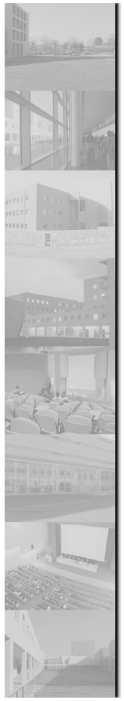

Métodos Formais em Engenharia de Software

Ana Paiva

apaiva@fe.up.pt

Universidade do Porto
Faculdade de Engenharia
FEUP

Proof of theorems

- A formal logical system consists of:
  - Notation (syntax).
  - A set of axioms.
  - A set of inference rules.
  - A formal proof is a sequence of statements. Each statement is built from the application of one or more rules of inference to the precedent statement(s).
  - A purely syntactic mechanism that does not worry about the meaning of the claims but only to their construction.

Proof of theorems

- Prove that an implementation (I) satisfies the specification (S) by mathematical reasoning.

$$I \rightarrow S$$
$$I \equiv S$$

- The implementation and specification are expressed by logical formulas.
- The (logical equivalence / logical implication) required is described as a theorem that has to be proven.
- A proof system provides a set of axioms and inference rules (simplification, rewriting, induction, etc.)

Hoare logic

- Simple logic to describe (and prove) programs. Set of axioms and inference rules that define the semantics of programs.
- What we want to prove is a Hoare triple:

  {P} S {Q}
  - P - pre-condition
  - S - program
  - Q - post-condition
- It is a logical expression that says: If the program runs S from an initial state satisfying the precondition P, then, on completion of the final state satisfies the postcondition Q
- Simply put: the program S establishes the postcondition Q from the pre-condition P
- Relates program with specification
- The specifications of the types of data input and output should be viewed also as pre-and post-conditions relevant to the test!

Examples

{ true } x := 12 { x = 12 }

{x < 40} x := 12 {10 <= x}

{x < 40} x := x+1 {x <= 40}

True for integers but not for reals

{m <= n} j :=(m+n)/2 {m <= j <= n}

{0 <= m < n <= a.length /\ a[m] = x} r := Find(a, m, n, x) {m <= r}

{false} S {Q}

True, for any program S-and post-codition Q, since false implies anything but useless

{P} S {false}

false if there is at least an initial state where P is true, but unrealizable

Triple accurate

- There are many Hoare triple {P} S {Q} true for the same program, normally interest us the most accurate that is, those that use the weakest precondition or strongest postcondition
- wp(S, Q) - Weakest precondition of S related with Q
  - It is the more general condition such that {P} S {Q}
  - {P} S {Q} iff P wp (S, Q)
  - Suggested method of proof of {P} S {Q} in the opposite direction: **(1)** calculate wp (S, Q) and **(2)** prove that P $\Rightarrow$ wp (S, Q)
  - Example: wp (x: = x +1, x> 10) = x> 9
{x=20} x := x = x +1 {x>10} is true for x = 20 $\Rightarrow$ x>9
- sp(P, S) - Strongest postcondition of S in relation to P
  - {P} S {Q} sse sp(P, S) $\Rightarrow$ Q
  - Less used than the weakest precondition

Total and partial correction

- **Partial correctness**: if the program ends, the final state satisfies the postcondition
- **The total correction**: the program ends in a final state satisfying the postcondition

- What interests us is the total correction
- Non-termination problem can arise with cycles

Rules of the Hoare logic

- And now, how do we prove?
- Using the axioms and inference rules of Hoare logic, which describe the semantics of the instructions used and other useful properties in proofs
- Inference rules are presented as

$$\frac{\textbf{premies}}{\textbf{conclusions}}$$

- Axioms are rules without premises (even if true)

Regras de inferência básicas

Nº	Instruction	Rule	Notes
R1	skip	{P} skip {P}	
R2	Assignment	{P[x:=E]} x := E {P}	(a)
R3	Sequence	$\dfrac{\{P\}\ S\ \{Q\}\ ,\quad \{Q\}\ T\ \{R\}}{\{P\}\ S;\ T\ \{R\}}$	
R4	If	$\dfrac{\{P \land C\}\ S\ \{Q\}\ ,\quad \{P \land \neg C\}\ T\ \{Q\}}{\{P\}\ \text{if } C \text{ then } S \text{ else } T\ \{Q\}}$	
R5	Cycle	$\dfrac{I \land C \Rightarrow v \in N,\ \ \{I \land C \land v=V\}\ S\ \{I \land v<V\}}{\{I\}\ \text{while } C \text{ do } S\ \{I \land \neg C\}}$	(b) (c)

(a) P[x:=E] - P with x replaced by E
(b) I - invariant of the cycle (actually before and after each iteration)
(c) v - function variant, non-negative integer strictly decreasing, to ensure termination

Regras de inferência adicionais

Nº	Description	Rule	Notes
R6	Strengthening the precondition	$\dfrac{P' \Rightarrow P,\quad \{P\}\ S\ \{Q\}}{\{P'\}\ S\ \{Q\}}$	Limit case: P = wp(S,Q) (weakest precondition)
R7	Weakening the postcondition	$\dfrac{\{P\}\ S\ \{Q\},\quad Q \Rightarrow Q'}{\{P\}\ S\ \{Q'\}}$	Limit case: Q = sp(P,S) (strongest postcondition)
R8	Intermediate assertions	{P∧A} assert A {P}	wp(assert A, P) = P∧A

Proof of precondition to the postcondition

To prove:

{P}
S_1;
S_2;
...
S_n;
{Q}

Calculate successively the strongest post-conditions:

{P}
1º S_1;
$\{Q_1\}$
2º S_2;
$\{Q_2\}$
...
...
$\{Q_{n-1}\}$
nº S_n;
$\{Q_n\}$

And then we must prove:

$Q_n \Rightarrow Q$

(correct by rules R3 and R7)

Reasoning to discover the strongest postcondition

{P} S {?}

If P is true before executing S, what can we say that is true after executing S?

Example:

{x > y} x := x+1 {?} → {x > y} x := x+1 {x-1 > y}

(se x > y before executing x := x+1, then, after executing x := x+1, we can state that x-1 > y)

Intuition can be confirmed with the rules of Hoare logic

Example:

{?} x := x+1 {x-1 > y}
⇒ {(x+1)-1>y} x := x+1 {x-1 > y} (by rule R2)
⇔ {x>y} x := x+1 {x-1 > y}

Proof of postcondition to a precondition

To prove:

$\{P\}$
$S_1;$
...
$S_{n-1};$
$S_n;$
$\{Q\}$

Calculate successively the weakest preconditions:

$\{P_1\}$
n° $S_1;$
$\{P_2\}$
...
$\{P_{n-1}\}$
2° $S_{n-1};$
$\{P_n\}$
1° $S_n;$
$\{Q\}$

And then we must prove:

$P \Rightarrow P_1$

It may be easier because some rules of Hoare logic work in this direction!

Reasoning to find the weakest precondition

{?} S {Q}

Q to be true after executing S (and to finish S) what must be true before executing S?

Example:

while k<N do s := s + a[k]; k := k+1 end
{s = (Σi | 0<=i<N · a[i])}

- You can also use the Dijkstra rules in the following slides

Rules for calculating the weakest precondition (Dijkstra)

Nº	Instrução	Regra
R1'	skip	wp(skip, P) = P
R2'	Assignment	wp(x := E, Q) = Q[x:=E]
R3'	Sequence	wp(S;T, Q) = wp(S, wp(T, Q))
R4'	If	wp(if C then S else T, Q) = $C \wedge$ wp(S,Q) $\vee$ $\neg C \wedge$ wp(T,Q)
R5'	Cycle	wp(while C do S, Q) = $P_0 \vee P_1 \vee \ldots$, com $P_0 = \neg C \wedge Q$, $P_k = C \wedge wp(S, P_{k-1})$, $k>0$
R8'	Assertion	wp(assert A, Q) = $A \wedge Q$

(Exercise: figure out the rules to calculate the strongest postcondition)

Verification of intermediate assertions

To prove

{P} assert A {Q}
prove that
$P \Rightarrow A$
and
$P \Rightarrow Q$

(by rule 8)

Since wp(assert A; Q) = $A \wedge Q$

prove that $P \Rightarrow A \wedge Q$

Verification of conditional statements

To prove

{P} if C then S else T {Q}
it is sufficient to prove that
{P ^ C} S {Q} (when C is true)
and
{P ^ ¬C} T {Q} (when C is false)
(by rule 4)

Schematically

if C then
{P ^ C} S {Q}
else
{P ^ ¬C} T {Q}

Examples

Sequence: {??} x = z+1; y = x+y; {y > 5}

wp(y = x+y;, y > 5) = x+y > 5
wp(x = z+1;, x+y > 5) = z+1+y > 5
?? é
z+1+y > 5

If without *else*: {??} if(x > y) y = x; {y = max(x, y)}

[x>y /\ wp(y=x,y=max(x,y))] \/ [x<=y /\ y=max(x,y)]
x>y /\ x=max(x,x) \/ True
True