Reasoning About Code; Hoare Logic

Announcements
- Homework 0 due on Friday
- Questions?
- Office hours schedule finalized at:
  - www.cs.rpi.edu/~milanova/csci2600

Outline
- Hoare logic
- Hoare Triples
- Rules for backward reasoning
  - Assignment, sequence, if-then-else, method calls
- Lab 1 (optional lab: 2pt extra credit on HW1)
  - The Dafny programming language
  - Submit your Dafny program in a *.txt file by midnight tomorrow

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

Hoare Triples
- A Hoare Triple: { P } code { Q }
  - P and Q are logical statements about program values, and code is program code (in our case, Java 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”

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 ??
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

Fall 15 CSCI 2600, A Milanova (based on slide by Michael Ernst)

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

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

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

Aside: Weaker and Stronger Conditions

"P is stronger than Q" means "P implies Q"
"P is stronger than Q" means "P guarantees more than Q"
  E.g., y>0 is stronger than y>-1
Which one is stronger?
x > 0 && y = 0     or    x > 0 && y ≥ 0
0 ≤ x ≤ 10       or    0 ≤ x ≤ 1
x = 5 && y%4 = 2  or x = 5 && y is even

Aside: Weaker and Stronger Conditions

Let the following be true:
P => Q  Q => R  S => T  T => U
{Q} code {T}
Which of the following are true?
(1){P} code {T}
(2){R} code {T}
(3){Q} code {S}
(4){Q} code {U}

Fall 15 CSCI 2600, A Milanova (based on slide by Michael Ernst)

Aside: Weaker and Stronger Conditions

In **backward reasoning**, we determine the precondition, given code and a postcondition Q
  We want the **weakest precondition**, wp(code,Q)

In **forward reasoning**, we determine the postcondition, given code and a precondition P
  Normally we want the **strongest postcondition**
Aside: Weaker and Stronger Conditions

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

- Forward reasoning:
  \[ \{ P \} \quad \text{code} \quad \{ Q' \} \quad \text{derive} \quad \{ Q \} \]
  Then show \( Q' \Rightarrow Q \)

- Backward reasoning:
  \[ \{ P \} \quad \text{code} \quad \{ Q \} \quad \text{derive} \quad \{ P' \} \]
  Then show \( P \Rightarrow P' \)

Rules for Backward Reasoning: Sequence

- // precondition: ??
  \( S_1 \); // statement
  \( S_2 \); // another statement
  // postcondition: Q
  Work backwards:
  \( \text{precondition is } \wp("S_1;S_2;",Q) = \wp("S_1;",\wp("S_2;",Q)) \)

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

Rule for Sequence

- // find weakest precondition for sequence \( S_1;S_2 \) and Q

  \[ \{ \wp(S_1,\wp(S_2,Q)) \} \]
  \( S_1 \); // statement
  \[ \{ \wp(S_2,Q) \} \]
  \( S_2 \); // another statement
  \[ \{ Q \} \]

  Working backwards:
  \( \text{precondition is } \wp(S_1;S_2,Q) = \wp(S_1,\wp(S_2,Q)) \)

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: ?? \((z>5 \&\& x>0) \mid (z<-5 \&\& x\leq0)\)

\( \text{if} \ (x > 0) \{
\quad y = z;
\}
\text{else}\ {
\quad y = -z;
\}
// postcondition: \( y>5 \)

Rules for Backward Reasoning: If-then-else

- // precondition: ??
  \( \text{if} \ (b) \ S_1 \ \text{else} \ S_2 \)
  // postcondition: Q
  Case analysis, just as we did in the example:
  \( \wp("\text{if} \ (b) \ S_1 \ \text{else} \ S_2",Q) \)
  \( = ( \ (b \&\& \wp("S_1",Q) \)
\quad \| \ (\neg b \&\& \wp("S_2",Q)) \) \)
Rule for If-then-else

// wp: ?? \((b \&\& wp(S1,Q)) \|\| (\neg b \&\& wp(S2,Q))\)
if (b) {
    S1;
} else {
    S2;
} { Q }

Fall 15 CSCI 2600, A Milanova

Exercise

Precondition: ??
z = 0;
if (x != 0) {
    z = x;
} else {
    z = z+1
}
Postcondition: z > 0;

Fall 15 CSCI 2600, A Milanova (example due to Michael Ernst)

Why Not This Rule?

// wp: ??
wp(S1,Q) && wp(S2,Q)
if (b) {
    S1;
} else {
    S2;
} { Q }

Fall 15 CSCI 2600, A Milanova

Exercise

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

Fall 15 CSCI 2600, A Milanova

Why Not This Rule?

// wp: ??
wp(S1,Q) || wp(S2,Q)
if (b) {
    S1;
} else {
    S2;
} { Q }

Fall 15 CSCI 2600, A Milanova

If-then-else Statement Review

Forward reasoning
{ P } \{ b \&\& wp("S1",Q) \|\| \neg b \&\& wp("S2",Q) \}
if (b) if (b)
{ P \&\& b } { wp("S1",Q) } S1 S1
{ Q1 } { Q } else else
{ P \&\& \neg b } { wp("S2",Q) } S2 S2
{ Q2 } { Q } { Q1 || Q2 } { Q }

Fall 15 CSCI 2600, A Milanova (based on slide by Michael Ernst)
If-then Statement

// precondition: ??
if (x > y) {
    z = x;
    x = y;
    y = z;
} // postcondition: x < y

Rules for Backward Reasoning: Method Call

// precondition: ??
x = foo()
// postcondition: Q
If method has no side-effects, just like assignment
// precondition: ??
x = Math.abs(y)
// postcondition: x = 1
Precondition is y = 1 || y = -1

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
- Install (on Windows) from http://dafny.codeplex.com
- 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

// precondition: ??
y = x + 4
if (x > 0) {
    y = x*x - 1;
}
else {
    y = y+x;
}
{ y = 0 }
x = x/y;

Find what input causes divide-by-zero at the last statement.

Answer:
Precondition: x=1 || x=-2
These are the inputs that cause divide-by-zero error
Our Division Example in Dafny

```daml
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;
    }
}
```

Equality test: ==, NOT =
Assignment: :=, NOT =

Lab Exercise

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

Fall '15-CSCI 2600, A Milanova (example due to Michael Ernst)