CS 326
Formal Reasoning

Stephen Freund

Reasoning About Programs

- What is true of a program's state as it executes?
  - Given initial assumption or final goal
- Examples:
  - If x > 0 initially, then y == 0 when loop exits
  - Contents of array are sorted
  - Except at one program point, x + y == z
  - For all instances of Node n,
    n.next == nil ⋁ n.next.prev == n
  - …

Forward Reasoning Example

- Suppose we initially know (or assume) w > 0

```
// w > 0
x = 17;
// w > 0 ∧ x == 17
y = 42;
// w > 0 ∧ x == 17 ∧ y == 42
z = w + x + y;
// w > 0 ∧ x == 17 ∧ y == 42 ∧ z > 59
…
```

Then we know various things after, e.g., z > 59

Backward Reasoning Example

- Suppose we want z < 0 at the end

```
// w + 17 + 42 < 0
x = 17;
// w + x + 42 < 0
y = 42;
// w + x + y < 0
z = w + x + y;
// z < 0
```

Then initially we need w < -59

Forward vs. Backward

- Forward Reasoning
  - Determine what follows from initial assumptions
  - Useful for ensuring an invariant is maintained
- Backward Reasoning
  - Determine sufficient conditions for a certain result
    - Desired result: assumptions need for correctness
    - Undesired result: assumptions needed to trigger bug
  - Less natural but often more useful

Conditional Example (Fwd)

```
// x >= 0
z = 0;
// x >= 0 ∧ z == 0
if (x != 0) {
   // x >= 0 ∧ z == 0 ∧ x != 0        (so x > 0)
   z = x;
   // … ∧ z > 0
} else {
   // x >= 0 ∧ z == 0 ∧ !(x != 0)     (so x == 0)
   z = x + 1;
   // … ∧ z == 1
}
// ( … ∧ z > 0) ∨ (… ∧ z == 1)        (so z > 0)
```

Pre- and PostConditions

Precondition → { w < 59 }

```
x = 17;
```

Postcondition → { x = 17 /\ w + x < -42 }

- An assertion holds if evaluating it in the current state produces true.

Hoare Triples

{ P } S { Q }

- Hoare triple {P} S {Q} is valid iff:
  - For all states where P holds, executing S always produces a state where Q holds

    "If P is true before S, then Q must be true after"

Hoare Triple Examples

- Valid or invalid?
  - Assume all variables are integers without overflow

```
{x != 0} y = x*x;  {y > 0}
{z != 1} y = z*z;  {y != z}
{x >= 0} y = 2*x;  {y > x}
{true} if (x > 7) { y=4; } else { y=3; } {y < 5}
{true} x = y; z = x; {y=z}
{x=7 ∧ y=5} tmp=x;  x=tmp;  y=x;  {y=7 ∧ x=5}
```

Assignment

{ P } x = e; { Q }

- Valid if: P => Q[x:= e]

Replace all occurrences of x with e

- Example: { z > 34 } `y = z + 1;` { y > 1 }
  - Valid: { z > 34 } => { z + 1 > 1 }

Sequence

{ P } S1; S2 { Q }

- Valid if: there is an R such that:
  1. { P } S1 { R }
  2. { R } S2 { Q }
- Example:

{ z ≥ 1 }
```
y = z + 1;
w = y * y;
```
{ w > y }

R is { }

1. { z ≥ 1 } `y = z+1` { R }

2. { R } `w = y*y` { w > y }

Conditional

{ P } if(b) S1 else S2 { Q }

- Valid if: there are Q1, Q2 such that:
  1. { P /\ b } S1 { Q1 }
  2. { P /\ !b } S2 { Q2 }
  3. Q1 \/ Q2 implies Q
- Example:

```
{ true }
if (x > 7)
  y = x;
 else
  y = 20;
{ y > 5 }
```

Q1 = { }. Q2 = { }

1. {true /\ x > 7} `y = x` { Q1 }

2. {true /\ x <= 7} `y = 20` { Q2 }

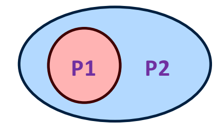
3. Q1 \/ Q2 => y > 5

Conditional

```
{ true }
if (x > 7)
  { true /\ x>7 }
  y = x;
  { y>7 }
 else
  { true /\ x<=7 }
  y = 20;
  { y=20 }
{ y>5 }
```

Weaker vs. Stronger Assertions

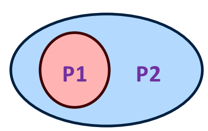
- If P1 => P2 then:
  - P1 is stronger than P2
  - P2 is weaker than P1



- Whenever P1 holds, P2 is guaranteed to hold

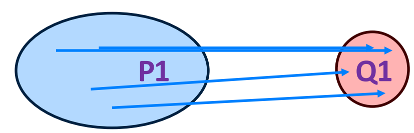
Weaker vs. Stronger Assertions

- If P1 => P2 then:
  - P1 is stronger than P2
  - P2 is weaker than P1



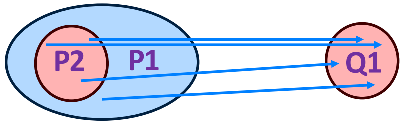
- x > 0 => x >= 0?
- x = 1 => x > 0?
- x > 0 => true?
- x > 0 => x != 1?

Strength and Hoare Logic



Suppose **{P1} S {Q1}**. **{x>0} S {y<5}**

Strength and Hoare Logic

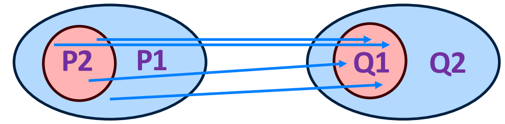


P2 => P1

Suppose {P1} S {Q1}. {x>0} S {y<5}

- {P2} S {Q1} {x>10} S {y<5}

Strength and Hoare Logic

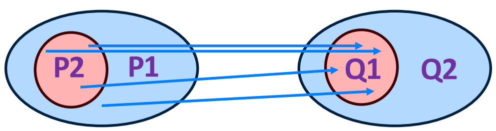


P2 => P1 Q1 => Q2

Suppose {P1} S {Q1}. {x>0} S {y<5}

- {P2} S {Q1} {x>10} S {y<5}
- {P1} S {Q2} {x>0} S {y<10}

Strength and Hoare Logic



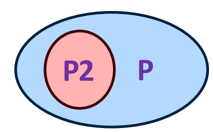
P2 => P1 Q1 => Q2

Suppose {P1} S {Q1}. {x>0} S {y<5}

- {P2} S {Q1} {x>10} S {y<5}
- {P1} S {Q2} {x>0} S {y<10}
- {P2} S {Q2} {x>10} S {y<10}

Backward Reasoning

- Given S and Q, find P such that {P} S {Q}.
- But which P?
  - { x > 0 } y = x*x { y > 0 }
  - { x != 0 } y = x*x { y > 0 }
- Weakest precondition P:
  - {P} S {Q}
  - For all P2, if {P2} S {Q} then P2 => P.
- Most relaxed requirements on program state.



Example

What is the **weakest precondition** P that ensures c is not null?

```
{ P }
Point c = null;
int z;
if (y < 0) {
  z = -2*y;
} else {
  z = x;
}
if (z > 10) {
  c = new Point(z,y);
}
{ Q: c != null }
```

$wp(x = e, Q) \equiv Q[x := e]$

wp(`x = y*y`, $x > 4$)

$\equiv y*y > 4$

$\equiv |y| > 2$

$wp(S1;S2, Q) \equiv wp(S1, wp(S2,Q))$

wp(`y=x+1; z=y+1`, $z > 2$)

$\equiv$ wp(`y=x+1`, wp(`z=y+1`, $z > 2$))

$\equiv$ wp(`y=x+1`, $y+1 > 2$)

$\equiv x+1+1 > 2$

$\equiv x > 0$

wp(if b S1 else S2, Q) $\equiv$

$(b \wedge wp(S1,Q))$

$\vee (!b \wedge wp(S2,Q))$

S:

```
if (x < 5) {
  x = x*x;
} else {
  x = x+1;
}
```

$\{ x \geq 9 \}$

wp(S, $x \geq 9$) $\equiv$

$(x<5 \wedge$ wp(x=x*x, $x\geq 9$))

$\vee (x\geq 5 \wedge$ wp(x=x+1, $x\geq 9$)) $\equiv$

$(x<5 \wedge x*x>9) \vee (x\geq 5 \wedge x+1\geq 9) \equiv$

$(x<5 \wedge |x|\geq 3) \vee (x\geq 5 \wedge x\geq 8)$