Announcements

- Homework 0 due on Tuesday
- Questions?

Office hours schedule finalized at:
- [www.cs.rpi.edu/~milanova/csci2600](http://www.cs.rpi.edu/~milanova/csci2600)
- Unfortunately, as of 1PM today, office hour room assignment is not finalized

---

**Reasoning About Code; Hoare Logic**

**Forward vs. Backward Reasoning**

- Forward reasoning is more intuitive, just simulates the code
  - Introduces facts that may be irrelevant to the goal
  - Takes longer to prove task or realize task is hopeless
- Backward reasoning is usually more helpful
  - Given a specific goal, shows what must hold beforehand in order to achieve this goal
  - Given an error, gives input that exposes error

---

**Forward Reasoning: Putting Statements Together**

Precondition: \( x \geq 0 \); 

\[
\begin{align*}
\text{if} \ (x \neq 0) \{ & \quad x > 0 \land z = x \\
\text{else} \{ & \quad x = 0 \land z = 1 \\
\end{align*}
\]

Postcondition: \( z > 0 \); 

Therefore, postcondition holds!

---

**Forward Reasoning With a Loop**

Precondition: \( x >= 0 \);

\[
\begin{align*}
i &= x; & \{ x >= 0 \land i = x \} \\
z &= 0; & \{ \} \\
\text{while} \ (i \neq 0) \{ & \quad x >= 0 \land i = x \land z = 0 \\
z &= z+1; & \quad ??? \\
i &= i-1; & \quad ??? \\
\} \\
\end{align*}
\]

Postcondition: \( x = z \);

Does the postcondition hold?

---

**Outline**

- Hoare logic
  - Hoare Triples
  - Rules for backward reasoning
    - Assignment, sequence, if-then-else, method calls
- Lab 1 (optional: 2pt extra homework credit)
  - The Dafny programming language
  - Submit your Dafny program in a *.txt file by midnight tomorrow

---

Fall 17 CSCI 2600, A. Milanova
Hoare Logic

- Formal framework for reasoning about code
- Sir Tony Hoare or Sir C.A.R. Hoare
  - Hoare logic
  - Quicksort algorithm
  - Other contributions to programming languages
  - Turing Award in 1980

Examples of Hoare Triples

{ x>0 } x++ { x>1 } is true
{ x>0 } x++ { x>-1 } is true
{ x≥0 } x++ { x>1 } is false. Why?

{ x>0 } x++ { x>0 } is ??
{ x<0 } x++ { x<0 } is ??
{ x=a } if ( x < 0 ) x=-x { x = | a | } is ??
{ x=y } x=x+3 { x=y } is ??

Hoare Triples

- A Hoare Triple: { P } code { Q }
  - P and Q are logical statements about program values (in our case, mostly interval constraints on values), and code is program code
  - "{ P } code { Q }" means "if P is true and we execute code, then Q is true afterward"
  - "{ P } code { Q }" is a logical formula, just like "0 ≤ index"

Backward Reasoning

// precondition: ??

x = x + 3;

{ 2(x+3) > 5 } or { x > -1/2 }
{ 2x > 5 }
y = 2x;
{ y > 5 }
x = 5;
// postcondition: y > x

Rules for Backward Reasoning: Assignment

// precondition: ??

x = expression
// postcondition: Q

Rule: precondition is Q with all occurrences of x in Q replaced by expression
// precondition: y+1 > 0 // precondition: z+1 > 0
x = y+1; z = z+1;
// postcondition: x>0 // postcondition: z>0

Weakest Precondition

Rule derives the weakest precondition
// precondition: y+1 > 0 (equiv. y > -1)

x = y+1
// postcondition: x>0
(y+1)>0 is the weakest precondition for code x=y+1 and postcondition x>0

Notation: wp stands for weakest precondition
wp("x=expression;",Q) = Q with all occurrences of x replaced by expression
Rule for Assignment

\{ \text{wp("x=expression",Q)} \}
x = \text{expression};
\{ Q \}

Rule: the weakest precondition \( \text{wp("x=expression",Q)} \) is \( Q \) with all occurrences of \( x \) in \( Q \) replaced by \( \text{expression} \)

Aside: Weaker and Stronger Conditions

- "P is stronger than Q" means "P implies Q"
- "P is stronger than Q" means "P guarantees at least as much as Q"
  - E.g., \( y > 0 \) is stronger than \( y > -1 \)

Which one is stronger?
\[ x > 0 \&\& y = 0 \quad \text{or} \quad x > 0 \&\& y \geq 0 \]
\[ 0 \leq x \leq 10 \quad \text{or} \quad 0 \leq x \leq 1 \]
\[ x = 5 \&\& y \% 4 = 2 \quad \text{or} \quad x = 5 \&\& y \text{ is even} \]

Aside: Weaker and Stronger Conditions

Let the following be true:

\[ P \Rightarrow Q \]
\[ Q \Rightarrow R \]
\[ S \Rightarrow T \]
\[ T \Rightarrow U \]

\{ Q \} \ \text{code} \ \{ T \}

Which of the following are true?

1. \{ P \} \ \text{code} \ \{ T \}
2. \{ R \} \ \text{code} \ \{ T \}
3. \{ Q \} \ \text{code} \ \{ S \}
4. \{ Q \} \ \text{code} \ \{ U \}

Aside: Weaker and Stronger Conditions

In backward reasoning, we determine the precondition, given \text{code} and a postcondition \( Q \)
- We find the weakest precondition, \( \text{wp(code,Q)} \)

In forward reasoning, we determine the postcondition, given \text{code} and a precondition \( P \)
- Normally, we want the strongest postcondition

Aside: Weaker and Stronger Conditions

Goal: Prove that \{ P \} \ \text{code} \ \{ Q \} is a valid triple

- Forward reasoning:
  \{ P \}
  \text{code}
  \{ Q' \}
- Backward reasoning:
  \{ P \}
  \text{derive} \ \{ P' \}

Then show \( Q' \Rightarrow Q \)
Then show \( P \Rightarrow P' \)

Rules for Backward Reasoning: Sequence

// precondition: ??
S1; // statement
S2; // another statement
// postcondition: Q

Work backwards:
precondition is \( \text{wp("S1;S2;",Q)} = \text{wp("S1;","S2;",Q)} \)

Example:
// precondition: ??
\[ x = 0; \]
\[ x = 0; \] // postcondition for \( x=0 \); same as
\[ y = x+1; \]
\[ y = x+1; \] // postcondition for \( y=x+1 \);
// postcondition: y>0
// postcondition y>0
Rule for Sequence

// find weakest precondition for sequence S1;S2 and Q

{ wp(S1,wp(S2,Q)) }  
S1; // statement
{ wp(S2,Q) }  
S2; // another statement
{ Q }

Working backwards:
precondition is wp(S1;S2, Q) = wp(S1, wp(S2, Q))

Exercise

{x + 1 + y > 1} equiv. to \{ x + y > 0 \}
x = x + 1;
\{ x + y > 1 \}
y = x + y;
\{ y > 1 \}

If-then-else Statement, Example

// precondition: ??(z>5 & & x>0) || (z<-5 & & x≤0)
if (x > 0) {
    y = z;
} else {
    y = -z;
}
// postcondition: y>5

wp(S1;S2, Q) = wp(S1, wp(S2, Q))

Rules for Backward Reasoning:

If-then-else

// precondition: ??
if (b) S1 else S2
// postcondition: Q

Case analysis, just as we did in the example:

wp("if (b) S1 else S2", Q) = (      b & & wp("S1",Q)
| |    (  b & & wp("S1",Q) )
    || (  b && wp("S2",Q) ) )

Exercise

Precondition: ??
z = 0;
if (x != 0) {
    z = x;
} else {
    z = z+1
}
Postcondition: z > 0;
why not this rule?

\[
\begin{align*}
&\text{wp: ??} \\
&\text{if (b) { \{ S1; \}}; } \\
&\text{else { \{ S2; \};}} \\
&\{ Q \}
\end{align*}
\]

Fall 17 CSCI 2600, A Milanova

Exercise

// precondition: ??
\[
\begin{align*}
y &= x + 4 \\
\text{if (x > 0) { \{ y = x*x - 1; \}}; } \\
\text{else { \{ y = y+x; \};}} \\
\text{Postcondition: \{ y = 0 \} } \\
\text{x = x/y; } \\
\text{// What inputs cause Divide-by-zero?}
\end{align*}
\]

Fall 17 CSCI 2600, A Milanova (based on slide by Michael Ernst)

If-then-else Statement Review

Forward reasoning
\[
\begin{align*}
\{ P \} \\
\text{if (b) \{ wp("S1","Q") \}| | wp("S2","Q") \}} \\
\text{if (b) \{ P \&\& wp("S1","Q") \}} \\
\text{S1 \{ Q1 \}} \\
\text{else \{ wp("S1","Q") \}} \\
\text{S1 \{ Q1 \}} \\
\text{else \{ wp("S2","Q") \}} \\
\text{S2 \{ Q2 \}} \\
\text{else \{ wp("S2","Q") \}} \\
\text{Q1 \| Q2 \}} \\
\end{align*}
\]

Fall 17 CSCI 2600, A Milanova (based on slide by Mike Ernst, UW)

Rules for Backward Reasoning: Method Call

// precondition: ??
\[
\begin{align*}
x &= \text{foo()} \\
\text{// postcondition: Q } \\
\text{If method has no side-effects, just like assignment} \\
\text{// precondition: ??} \\
\text{x = Math.abs(y) } \\
\text{// postcondition: x = 1 } \\
\text{Precondition is y = 1 || y = -1}
\end{align*}
\]

Fall 17 CSCI 2600, A Milanova (based on slide by Mike Ernst, UW)
Summary So Far

- Intro to reasoning about code. Concepts
  - Specifications, preconditions and postconditions, forward and backward reasoning
- Hoare triples
- Rules for backward reasoning
  - Rule for assignment
  - Rule for sequence of statements
  - Rule for if-then-else
- Method call

Lab 1

- Dafny: programming language and verifier
  - Author: K. Rustan M. Leino, Microsoft Research
  - Programmer writes programs with specifications
  - Verifier proves that program obeys specification
- Binary downloads at https://github.com/Microsoft/dafny
- Try online at http://www.rise4fun.com/dafny

Dafny Basics

- The smallest unit of verification is the method
  method Foo(x: int, y: int) returns (z: int, w: int)

  - Preconditions
    requires x == 0 && y >= 0
  - Postconditions
    ensures z != 0 || w != 0

Our Earlier Exercise, in Dafny

```daffny
// precondition: ??
if (x > 0) {
y = x*x - 1;
} else {
y = y+x;
}
{ y = 0 }
x = x/y;
```

Answer:
Precondition: x=1 || x=2
These are the inputs that cause divide-by-zero error

Our Division Example in Dafny

```daffny
method DivisionByZero(x: int) returns (y: int)
requires x == 1 || x == -2
ensures y == 0
{
y := x + 4;
if (x > 0) {
y := x*x - 1;
} else {
y := y + x;
}
}
```

1. Compute the weakest precondition
2. Verify your answer with Dafny
3. Cut and paste your program into file DafnyCode.txt, then submit it to Submitty, Principles of Software Lab 1, Deadline midnight Saturday

Lab Exercise rise4fun.com/Dafny

```daffny
// weakest precondition: ??
if (x < 5) {
y = x*x;
} else {
y = x+1;
}
// postcondition: y ≥ 9
```