Deductive Reasoning and Hoare Logic

Announcements

- No class 4/21 and 4/30
- Lab7 - due last night
- HW3 - due next Monday (4/7)
- Project part1
  - Deadline tonight
  - Each group submits once to gradescope
  - **No extensions - submit whatever you have**

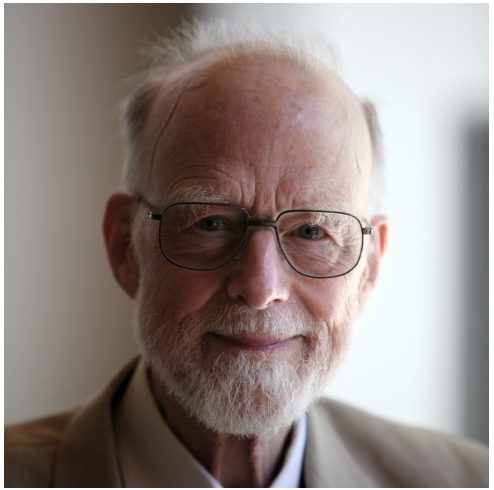
Next Class:

- Project check-ins
- Attendance required
- Group 1: 10:10-10:25
  - Khanh Ha, Rachel, Ruth, Ferida, Emily
- Group 2: 10:25-10:40
  - Glory, Rebecca, Ally, Megan, Keziah
- Group 3: 10:40-10:55
  - Clara, Emma, Alison, Amina, Ranty, Caren
- Group 4: 10:55-11:10
  - Tianyun, Cecilia C, Cecilia Z, Yang
- Group 5: 11:10-11:30
  - Hazel, Sarah, Bridge, Jenny, Reagan

Deductive Reasoning

Deductive Reasoning - the backbone of Formal Verification

"Computer programming is an exact science in that all the properties of a program and all the consequences of executing it in any given environment can, in principle, be found out from the text of the program itself by means of purely deductive reasoning." - Tony Hoare 1960



“Strength” of a condition

A predicate P is *stronger* than Q if it is the case that P implies Q.

(Similarly Q is weaker than P.)

A politicians example:

- I will keep unemployment below 3% is *stronger than*
- I will keep unemployment below 15%

Strength of Triples

$(x = 6) \Rightarrow (x > 0)$, so $(x = 6)$ is stronger than $(x > 0)$

The triple:

```
{x = 5} x := x + 1 {x = 6}
```

says more about the code than:

```
{x = 5} x := x + 1 {x > 0}
```

Strength of Preconditions

The condition (x > 0) says less about a state than the condition (x = 5).

It is the weaker condition

but the statement

```
{x > 0} x := x + 1 {x > 1}
```

says more about the code than:

```
{x = 5} x := x + 1 {x > 1}
```

Strength of Preconditions

If a precondition P1 is weaker than P2,

then {P1}S {Q} is stronger than {P2}S {Q}

A weaker precondition leads to a stronger triple

Strengthening a Precondition

It is always safe to make a precondition *stronger*

The rule:

$$\frac{P_s \Rightarrow P_w \qquad \{P_w\}\ S\ \{Q\}}{\{P_s\}\ S\ \{Q\}}$$

An instance:

$$\frac{(y = 2) \Rightarrow (y > 0) \qquad \{y > 0\}\ \texttt{x := y}\ \{x > 0\}}{\{y = 2\}\ \texttt{x := y}\ \{x > 0\}}$$

Weakening a postcondition

It is always safe to make a postcondition *weaker*

The rule:

$$\frac{\{P\}\ S\ \{Q_s\} \qquad Q_s \Rightarrow Q_w}{\{P\}\ S\ \{Q_w\}}$$

An instance:

$$\frac{\{x > 2\}\ \texttt{x} := \texttt{x} + 1\ \{x > 3\} \qquad (x > 3) \Rightarrow (x > 1)}{\{x > 2\}\ \texttt{x} := \texttt{x} + 1\ \{x > 1\}}$$

Deductive Reasoning

1. **Forward Reasoning:** Given a precondition, does the postcondition hold?

2. **Backward Reasoning:** Given a postcondition, what is the proper precondition?

Backward reasoning is usually preferable to forward reasoning. Given a specific goal, backward reasoning shows what must be true before execution to achieve desired results. Given an error, it gives input that exposes the error.

Forward Reasoning

```
// precondition: x is even
x = x + 3;  // x is odd
y = 2x;     // x is odd && y is even
x = 5;      // x = 5 && y is even
// postcondition: x = 5 && y is even
```

Backward Reasoning

Given a postcondition, what is the weakest precondition that this implies?

```
// precondition: x > ( -1 / 2 )
x = x + 3;
y = 2x;
x = 5;
// postcondition: y > x
```

We can use this to *verify* or *derive* a precondition

Example: Forward Reasoning

{ x > 0 }

z = 5;

{ z = 5 }

x = z + 5;

{ _____ }

y = -x;

{ _____ }

z = y / 2;

{ _____ }

x = 0;

{ _____ }

Find the strongest postcondition using forward reasoning

Example: Backward Reasoning

{ _____ }

z = z + 3;

{ _____ }

x = 3 * z;

{ x ≤ 9 }

Find the weakest precondition with backward reasoning

$$\vdash \{Q[E/x]\}\ x = E\ \{Q\} \qquad \text{(Assignment)}$$

Proof Rules for IMP language

$$\vdash \{Q[E/x]\}\ x = E\ \{Q\} \qquad \text{(Assignment)}$$

$$\frac{\vdash \{P\}C_1\{Q\} \quad \vdash \{Q\}C_2\{R\}}{\vdash \{P\}C_1; C_2\{R\}} \qquad \text{(Composition)}$$

$$\frac{\begin{array}{cccc} \vdash & \{P \wedge C\} & S_1 & \{Q\} \\ \vdash & \{P \wedge \neg C\} & S_2 & \{Q\} \end{array}}{\vdash \{P\}\ \texttt{if}\ C\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{Q\}} \qquad \text{(If)}$$

$$\frac{\vdash \{P \wedge C\}S\{P\}}{\vdash \{P\}\texttt{while}\ C\ \texttt{do}\ S\{P \wedge \neg C\}} \qquad \text{(While)}$$

Example: Backward Reasoning

{true}

if (x > 0)

{ _____ }

y = x

{ _____ }

else

{ _____ }

y = -x

{ _____ }

{y >= 0 }

Prove the correctness of this Hoare triple

Example: Backward Reasoning

{x ≥ 0}
i := 0; y := 0;
{ _____ }
while i < x do
    { _____ }
    y := y + (2 * i + 1);
    { _____ }
    i := i + 1;
    { _____ }
{ _____ }
{y = x *x }

Step 1: come up a suitable loop invariant

What is a loop invariant?
- True before the loop starts
- True after each iteration

Loop invariant:
y = i * i && 0 <= i < x

Lab Today: Dafny

Lab Today

Part 1: handwritten deductive reasoning

Part 2: automated deductive reasoning via dafny

Dafny

- Install dafny-lang vs code extension

- Dafny is a *verification-aware programming language*
  - Developed at MSFT

- As you type in your program, Dafny's verifier constantly looks over your shoulder, flags any errors, shows you counterexamples, and congratulates you when your code matches your specifications. When you're done, Dafny can **compile your code to C#, Go, Python, Java, or JavaScript**

Summary

- Formal verification
  - More guarantees than testing
  - Does not execute the program

- FV either requires
  - Annotations
  - program as part of the proof