CSE 331
Software Design & Implementation

Autumn 2021

Section 1 – Code Reasoning

Administrivia

- HW0 due tomorrow (Friday 10/1 by 5PM).
- Any questions before we dive in?
  - What are the most interesting/confusing/puzzling things so far in the course?

Agenda

- Introductions?
- Website tour
- Review and practice logical reasoning about code with Hoare Logic
- Review and practice logical strength of assertions (weaker *vs.* stronger)
- Onwards to forward reasoning

Introductions

Website Tour

Why reason about code?

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

From lecture:

Hoare Logic: First definitions

- **Program State:** Values of all related variables
- **Assertion:** True/False claim (proposition) about the program state at a certain point in execution
- An assertion **holds** for a program state if it is true at that point.
- **Precondition:** Assertion before the code
    - Assumptions about when the code is used
- **Postcondition:** Assertion after the code
    - What we want the result of the code to be

From lecture:

Hoare Logic

- **Hoare Triple:** Two assertions surrounding a piece of code
  - {{ P }} S {{ Q }}
    - P is the precondition, S is the code, Q is the postcondition
    - P,Q are **specifications**
- A Hoare triple {{ P }} S {{ Q }} is **valid** if in any state that P holds, Q holds after running the code S.
  - If P is true, after running S we have that Q is true.
  - Otherwise the triple is **invalid.**

Let's practice!
(Q1, Q2)

A Note on Implication (=>)

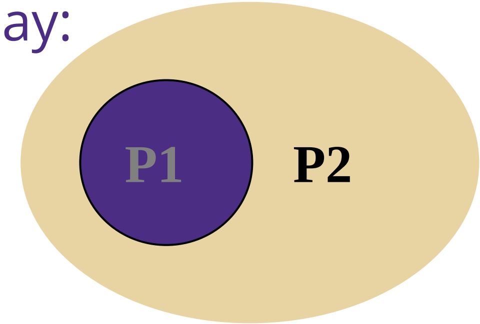
- Implication might be a bit new, but the basic idea is pretty simple. Implication p=>q is true as long as q is always true whenever p is

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

From lecture:

Weaker / Stronger Assertions

- If P1 implies P2 (written P1 ⇒ P2) then we say:
  - P1 is **stronger** than P2
  - P2 is **weaker** than P1
- In other words:
  - P1 is “more difficult” to satisfy than P2
  - P1 puts more constraints on program states
  - P1 gives us more information about the program state



Let's practice!

(Q3)

Forward Reasoning

Forward Reasoning

- "What facts follow from initial assumptions about the code?"
- Precondition is **given**
- Fill in the **strongest** postcondition
  - For an assignment statement x = y
    - Add fact "x = y" to what is known
    - important subtleties here (more later...)
  - Later: if statements and loops

From lecture:

Example of Forward Reasoning

$\{\{\ w > 0\ \}\}$

x = 17;

$\{\{\ w > 0 \land x = 17\ \}\}$

y = 42;

$\{\{\ w > 0 \land x = 17 \land y = 42\ \}\}$

z = w + x + y;

$\{\{\ w > 0 \land x = 17 \land y = 42 \land z = w + 59\ \}\}$

Let’s Try One Together (Q4)

Suppose we know that `i >= 2` at the start…

```
{{ i >= 2 }}
x = 2 * i;


y = x;


z = (x + y) / 2;
```

Let's Try One Together (Q4)

Suppose we know that `i >= 2` at the start, what do we know about z at the end?

```
{{ i >= 2 }}
x = 2 * i;
{{ x = 2 * i ∧ i >= 2 }}
y = x;

z = (x + y) / 2;
```

Let's Try One Together (Q4)

Suppose we know that `i >= 2` at the start, what do we know about z at the end?

```
{{ i >= 2 }}
x = 2 * i;
{{ x = 2 * i ∧ i >= 2 }}
y = x;
{{ y = x ∧ x = 2 * i ∧ i >= 2 }}
z = (x + y) / 2;
```

Let's Try One Together (Q4)

Suppose we know that `i >= 2` at the start, what do we know about z at the end?

```
{{ i >= 2 }}
x = 2 * i;
{{ x = 2 * i ∧ i >= 2 }}
y = x;
{{ y = x ∧ x = 2 * i ∧ i >= 2 }}
z = (x + y) / 2;
{{ z = (x + y) / 2 ∧ y = x ∧ x = 2 * i ∧ i >= 2
}}
```

Let's Try One Together (Q4)

Suppose we know that **`i >= 2`** at the start, what do we know about z at the end?

```
{{ i >= 2 }}
x = 2 * i;
{{ x = 2 * i ∧ i >= 2 }}
y = x;
{{ y = x ∧ x = 2 * i ∧ i >= 2 }}
z = (x + y) / 2;
{{ z = (x + y) / 2 ∧ y = x ∧ x = 2 * i ∧ i >= 2
}}
⇒ {{ z = (2 * i + 2 * i) / 2 ∧ i >= 2 }}
```

Let's Try One Together (Q4)

Suppose we know that `i >= 2` at the start, what do we know about z at the end?

```
{{ i >= 2 }}
x = 2 * i;
{{ x = 2 * i ∧ i >= 2 }}
y = x;
{{ y = x ∧ x = 2 * i ∧ i >= 2 }}
z = (x + y) / 2;
{{ z = (x + y) / 2 ∧ y = x ∧ x = 2 * i ∧ i >= 2
}}
⇒ {{ z = (2 * i + 2 * i) / 2 ∧ i >= 2 }}
⇒ {{ z = 2 * i ∧ i >= 2 }}
```

Let's Try One Together (Q4)

Suppose we know that **`i >= 2`** at the start, what do we know about z at the end?

```
{{ i >= 2 }}
x = 2 * i;
{{ x = 2 * i ∧ i >= 2 }}
y = x;
{{ y = x ∧ x = 2 * i ∧ i >= 2 }}
z = (x + y) / 2;
{{ z = (x + y) / 2 ∧ y = x ∧ x = 2 * i ∧ i >= 2
}}
⇒ {{ z = (2 * i + 2 * i) / 2 ∧ i >= 2 }}
⇒ {{ z = 2 * i ∧ i >= 2 }}
⇒ {{ z >= 4 }}
```

Let’s practice!
(Q5,Q6)

Questions?

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