# CSE 321 Discrete Structures

January 27, 2010

Lecture 10: Program Correctness

# Program Verification

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

What are x and y at the end ??

# Partial Correctness

Hoare triple:

$$p \ \{S\} \ q$$

A program (or program segment) S is *partially correct* w.r.t. the precondition p and postcondition q, if:

whenever p is true before the execution then q is true after executing S

# Program Verification

$$x = x_0 \land y = y_0$$

$$\{\ x = y - x;$$

$$y = y - x;$$

$$x = y + x;$$

$$\}$$

$$x = y_0 \land y = x_0$$

How do we prove this ?

# Rules of inference

- Assignment:
  - q results from p by substituting x with expression

$$p\ \{x = \text{Expression};\}\ q$$

- Composition:

$$\frac{p\ \{S1\}\ q; \quad q\ \{S2\}\ r}{p\ \{S1;S2\}\ r}$$

- Conditional:

$$\frac{(p \wedge C)\ \{S1\} q; \quad (p \wedge \neg C)\ \{S2\}\ q;}{p\ \{\ \textbf{if}\ (C)\ S1;\ \textbf{else}\ S2;\ \}\ r}$$

- Loops:

$$\frac{(p \wedge C) \quad \{S\} \quad p \qquad .}{p \quad \{\ \textbf{while}\ (C)\ S;\ \}\ (\ p \wedge \neg C)}$$

# Assignment and Composition Rules

$$x = x_0 \wedge y = y_0 \quad \{\ x = y - x;\ \} \quad x = y_0 - x_0 \wedge y = y_0;$$

$$x = y_0 - x_0 \wedge y = y_0 \quad \{\ y = y - x;\ \} \quad x = y_0 - x_0 \wedge y = x_0;$$

$$x = y_0 - x_0 \wedge y = x_0; \quad \{\ x = y + x;\ \} \quad x = y_0 \wedge y = x_0;$$

---

$$x = x_0 \wedge y = y_0$$

$$\{\ x = y - x;$$

$$y = y - x;$$

$$x = y + x;$$

$$\}$$

$$x = y_0 \wedge y = x_0$$

# Quiz: What does this do ?

$$
\begin{aligned}
x &= y - x; \\
y &= z - y; \\
z &= z - x; \\
x &= x + z; \\
z &= y - z; \\
y &= x - y;
\end{aligned}
$$

# Conditional: Recall $\vee$- Elimination !

**if** (x > y) z = x;
**else** z = y;

Prove that z = max(x,y)

For this, prove: $x = x_0 \wedge y = y_0 \wedge x_0 > y_0 \rightarrow x_0 = \max(x_0, y_0)$

$$\frac{x > y \quad \{\ z = x;\ \}\ \ z = \max(x,y) \qquad\qquad x \leq y \quad \{\ z = y;\ \}\ \ z = \max(x,y)}{T \quad \{\ \textbf{if}\ (x>y)\ z = x;\ \textbf{else}\ z=y;\ \} \quad z = \max(x,y)}$$

# The Mystery Program

```
{ int x = a; int y = b; int z = 0;
  while (x > 0) {
    if ((x & 1) == 0) { x >>= 1; y <<= 1; }
    else { x--;      z += y; }
  }
  /* what is z ? */
}
```

Note: (x & 1) == 0) means “is x even ?”
x >>= 1 means “x = x / 2”
y <<= 1 means “y = y*2”

# Loop Invariants

```
/* pre condition: a ≥ 0 */
{ int x = a; int y = b; int z = 0;
  { while (x > 0) {
     /* loop invariant:   a*b = x*y + z ∧ x ≥0   */
     if ((x & 1) == 0) { x >>= 1; y <<= 1; }
     else { x--;     z += y; }
 }
/* post condition: z = a*b */
```

It computes the product a*b !
Called "Fast Multiplication". Why "fast" ?

# Loop Invariant

```
a ≥ 0
{ int x = a
  int y = b;
  int z = 0;}
a*b = x*y + z ∧ x ≥0
```

```
a*b = x*y + z  ∧ x > 0
{  if ((x & 1) == 0)
         { x >>= 1; y <<= 1; }
   else { x--;     z += y; }
}
 a*b = x*y + z ∧ x ≥0
```

---

```
a*b = x*y + z ∧ x ≥0
{ while (x > 0) { if ((x & 1) == 0) { x >>= 1; y <<= 1; }
                  else { x--; z += y; }  }
}
 a*b = x*y + z  ∧ x ≥0 ∧ x ≤ 0
```

# Conditional

$x=x_0 \wedge y=y_0 \wedge z=z_0$

$a*b = x*y + z \wedge x > 0 \wedge x$ is even

{ x >>= 1; y <<= 1; }

$a*b = x*y + z \wedge x \geq 0$

$a*b = (x_0/2)*(2*y_0) + z_0$

$a*b = x*y + z \wedge x > 0 \wedge x$ is odd

{ x--;     z += y; }

$a*b = x*y + z \wedge x \geq 0$

$a*b = (x_0-1)*y_0 + z_0 + y_0$

---

$a*b = x*y + z \wedge x > 0$

{ **if** ((x & 1) == 0) { x >>= 1; y <<= 1; }
  **else** { x--;     z += y; }
}

$a*b = x*y + z \wedge x \geq 0$