Section 1:
Code Reasoning

cse331-staff@cs.washington.edu

My name is: Vincent Liew
Email: vliew@cs

Today's Goals

- Review of code reasoning
- Practice forward and backward reasoning on straight-line and if-statement code
- Practice identifying the strongest assertion

Before we begin . . .

- “=” vs. “==”
- Read the lecture notes

Reasoning About Code

- Two purposes
  - Prove our code is correct
  - Understand why code is correct

- Forward reasoning: determine what follows from initial conditions
- Backward reasoning: determine sufficient conditions to obtain a certain result

Reasoning about "if"

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

- When S1 executes, we know **`P and b`**
- When S2 executes, we know **`P and not b`**

- Triple is valid iff: there are assertions **`Q1`** and **`Q2`** such that
  - **`{{ P and b }} S1 {{ Q1 }}`** is valid and
  - **`{{ P and not b }} S2 {{ Q2 }}`** is valid and
  - **`Q1 or Q2`** implies **`Q`**
    - we only know that one holds (which depends on **`b`**)

Reasoning about "if"

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

- When S1 executes, we know **`P and b`**
- When S2 executes, we know **`P and not b`**

- Triple is valid iff: there are assertions **`Q1`** and **`Q2`** such that
  - **`{{ P and b }} S1 {{ Q1 }}`** is valid and
  - **`{{ P and not b }} S2 {{ Q2 }}`** is valid and
  - **`Q1 or Q2`** implies **`Q`**
    - we only know that one holds (which depends on **`b`**)
    - **Equivalent to (`Q1`→`Q and Q2` → `Q`)**
      - **May be easier to prove in this form**

Worksheet

- Problems 1 through 4
- 15 Minutes – get as far as you can
- You can collaborate with other students
- Grab a TA if you feel stuck

Forward Reasoning

```
{x >= 0, y >= 0}
y = 16;
{x >= 0, y = 16}
x = x + y
{x >= 16, y = 16}
x = sqrt(x)
{x >= 4, y = 16}
y = y - x
{x >= 4, y <= 12}
```

Forward Reasoning

```
{true}
if (x > 0) {
      {x > 0}
      abs = x
      {x > 0, abs = x}
}
else {
      {x <= 0}
      abs = -x
      {x <= 0, abs = -x}
}
{x > 0, abs = x OR x <= 0, abs = -x}

{abs = |x|}
```

Backward Reasoning

```
{x + 3b - 4 > 0}
a = x + b;
{a + 2b - 4 > 0}
c = 2b - 4
{a + c > 0}
x = a + c
{x > 0}
```

Backward Reasoning

```
{y > 15 || (y <= 5 && y + z > 17)}
if (y > 5) {
      {y > 15}
      x = y + 2
      {x > 17}
}
else {
      {y + z > 17}
      x = y + z;
      {x > 17}
}
{x > 17}
```

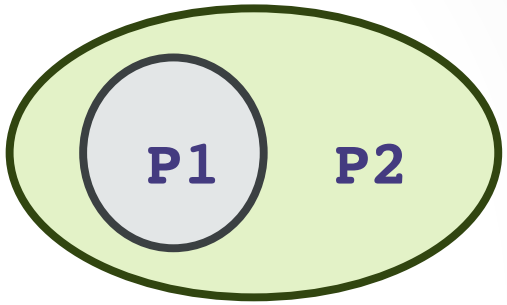
Implication

- Hoare triples are just an extension of logical implication
  - Hoare triple: {P} S {Q}
  - P → Q after statement S
- Everything implies true
- False implies everything

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

Weaker vs. Stronger

- If P1 → P2, then
  - P1 is stronger than P2
  - P2 is weaker than P1



- Weaker statements are more general
- Stronger statements are more restrictive

Worksheet

- Problem 6

Worksheet

- “I attend quiz sections.” “I attend quiz sections on Thursdays.”
- “y > 23” “y >= 23”
- “y = 23” “y >= 23”
- “y < 0.00023” “y < 0.23”
- “y is prime” “y <= 17”

Worksheet

- "I attend quiz sections." "I attend quiz sections on Thursdays."
- "y > 23" "y >= 23"
- "y = 23" "y >= 23"
- "y < 0.00023" "y < 0.23"
- "y is prime" "y <= 17"

Worksheet

- "I attend quiz sections." "I attend quiz sections on Thursdays."
- "y > 23" "y >= 23"
- "y = 23" "y >= 23"
- "y < 0.00023" "y < 0.23"
- "y is prime" "y <= 17"

Worksheet

- "I attend quiz sections." "I attend quiz sections on Thursdays."
- "y > 23" "y >= 23"
- "y = 23" "y >= 23"
- "y < 0.00023" "y < 0.23"
- "y is prime" "y <= 17"

Worksheet

- "I attend quiz sections." "I attend quiz sections on Thursdays."
- "y > 23" "y >= 23"
- "y = 23" "y >= 23"
- "y < 0.00023" "y < 0.23"
- "y is prime" "y <= 17"

Worksheet

- "I attend quiz sections." "I attend quiz sections on Thursdays."
- "y > 23" "y >= 23"
- "y = 23" "y >= 23"
- "y < 0.00023" "y < 0.23"
- "y is prime" "y <= 17" -- ?

Weakest Precondition

- The most lenient assumptions such that a postcondition will be satisfied
- If P* is the weakest precondition for {P} S {Q}, then P → P* for all P that make the Hoare triple valid
- Notation: WP = wp(S, Q)

Weakest Precondition

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

Weakest Precondition

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

|y| > 2

Weakest Precondition

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

```
|y| > 2
```

```
wp(y = x+1;z = y-3, z = 10)
```

Weakest Precondition

```
wp(x = y*y, x > 4)
|y| > 2

wp(y = x+1;z = y-3, z = 10)
wp(y = x+1, wp(z = y-3, z = 10))
wp(y = x+1, y-3 = 10)
wp(y = x+1, y = 13)
x = 12
```

Questions