CSE 331

Software Design & Implementation

Section: Reasoning about Conditionals

Reminders

- Read the welcome email
- Check your access to Ed, Gradescope, and Canvas

Upcoming Deadlines

- HW1 due 11pm tonight (6/23)
- Syllabus Quiz due 11pm tonight (6/23)

Last Time…

- Introduction to straight line reasoning

Today's Agenda

- Introductions & Icebreakers
- Reasoning Worksheet
- New: Reasoning about conditionals

Introduction & Icebreaker

- Turn to a group of 3-5:
  - Name
  - Class standing
  - Interesting thing you're doing this summer
  - First 300-level CSE class you took (or are taking)?
  - Least & most favorite CSE class?
    - Why?

- Introduce someone you met to the class! (first three bullet points)

Why reason about code?

- Prove that code is correct
- Understand *why* code is correct
- Diagnose why/how code is *not* correct
- Specify code behavior

Logical reasoning about code

- Determine facts that hold of program state between statements
  - "Fact" ~ assertion (logical formula over program state, informally "value(s) of some/all program variables)
  - Driven by assumption (precondition) or goal (postconditon)

- Forward reasoning
  - What facts follow from initial assumptions?
  - Go from precondition to postcondition

- Backward reasoning
  - What facts need to be true to reach a goal?
  - Go from postcondition to precondition

Hoare Logic: Validity by Reasoning

- Checking validity of **{{*P*}} S {{*Q*}}**
  - Valid iff, starting from any state satisfying ***P***, executing ***S*** results in a state satisfying ***Q***

- Forward reasoning:
  - Reason from ***P*** to strongest postcondition **{{*P*}} *S* {{*R*}}**
  - Check that ***R*** implies ***Q*** (i.e., ***Q*** is weaker)

- Backward reasoning:
  - Reason from ***Q*** to get weakest precondition **{{*R*}} *S* {{*Q*}}**
  - Check that ***P*** implies ***R*** (i.e., ***P*** is stronger)

Implication (=>)

- Logic formulas with *and* (&, &&, or ∧), *or* (|, ||, or ∨) and *not* (! or ¬) have the same meaning they do in programs

- Implication might be a bit new, but the basic idea is pretty simple. Implication p=>q is true as long as q is always true whenever p is

p	q	p => q
T	T	T
T	F	F
F	T	T
F	F	T

Assignment Statements

- Reasoning about `x = y;`

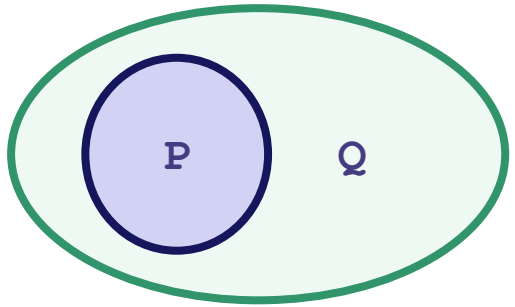
- Forward reasoning:
  - add “x = y” as a new fact
  - (also rewrite any existing references to “x” to use new value)

- Backward reasoning:
  - replace all instances of “x” in the postcondition with “y”

Weaker vs. stronger

Formal definition:

- If $P \Rightarrow Q$, then
  - Q is weaker than P
  - P is stronger than Q



Intuitive definition:

- "Weak" means unrestrictive; a weaker assertion has a larger set of possible program states (*e.g.*, `x != 0`)
- "Strong" means restrictive; a stronger assertion has a smaller set of possible program states (*e.g.*, `x = 1` or `x > 0` are both stronger than `x != 0`).

Worksheet (Reasoning)

- Problems: 1, 3, 7
  - Take ~10 minutes to get where you can
- Find a partner and work with them
- Let me know if you feel stuck
- We'll walk through some solutions afterwards

Worksheet – problem 1

```
{{ 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 = 16 - x }}
⇒ {{ x >= 4 ∧ y <= 12 }}
```

Worksheet – problem 3

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

Worksheet – problem 7

```
{{ y > 23 }}
```

```
{{ y >= 23 }}
```

```
{{ y = 23 }}
```

```
{{ y >= 23 }}
```

```
{{ y < 0.23 }}
```

```
{{ y < 0.00023 }}
```

```
{{ x = y * z }}
```

```
{{ y = x / z }}
```

```
{{ is_prime(y) }}
```

```
{{ is_odd(y) }}
```

Worksheet – problem 7

`{{ y > 23 }}`	is stronger than	`{{ y >= 23 }}`
`{{ y = 23 }}`		`{{ y >= 23 }}`
`{{ y < 0.23 }}`		`{{ y < 0.00023 }}`
`{{ x = y * z }}`		`{{ y = x / z }}`
`{{ is_prime(y) }}`		`{{ is_odd(y) }}`

Worksheet – problem 7

`{{ y > 23 }}` is stronger than `{{ y >= 23 }}`

`{{ y = 23 }}` is stronger than `{{ y >= 23 }}`

`{{ y < 0.23 }}` `{{ y < 0.00023 }}`

`{{ x = y * z }}` `{{ y = x / z }}`

`{{ is_prime(y) }}` `{{ is_odd(y) }}`

Worksheet – problem 7

`{{ y > 23 }}`	is stronger than	`{{ y >= 23 }}`
`{{ y = 23 }}`	is stronger than	`{{ y >= 23 }}`
`{{ y < 0.23 }}`	is weaker than	`{{ y < 0.00023 }}`
`{{ x = y * z }}`		`{{ y = x / z }}`
`{{ is_prime(y) }}`		`{{ is_odd(y) }}`

Worksheet – problem 7

`{{ y > 23 }}` is stronger than `{{ y >= 23 }}`

`{{ y = 23 }}` is stronger than `{{ y >= 23 }}`

`{{ y < 0.23 }}` is weaker than `{{ y < 0.00023 }}`

`{{ x = y * z }}` is **incomparable** with `{{ y = x / z }}`

`{{ is_prime(y) }}` `{{ is_odd(y) }}`

Worksheet – problem 7

`{{ y > 23 }}` is stronger than `{{ y >= 23 }}`

`{{ y = 23 }}` is stronger than `{{ y >= 23 }}`

`{{ y < 0.23 }}` is weaker than `{{ y < 0.00023 }}`

`{{ x = y * z }}` is **incomparable** with `{{ y = x / z }}`

`{{ is_prime(y) }}` is **incomparable** with `{{ is_odd(y) }}`

Questions?

- What is the most surprising thing about this?
- What is the most confusing thing?
- What will need a bit more thinking to digest?

If Statements

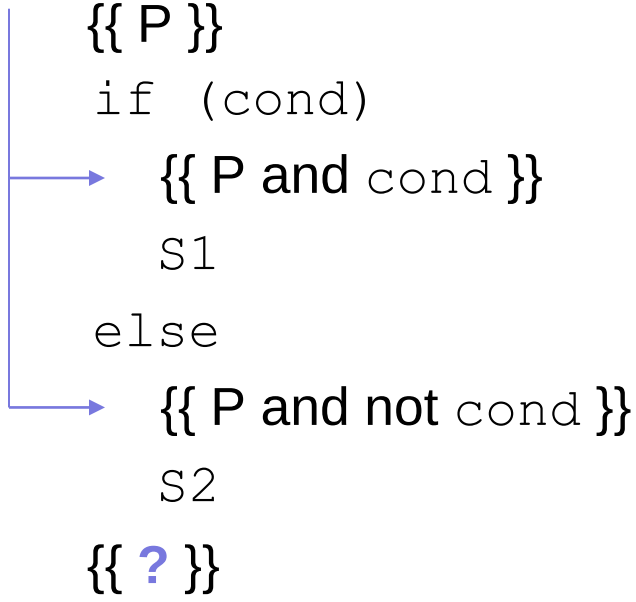
If Statements

Forward reasoning

{{ P }}
```
if (cond)
  S1
else
  S2
```
{{ ? }}

If Statements

Forward reasoning



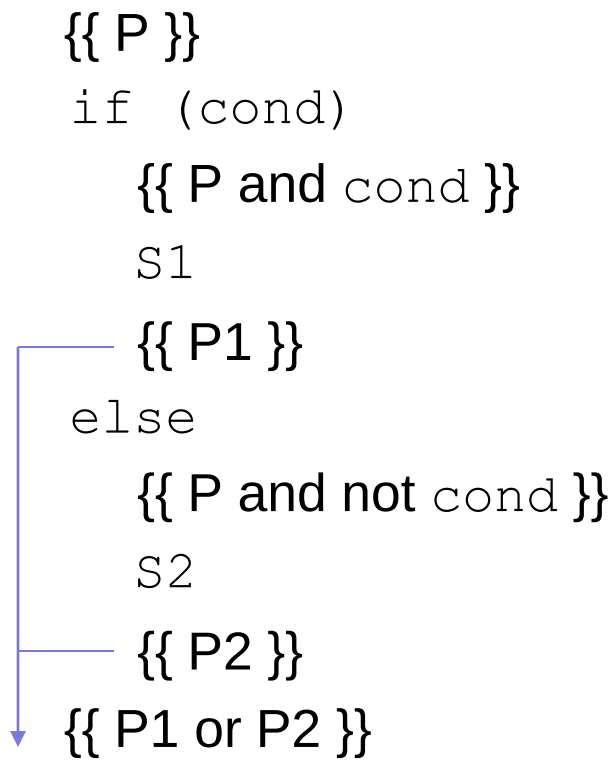
If Statements

Forward reasoning

{{ P }}
`if (cond)`
  {{ P and `cond` }}
  `S1`
  {{ P1 }}
`else`
  {{ P and not `cond` }}
  `S2`
  {{ P2 }}
{{ ? }}

If Statements

Forward reasoning



If Statements

Backward reasoning

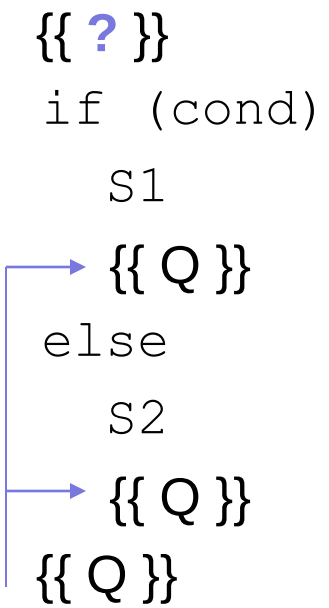
{{ ? }}

```
if (cond)
  S1
else
  S2
```

{{ Q }}

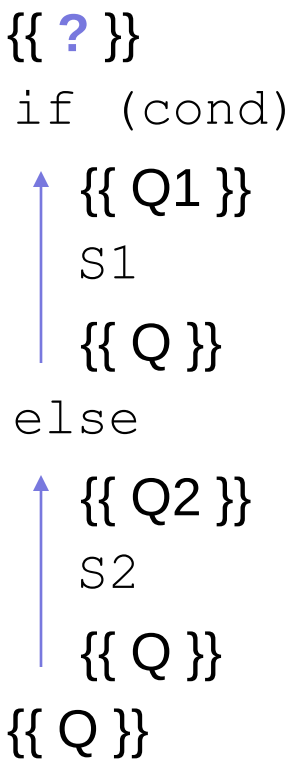
If Statements

Backward reasoning



If Statements

Backward reasoning



If Statements

Backward reasoning

{{ `cond` and Q1 or
not `cond` and Q2 }}

```
if (cond)
    {{ Q1 }}
    S1
    {{ Q }}
else
    {{ Q2 }}
    S2
    {{ Q }}
{{ Q }}
```

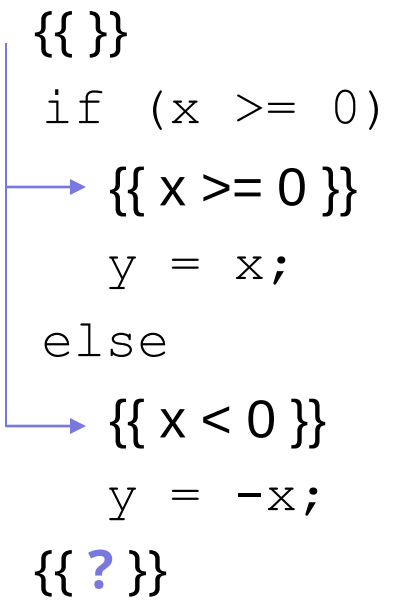
If-Statement Example

Forward reasoning

```
{{ }}
if (x >= 0)
  y = x;
else
  y = -x;
{{ ? }}
```

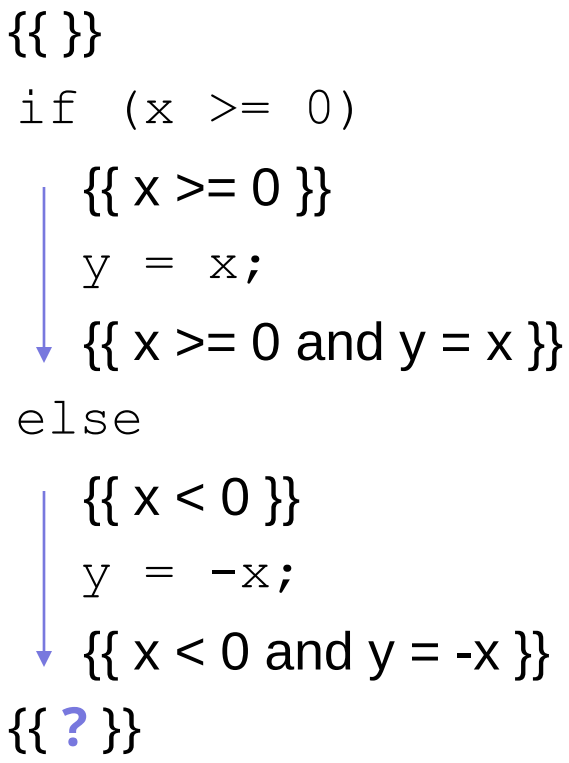
If-Statement Example

Forward reasoning



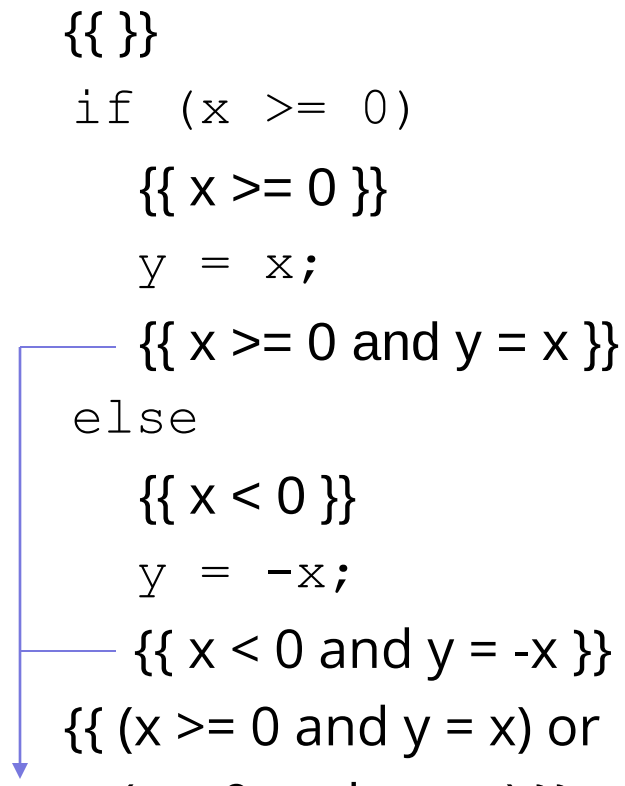
If-Statement Example

Forward reasoning



If-Statement Example

Forward reasoning



If-Statement Example

Forward reasoning

{{ }}
```
if (x >= 0)
```
  {{ x >= 0 }}
  ```
  y = x;
  ```
  {{ x >= 0 and y = x }}
```
else
```
  {{ x < 0 }}
  ```
  y = -x;
  ```
  {{ x < 0 and y = -x }}

{{ y = |x| }}

If-Statement Example

Forward reasoning

{{ }}
```
if (x >= 0)
```
   {{ x >= 0 }}
   ```
   y = x;
   ```
   {{ x >= 0 and y = x }}
```
else
```
   {{ x < 0 }}
   ```
   y = -x;
   ```
   {{ x < 0 and y = -x }}
{{ y = |x| }}

Warning: many write {{ y >= 0 }} here

That is true but it is *strictly* weaker.
(It includes cases where y != x)

If-Statement Example

Forward reasoning

{{ }}
```
if (x >= 0)
```
{{ x >= 0 }}
```
  y = x;
```
{{ x >= 0 and y = x }}
```
else
```
{{ x < 0 }}
```
  y = -x;
```
{{ x < 0 and y = -x }}
{{ y = |x| }}

Backward reasoning

{{ ? }}
```
if (x >= 0)
  y = x;
else
  y = -x;
```
{{ y = |x| }}

If-Statement Example

Forward reasoning

{{ }}
```
if (x >= 0)
```
{{ x >= 0 }}
```
   y = x;
```
{{ x >= 0 and y = x }}
```
else
```
{{ x < 0 }}
```
   y = -x;
```
{{ x < 0 and y = -x }}
{{ y = |x| }}

Backward reasoning

{{ ? }}
```
if (x >= 0)
   y = x;
```
{{ y = |x| }}
```
else
   y = -x;
```
{{ y = |x| }}
{{ y = |x| }}

If-Statement Example

Forward reasoning

{{ }}
```
if (x >= 0)
```
{{ x >= 0 }}
```
  y = x;
```
{{ x >= 0 and y = x }}
```
else
```
{{ x < 0 }}
```
  y = -x;
```
{{ x < 0 and y = -x }}
{{ y = |x| }}

Backward reasoning

{{ ? }}
```
if (x >= 0)
```
{{ x = |x| }}
```
  y = x;
```
{{ y = |x| }}
```
else
```
{{ -x = |x| }}
```
  y = -x;
```
{{ y = |x| }}
{{ y = |x| }}

If-Statement Example

Forward reasoning

```
{{ }}
if (x >= 0)
   {{ x >= 0 }}
   y = x;
   {{ x >= 0 and y = x }}
else
   {{ x < 0 }}
   y = -x;
   {{ x < 0 and y = -x }}
{{ y = |x| }}
```

Backward reasoning

```
{{ ? }}
if (x >= 0)
   {{ x >= 0 }}
   y = x;
   {{ y = |x| }}
else
   {{ x <= 0 }}
   y = -x;
   {{ y = |x| }}
{{ y = |x| }}
```

If-Statement Example

Forward reasoning

{{ }}
```
if (x >= 0)
```
{{ x >= 0 }}
```
  y = x;
```
{{ x >= 0 and y = x }}
```
else
```
{{ x < 0 }}
```
  y = -x;
```
{{ x < 0 and y = -x }}
{{ y = |x| }}

Backward reasoning

{{ (x >= 0 and x >= 0) or
(x < 0 and x <= 0) }}
```
if (x >= 0)
```
{{ x >= 0 }}
```
  y = x;
```
{{ y = |x| }}
```
else
```
{{ x <= 0 }}
```
  y = -x;
```
{{ y = |x| }}
{{ y = |x| }}

If-Statement Example

Forward reasoning

{{ }}
```
if (x >= 0)
```
  {{ x >= 0 }}
  ```
  y = x;
  ```
  {{ x >= 0 and y = x }}
```
else
```
  {{ x < 0 }}
  ```
  y = -x;
  ```
  {{ x < 0 and y = -x }}
{{ y = |x| }}

Backward reasoning

{{ x >= 0 or x < 0 }}
```
if (x >= 0)
```
  {{ x >= 0 }}
  ```
  y = x;
  ```
  {{ y = |x| }}
```
else
```
  {{ x <= 0 }}
  ```
  y = -x;
  ```
  {{ y = |x| }}
{{ y = |x| }}

If-Statement Example

Forward reasoning

{{ }}
```
if (x >= 0)
```
  {{ x >= 0 }}
  ```
  y = x;
  ```
  {{ x >= 0 and y = x }}
```
else
```
  {{ x < 0 }}
  ```
  y = -x;
  ```
  {{ x < 0 and y = -x }}
{{ y = |x| }}

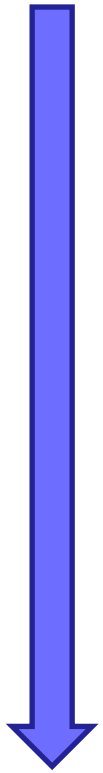
Backward reasoning

{{ }}
```
if (x >= 0)
```
  {{ x >= 0 }}
  ```
  y = x;
  ```
  {{ y = |x| }}
```
else
```
  {{ x <= 0 }}
  ```
  y = -x;
  ```
  {{ y = |x| }}
{{ y = |x| }}

Conditionals, summary

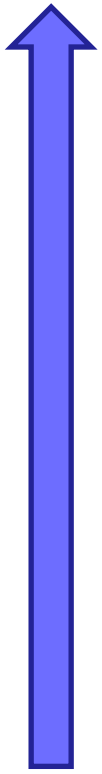
Forward reasoning

```
{{P}}
if (b)
    {{P ∧ b}}
    S1
    {{Q1}}
else
    {{P ∧ !b}}
    S2
    {{Q2}}
{{Q1 ∨ Q2}}
```



Backward reasoning

```
{{ (b ∧ P1) ∨ (!b ∧ P2) }}
if (b)
  {{P1}}
  S1
  {{Q}}
else
  {{P2}}
  S2
  {{Q}}
{{Q}}
```



Worksheet (Conditionals)

- Problems: 2, 4, 6
  - Take ~10 minutes to get where you can
- Find a partner and work with them
- Let me know if you feel stuck
- We'll walk through some solutions afterwards

Worksheet – problem 2

```
{{ true }}
if (x > 0) {
  {{ x > 0 }}
  y = 2 * x;
  {{ x > 0 ∧ y = 2x }}
} else {
  {{ x <= 0 }}
  y = -2 * x;
  {{ x <= 0 ∧ y = -2x }}
}
{{ (x > 0 ∧ y = 2x) ∨ (x <= 0 ∧ y = -2x) }}
⇒ {{ y = 2|x| }}
```

Worksheet – problem 4

```
{{ y > 15 V (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 }}
```

Worksheet – problem 6 (forward)

```
{{ true }}
if (x < y) {
  {{ true ∧ x < y }}
  m = x;
  {{ x < y ∧ m = x }}
} else {
  {{ true ∧ x >= y }}
  m = y;
  {{ x >= y ∧ m = y }}
}
{{ (x < y ∧ m = x) ∨ (x >= y ∧ m = y) }}
⇒ {{ m = min(x, y) }}
```

Worksheet – problem 6 (backward)

```
{{ true }} ⇔
{{ (x <= y ∧ x < y) ∨ (y <= x ∧ x >= y) }}
if (x < y) {
  {{ x = min(x, y) }} ⇔ {{ x <= y }}
  m = x;
  {{ m = min(x, y) }}
} else {
  {{ y = min(x, y) }} ⇔ {{ x >= y }}
  m = y;
  {{ m = min(x, y) }}
}
{{ m = min(x, y) }}
```

Before next lecture…

1. Familiarize yourself with website:

   http://courses.cs.washington.edu/courses/cse331/22su/

   - read the syllabus
   - read the academic integrity policy
   - find the homework list
   - find the link to Canvas

2. Do HW1 tonight! (reminder: deadline is 11pm)
   - submit a PDF on Gradescope
   - limit this to at most 90 min
   - do not use formal reasoning