CSE 331
Software Design & Implementation

Spring 2022

Section 1 – Code Reasoning

Administrivia

- HW1 due next Wednesday.
  - Last two questions have loops: covered tomorrow in lecture.

- Any questions before we dive in?
  - What are the most interesting/confusing/puzzling things so far in the course?

Agenda

- Introductions?
- Review logical reasoning about code with Floyd Logic
- Practice both forward and backward modes
  - Just assignment, conditional (“if-then-else”), and sequence
  - Logical rules from yesterday’s lecture/notes
- Review logical strength of assertions (weaker *vs.* stronger)
- Practice determining stronger/weaker assertions

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 <u>pre</u>condition to <u>post</u>condition

- Backward reasoning
  - What facts need to be true to reach a goal?
  - Go from <u>post</u>condition to <u>pre</u>condition

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”

Conditionals, more closely

Forward reasoning

$\{\{P\}\}$

`if (b)`

  $\{\{P \wedge \mathtt{b}\}\}$

  S_1

  $\{\{Q_1\}\}$

`else`

  $\{\{P \wedge \mathtt{!b}\}\}$

  S_2

  $\{\{Q_2\}\}$

$\{\{Q_1 \vee Q_2\}\}$

Backward reasoning

$\{\{\ (\mathtt{b} \wedge P_1) \vee (\mathtt{!b} \wedge P_2)\ \}\}$

`if (b)`

  $\{\{P_1\}\}$

  S_1

  $\{\{Q\}\}$

`else`

  $\{\{P_2\}\}$

  S_2

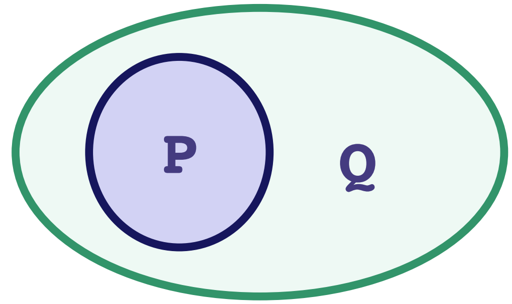
  $\{\{Q\}\}$

$\{\{Q\}\}$

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

- 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 ∨ (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) }}
```

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?