innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: **PROGRAM VERIFICATION**

Floyd-Hoare Logic: Verifying Loops: Loop Invariants

Hoare Logic – Basic Method

- Hoare Logic reduces basic correctness argument for a program

  /* Precondition: Input x satisfies some properties */

  A(x)

  /* Postcondition: Running A on x results in ... */

- to each statement in a program:

  /* p0 */

  S1

  /* p1 */

  S2

  /* p2 * /

  ...

  /* pN */

such that pN is the required post-condition and p0 is the given precondition.

- How would this approach work for iterative statements (i.e. loops)?

- How many iterations will one go through?

Hoare Logic – Rule for Iterations – Invariants

- Rule for Iterative Statement

$$\frac{\langle \phi \wedge B,\ S,\ \phi \rangle}{\langle \phi,\ \text{while } B \text{ do } \{ S \},\ \phi \wedge \neg B \rangle}$$

This premise states that ϕ remains *invariant* (i.e. unchanged) over one iteration.

- Alternatively

/* Precondition: ϕ */

while (B) do { /* $\phi \wedge B$ */ S /* ϕ */ }

/* Postcondition: $\phi \wedge \neg B$ */

Therefore, *by induction*, ϕ remains *invariant over any number of iterations.*

Hoare Logic – Iterations - Example

```
/* Pre-condition:
    gcd(x,y) = gcd(A,B)*/
while (y != 0) do {
    /* gcd(x,y) = gcd(A,B) ∧ ¬(y = 0)
*/
       t = x % y;
       x = y;
       y = t;
       /* gcd(x,y) = gcd(A,B)*/
}
/* Post-condition:
  gcd(x,y) = gcd(A, B) ∧ y=0 */
```

Hoare Logic – Invariant - Example

```
/* Pre-condition:
    gcd(x,y) = gcd(A,B)*/
while (y != 0) do {
    /* gcd(x,y)=gcd(A,B) ∧ ¬(y = 0)
*/
      t = x % y;
      x = y;
      y = t;
      /* gcd(x,y) = gcd(A,B)*/
}
/* Post-condition:
  gcd(x,y) = gcd(A, B) ∧ y=0 */
```

```
/* gcd(y,x%y) = gcd(A,B)*/
t = x % y;
    /* gcd(y,t) = gcd(A,B)*/
x = y;
    /* gcd(x,t) = gcd(A,B)*/
y = t;
    /* gcd(x,y) = gcd(A,B)*/
```

Hoare Logic – Invariant - Example

```
/* Pre-condition:
    gcd(x,y) = gcd(A,B)*/
while (y != 0) do {
    /* gcd(x,y)=gcd(A,B) ∧ ¬(y = 0) */
      t = x % y;
       x = y;
      y = t;
       /* gcd(x,y) = gcd(A,B)*/
}
/* Post-condition:
   gcd(x,y) = gcd(A, B) ∧ y=0 */
```

/* gcd(x,y)=gcd(A,B) ∧ ¬(y = 0) */

because
gcd(x,y) = gcd(x-y,y)

/*gcd(x%y,y) = gcd(A,B) */

because
gcd is ***commutative***

```
/* gcd(y,x%y) = gcd(A,B)*/
t = x % y;
      /* gcd(y,t) = gcd(A,B)*/
x = y;
      /* gcd(x,t) = gcd(A,B)*/
y = t;
      /* gcd(x,y) = gcd(A,B)*/
```

This remains invariant!

Floyd-Hoare Logic: Loop Invariants

- A ***loop invariant*** is a condition that
  - is true before the loop statement (i.e. *is the pre-condition*)
  - remains invariant over one (arbitrary) iteration of the loop
  - is true after the loop statement (i.e. *is the post-condition*)

- If the condition is invariant over one (arbitrary) iteration,
  - then the proof (of correctness of the statement) is
    - <u>*not dependent on the number of iterations*</u> of the loop

Floyd-Hoare Logic: Correctness of Loops

- Given the Hoare-triple
  - < ϕ, while B do S, ψ>,
- the proof (i.e. the verification) for partial correctness proceeds as follows:
  - guess a loop invariant condition, say, ι
  - and verify the following:
    - $|\!-\ \phi \wedge B \dashrightarrow \iota$
    - $< \iota \wedge B, S, \iota >$
    - $|\!-\ \iota \wedge \neg B \dashrightarrow \psi$