Section 1
Code Reasoning + Version Control

CSE 331 - Summer 2018

Slides borrowed and adapted from CSE331 18sp Sec01 Slides
## Motivation

- Two purposes
  - Know that our code is correct
  - Understand *why* our code is correct
- Forward reasoning: determine what follows from initial conditions
- Backward reasoning: determine sufficient conditions to obtain a result

## Terminology

**Program State**

The *program state* is the values of all (relevant) variables.
Terminology

Program State

The program state is the values of all (relevant) variables.

Assertion

- An assertion is a logical formula referring to the program state at a given point.
- An assertion holds for a program state if the formula is true when those values are substituted for the variables.
- An assertion before the code is a precondition - these represent assumptions about when that code is used.
- An assertion after the code is a postcondition - these represent what we want the code to accomplish.

Forward Reasoning

- Given: precondition
- Finds: postcondition
- Aka find the program state after executing code, when using given assumptions of program state before execution.

Forward Reasoning

```c
// { x ≥ 0 ∧ y ≥ 0 }
y = 16;

x = x + y;

x = sqrt(x);

y = y - x;
```
Forward Reasoning

// { x ≥ 0 ∧ y ≥ 0 }
y = 16;
// { x ≥ 0 ∧ y = 16 }
x = x + y;
// { x ≥ 16 ∧ y = 16 }
x = sqrt(x);
// { x ≥ 16 ∧ y = 16 }
y = y - x;
// { x ≥ 4 ∧ y = 16 }

Forward Reasoning

// { true }
if (x > 0) {
abs = x;
} else {
abs = -x;
}
Forward Reasoning

// { true }
if (x > 0) {
  // { x > 0 }
  abs = x;
} else {
  // { x ≤ 0 }
  abs = -x;
}

// { (x > 0 ∧ abs = x) ∨ (x ≤ 0 ∧ abs = −x) }

Backward Reasoning

// { true }
if (x > 0) {
  // { x > 0 }
  abs = x;
} else {
  // { x ≤ 0 }
  abs = -x;
}

// { abs = |x| }
### Backward Reasoning

- Given: postcondition
- Finds: **weakest** precondition
- What is weakest precondition?

### Weaker/Stronger

- Weaker statements = more general
- Stronger statements = more specific / restrictive / informational
- If $A \rightarrow B$, $A$ is stronger and $B$ is weaker
- If $B \rightarrow A$, $B$ is stronger and $A$ is weaker
- If neither, then $A$ and $B$ not comparable.
Weaker/Stronger

- Weaker statements = more general
- Stronger statements = more specific / restrictive / informational
- If $A \rightarrow B$, $A$ is stronger and $B$ is weaker
- If $B \rightarrow A$, $B$ is stronger and $A$ is weaker
- If neither, then $A$ and $B$ not comparable.

Example
- $x = 16$ is stronger than $x > 0$
- “Frank is an awesome TA” is stronger than “Frank is a TA”

Backward Reasoning

- Given: postcondition
- Finds: weakest precondition
- Aka finds most general assumption code will use to get given postcondition.

\[
a = x + b; \\
c = 2b - 4; \\
x = a + c; \\
// \{ x > 0 \}
\]
Backward Reasoning

// Backward reasoning is used to determine the weakest precondition

// { x + 3b − 4 > 0 }
a = x + b;
// { a + 2b − 4 > 0 }
c = 2b − 4;
// { a + c > 0 }
x = a + c;
// { x > 0 }

// { x > 0 }

a = x + b;

// { a + c > 0 }
x = a + c;
// { x > 0 }

c = 2b − 4;
// { a + c > 0 }

x = a + c;
// { x > 0 }

// Backward reasoning is used to determine the weakest precondition

// { x + 3b − 4 > 0 }
a = x + b;
// { a + 2b − 4 > 0 }
c = 2b − 4;
// { a + c > 0 }
x = a + c;
// { x > 0 }
Hoare Triples

- Hoare triples are just an extension of logical implication
  - \{P\} S \{Q\}
  - P = precondition
  - S = code
  - Q = postcondition
- A Hoare triple can be valid or invalid
  - **Valid** if for all states for which P holds, executing S always produces a state for which Q holds
  - **Invalid** otherwise

\{ P \} S \{ true \}

\{ x \neq 0 \} y = x*x; \{ y > 0 \}

\{ false \} S \{ Q \}

\{ P \} S \{ true \}

\{ x \neq 0 \} y = x*x; \{ y > 0 \}

\{ false \} S \{ Q \}

- When P is false, there is no condition when P holds
- For all states where P holds (i.e. none) executing S will produce a state in which Q holds
- \{ P \} S \{ true \}
Hoare Triples

- \{ x \neq 0 \} y = x \times x; \{ y > 0 \} valid
- \{ \text{false} \} S \{ Q \} valid
  - When P is false, there is no condition when P holds
  - For all states where P holds (i.e. none) executing S will produce a state in which Q holds
- \{ P \} S \{ true \} valid
  - Any state for which P holds that is followed by the execution of S will produce some state
  - For any state, true always holds (i.e. true is true)

Outline

1 Intro
2 Code Reasoning
   - Forward Reasoning
     - Weaker/Stronger Statements
     - Backward Reasoning
     - Hoare Triples
3 Version Control

What is Version Control?

- Aka source control / revision control
- Tracking changes to code
  - See a history of changes
  - Revert back to an older version
  - Merge changes from multiple sources
- We will use git/Gitlab, but others exist
  - Gitlab is very similar to GitHub but can be tied to CSE accounts and authentication
  - Subversion, Mercurial, CVS
  - Email, Dropbox, USB sticks (don’t even think of doing this)
- git can be used in many ways, and we are using it in a centralized way
  - The repo on the CSE Gitlab Server is the master repo.

git for This Course

1 TAs create a repository for each student on the CSE Gitlab server.
2 You clone the repo from the server to get a local copy on your computer.
3 TAs push starter code for each assignment to your repo on the server.
4 You pull the starter code from the server to your local copy of your repo.
5 You modify (write code) files in your local repo.
6 You add each file you modified and commit those changes to your local repo.
7 You push the changes to your local repo to the server repo.
8 You create a tag pointing to your final version and push the tag.
9 TAs pull the version of your code referred by your tag and grade it.