\}$
- We define loop variant V to be the expression $(B - b)$
- Note: V strictly decreases with every loop iteration, because $[(B - (b + 1)) < (B - b)]$
- How do we show that V is non-negative?
  by showing that $(B \geq b)$ is a loop invariant

# Summary: Total Correctness Rule for Loops

- To show $\{P\}$ $WHILE\ (G)\ DO\ S\ END$ $\{Q\}$ we find a loop invariant predicate J and a loop variant expression V such that
  - J holds initially $[P \Rightarrow J]$
  - J establishes the postcondition upon exit $[(J \wedge \neg G) \Rightarrow Q]$
  - is preserved by loop body $\{J\}\ S\ \{J\}$
  - variant V strictly decreases in every iteration $\{V = K\}$ S $\{V < K\}$
  - variant V is always non-negative $[J \Rightarrow (V \geq 0)]$

# Next lecture

- Correctness - Dijkstra
- Static analysis - ESC/Java tool - Topic of Laboratory 6!

# References I

[Flo67] Robert W. Floyd.
Assigning meanings to programs.
In *Proceedings of Symposia in Applied Mathematics*, pages 19–32, 1967.

[Hoa69] C. A. R. Hoare.
An axiomatic basis for computer programming.
*Commun. ACM*, 12(10):576–580, 1969.

[Pre00] Roger S. Pressman.
*Software Engineering: A Practitioner's Approach*.
McGraw-Hill, Inc., 2000.