Reasoning about code

CSE 331
University of Washington

Winter 2013, Section 1
Slides from Autumn 2012

Course Logistics

Take the office hours doodle poll

ex0 done, grades soon

hw1 out, due next Tuesday

Can do with notes + slides from lecture/section

Reasoning about code

Determine what facts are true during execution

`x > 0`

for all nodes `n`: `n.next.previous == n`

array `a` is sorted

`x + y == z`

if `x != null`, then `x.a > x.b`

Applications:

Ensure code is correct (via reasoning or testing)

Find errors

Understand why code is incorrect

Forward reasoning

You know what is true before running the code

What is true after running the code?

Given a precondition, what is the postcondition?

Example:

```
// precondition:  x is even
x = x + 3;
y = 2 * x;
x = 5;
// postcondition:  ??
```

Forward reasoning

You know what is true before running the code

What is true after running the code?

Given a precondition, what is the postcondition?

Example:

```
// precondition:  x is even
x = x + 3;
y = 2 * x;
x = 5;
// postcondition:  x = 5, y is even
```

Forward vs. backward reasoning

Forward reasoning is more intuitive for most people

Helps you understand what will happen (simulates the code)

Introduces facts that may be irrelevant to the goal

- Set of current facts may get large

Takes longer to realize that the task is hopeless

Backward reasoning is usually more helpful

Helps you understand what should happen

Given a specific goal, indicates how to achieve it

- Given an error, gives a test case that exposes it

Backward reasoning

You know what you want to be true after running the code

What must be true beforehand in order to ensure that?

Given a postcondition, what is the corresponding precondition?

Example:

```
// precondition:  ??
x = x + 3;
y = 2 * x;
x = 5;                          { ??? }
// postcondition:  y > x
```

Backward reasoning

You know what you want to be true after running the code

What must be true beforehand in order to ensure that?

Given a postcondition, what is the corresponding precondition?

Example:

```
// precondition:  ??
x = x + 3;
y = 2 * x;
x = 5;                          { y > 5 }
// postcondition:  y > x
```

Backward reasoning

You know what you want to be true after running the code

What must be true beforehand in order to ensure that?

Given a postcondition, what is the corresponding precondition?

Example:

```
// precondition:  ??
x = x + 3;
y = 2 * x;
x = 5;
// postcondition:  y > x
```

{ 2x > 5 }

{ y > 5 }

Backward reasoning

You know what you want to be true after running the code

What must be true beforehand in order to ensure that?

Given a postcondition, what is the corresponding precondition?

Example:

```
// precondition:  ??
x = x + 3;
y = 2 * x;
x = 5;
// postcondition:  y > x
```

```
{ 2(x+3) > 5 } => { 2x > -1 } => { x > -
}
{ 2x > 5 }

{ y > 5 }
```

Backward reasoning

You know what you want to be true after running the code

What must be true beforehand in order to ensure that?

Given a postcondition, what is the corresponding precondition?

Example:

```
// precondition:  x is non-negative
x = x + 3;
y = 2 * x;
x = 5;
// postcondition:  y > x
```

Backward reasoning exercises

```
z = x – y + 2;

z = 3 * z – 6;

{z != 0}
```

```
y =
Math.sqrt(w);

x = 2 * y;

x = x + 1;

{-5 < x < 5}
```

Backward reasoning exercises

```
{        }

z = x – y + 2;

{        }

z = 3 * z – 6;

{z != 0}
```

Backward reasoning exercises

```
{        }

z = x – y + 2;

{ 3z-6 ≠ 0 } => { z ≠ 2 }

z = 3 * z – 6;

{z != 0}
```

Backward reasoning exercises

```
{ 2 ≠ x − y + 2 } => { x ≠ y }

z = x − y + 2;

{ 3z-6 ≠ 0 } => { z ≠ 2 }

z = 3 * z − 6;

{z != 0}
```

Backward reasoning exercises

```
{           }

y = Math.sqrt(w);

{           }

x = 2 * y;

{           }

x = x + 1;

{-5 < x < 5}
```

Backward reasoning exercises

```
{           }

y = Math.sqrt(w);

{           }

x = 2 * y;

{ -5 < x+1 < 5 } => { -6 < x < 4 }

x = x + 1;

{-5 < x < 5}
```

Backward reasoning exercises

```
{            }

y = Math.sqrt(w);

{ -6 < 2y < 4 } => { -3 < y < 2 }

x = 2 * y;

{ -5 < x+1 < 5 } => { -6 < x < 4 }

x = x + 1;

{-5 < x < 5}
```

Backward reasoning exercises

```
{ -3 < Math.sqrt(w) < 2 } =>
  { 0 <= Math.sqrt(w) < 2 } => { 0 <= w < 4 }
y = Math.sqrt(w);

{ -6 < 2y < 4 } => { -3 < y < 2 }

x = 2 * y;

{ -5 < x+1 < 5 } => { -6 < x < 4 }

x = x + 1;

{-5 < x < 5}
```

Reasoning with if statements

```
{P}

if (B) {

  S1;

} else  {

       S2;

}

{Q}
```

```
{P}

if (B) {
  {P && B}
  S1;
      {Q1}
} else  {
  {P && !B}
      S2;
      {Q2}
}

{Q1 || Q2 => Q}
```

Reasoning with if statements example

```
assert x >= 0;
      // x ≥ 0
z = 0;
                        // x ≥ 0 & z = 0
if (x != 0) {

   z = x;

} else {

   z = z + 1;

}

assert z > 0;
```

Using forward reasoning: Does the postcondition hold?

Reasoning with if statements example

```
assert x >= 0;
        // x ≥ 0
z = 0;
                              // x ≥ 0 & z = 0
if (x != 0) {
        // x > 0 & z = 0
   z = x;
        // x > 0 & z = x
} else {
        // x = 0 & z = 0
   z = z + 1;
        // x = 0 & z = 1
}
        // (x > 0 & z = x) || (x = 0 & z = 1)
assert z > 0;
```

Using forward reasoning: Does the postcondition hold?

Reasoning with if statements example

```
assert x >= 0;
      // x ≥ 0
z = 0;
                          // x ≥ 0 & z = 0
if (x != 0) {
      // x > 0 & z = 0
   z = x;
      // x > 0 & z = x
} else {
      // x = 0 & z = 0
   z = z + 1;
      // x = 0 & z = 1
}
      // (x > 0 & z = x) || (x = 0 & z = 1)
assert z > 0;
```

Using forward reasoning: Does the postcondition hold? **Yes!**

Reasoning with if statements exercise

```
{ true }

if (x > y) {

    m = x;

} else {

  m = y;

}

{ m = max(x,y) }
```

Reasoning with if statements exercise

```
{ true }

if (x > y) {
  { true && x > y }
    m = x;

} else {

  m = y;

}

{ m = max(x,y) }
```

Reasoning with if statements exercise

```
{ true }

if (x > y) {
  { true && x > y }
    m = x;
  { x > y && m = x }
} else {

  m = y;

}

{ m = max(x,y) }
```

Reasoning with if statements exercise

```
{ true }

if (x > y) {
  { true && x > y }
    m = x;
  { x > y && m = x }
} else {
  { true && x <= y }
  m = y;

}

{ m = max(x,y) }
```

Reasoning with if statements exercise

```
{ true }

if (x > y) {
  { true && x > y }
    m = x;
  { x > y && m = x }
} else {
  { true && x <= y }
  m = y;
  { x ≤ y && m = y }
}

{ m = max(x,y) }
```

Reasoning with if statements exercise

```
{ true }

if (x > y) {
  { true && x > y }
    m = x;
  { x > y && m = x }
} else {
  { true && x <= y }
  m = y;
  { x ≤ y && m = y }
}

{(x > y && m = x) || (x ≤ y && m = y)
    => m = max(x,y) }
```

Fix the precondition

```
{ true }


if (x > 0) {
    z =
Math.pow(y,x);
} else {
    z = y - x;
}


{z >= y}
```

Fix the precondition

```
{ true }

if (x > 0) {
  { true && x > 0 }
    z = Math.pow(y,x);
  { x > 0 && z = y ^ x } => { x > 0 && _____ }
} else {
  { true && x <= 0 }
    z = y - x;
    { x < 0 && z = y – x } => { z >= y }
}

{ (x > 0 && ____) || (x <= 0 && z >= y)
    => z >= y }
```

Fix the precondition

```
{ true }

if (x > 0) {
  { true && x > 0 }
    z = Math.pow(y,x);
  { x > 0 && z = y ^ x } Want { z >= y }
} else {
  { true && x <= 0 }
    z = y - x;
    { x < 0 && z = y – x } => { z >= y }
}

{ (x > 0 && z >= y) || (x <= 0 && z >= y)
    => z >= y }
```

Fix the precondition

```
{ y >= 0 || x is even }

if (x > 0) {
  { true && x > 0 }
    z = Math.pow(y,x);
  { x > 0 && z = y ^ x } => { z >= y }
} else {
  { true && x <= 0 }
    z = y - x;
    { x < 0 && z = y – x } => { z >= y }
}

{ (x > 0 && z >= y) || (x <= 0 && z >= y)
    => z >= y }
```

More backward reasoning exercises

```
{ ???? }
y = x;
y = y + 1;
{ y > x }
```

```
{ ???? }
y = y + 3;
x = 2 * y;
z = x + 8;
{ z > 2w }
```

More backward reasoning exercises

```
{ x+1 > x } => { true }

y = x;

{ y+1 > x }

y = y + 1;

{ y > x }
```

```
{ y+3+4 > w }
    => { y+7 > w }
y = y + 3;
{ 2y+8 > 2w }
    => { y+4 > w }
x = 2 * y;
{ x+8 > 2w }
z = x + 8;
{ z > 2w }
```