Hoare Logic: Loops & Framing

COS 441 Slides 12

Agenda

- Last few lectures
  - Hoare Logic:
    - { P } C { Q }
    - If P is true in the initial state s. And C in state s evaluates to s'. Then Q must be true in s'.
  - Rules of Hoare logic:
    - rule of consequence
    - assignment rule, skip rule, sequence rule, if rule
- This time:
  - While Loops

HOARE LOGIC:
WHILE LOOPS

While Statements

- Rule for while statements

If ???

then { P } while (e > 0) do C { Q }

While Statements

- Bogus rule for while statements

If { P & e > 0 } C { Q }

then { P } while (e > 0) do C { Q }

While Statements

- Bogus rule for while statements

If { P & e > 0 } C { Q }

then { P } while (e > 0) do C { Q }

basic problem:
this rule only
captures 1 iteration
of the loop,
not all of them

```
{ i = N & a = 0 }
while (i > 0) do
   a = a + K;
   i = i - 1;
{ a = N * K }
```

While Statements

- Bogus rule for while statements

If { P & e > 0 } C { Q }

then { P } while (e > 0) do C { Q }

```
{ i = N & a = 0 }
while (i > 0) do
    a = a + K;
    i = i - 1;
{ a = N * K }
```

must prove →

```
{ i = N & a = 0 & i > 0 }
    a = a + K;
    i = i - 1;
{ a = N * K }
```

While Statements

- Bogus rule for while statements

If { P & e > 0 } C { Q }

then { P } while (e > 0) do C { Q }

```
{ i = N & a = 0 }
while (i > 0) do
    a = a + K;
    i = i - 1;
{ a = N * K }
```

must prove →

```
{ i = N & a = 0 & i > 0 }
    a = a + K;
    i = i - 1;
{ a = N * K }
```

this isn't even
close to a valid triple!
With that precondition,
a = K at the end!

While Statements

- Problem: We need to verify all iterations of a loop and we need to do it with a finite amount of work
- Solution: We will come up with an invariant that holds at the beginning and end of all iterations.
  - We prove that the loop body preserves the invariant *every* time around
- Unfortunate reality: Inferring invariants automatically is undecideable.
  - This puts significant limits on the degree to which we can automate verification.

While Statements

- While rule:

loop invariant I

If P => I and { e > 0 & I } C { I } and I & ~(e > 0) => Q

then { P } while (e > 0) do C { Q }

While Statements

- While rule:

loop invariant I

If P => I and { e > 0 & I } C { I } and I & ~(e > 0) => Q

then { P } while (e > 0) do C { Q }

- Inference rule notation:

$$\frac{P \Rightarrow I \qquad \{\, e > 0 \;\&\; I \,\}\ C\ \{\, I \,\} \qquad I \;\&\; \sim(e > 0) \Rightarrow Q}{\{\, P \,\}\ \text{while } (e > 0) \text{ do } C\ \{\, Q \,\}}$$

While Statements

P => I { e > 0 & I } C { I } I & ~(e > 0) => Q

{ P } while (e > 0) do C { Q }

{i = 7 & a = 0} while (i > 0) do a = a+K; i = i-1; {a = 7*K}

While Statements

$$\frac{P \Rightarrow I \qquad \{ e > 0 \,\&\, I \}\ C\ \{ I \} \qquad I \,\&\, \sim(e > 0) \Rightarrow Q}{\{ P \}\ \text{while } (e > 0) \text{ do } C\ \{ Q \}}$$

$$\frac{i = 7 \,\&\, a = 0 \Rightarrow I \qquad \{i > 0 \,\&\, I\}\ a = a+K;\ i = i-1;\ \{I\} \qquad I \,\&\, \sim(i > 0) \Rightarrow a = 7*K}{\{i = 7 \,\&\, a = 0\}\ \text{while } (i > 0) \text{ do } a = a+K;\ i = i-1;\ \{a = 7*K\}}$$

While Statements

P => I   { e > 0 & I } C { I }   I & ~(e > 0) => Q

{ P } while (e > 0) do C { Q }

What works as I?

- true initially
- true before/after each iteration
- must imply Q when loop terminates

i = 7 & a = 0 => I   {i > 0 & I} a = a+K; i = i-1; {I}   I & ~(i > 0) => a = 7*K

{i = 7 & a = 0} while (i > 0) do a = a+K; i = i-1; {a = 7*K}

While Statements

P => I   { e > 0 & I } C { I }   I & ~(e > 0) => Q

{ P } while (e > 0) do C { Q }

Invariant I is (a = (7-i)*K) & i >= 0

What works as I?

- true initially
- true before/after each iteration
- must imply Q when loop terminates

i = 7 & a = 0 => I   {i > 0 & I} a = a+K; i = i-1; {I}   I & ~(i > 0) => a = 7*K

{i = 7 & a = 0} while (i > 0) do a = a+K; i = i-1; {a = 7*K}

While Statements

$$\frac{P \Rightarrow I \qquad \{ e > 0 \,\&\, I \}\ C\ \{ I \} \qquad I \,\&\, \sim(e > 0) \Rightarrow Q}{\{ P \}\ \text{while } (e > 0) \text{ do } C\ \{ Q \}}$$

Invariant I is (a = (7-i)*K) & i >= 0

Checking I:

- i = 7 & a = 0 => (a = (7-i)*K) & i >= 0

substitute:
7 for i
0 for a

7 >= 0

0 = (7 - 7)*K

$$\frac{i = 7 \,\&\, a = 0 \Rightarrow I \qquad \{i > 0 \,\&\, I\}\ a = a+K;\ i = i-1;\ \{I\} \qquad I \,\&\, \sim(i > 0) \Rightarrow a = 7*K}{\{i = 7 \,\&\, a = 0\}\ \text{while } (i > 0) \text{ do } a = a+K;\ i = i-1;\ \{a = 7*K\}}$$

While Statements

$$\frac{P \Rightarrow I \qquad \{ e > 0 \;\&\; I \}\ C\ \{ I \} \qquad I \;\&\; \sim(e > 0) \Rightarrow Q}{\{ P \}\ \text{while } (e > 0) \text{ do } C\ \{ Q \}}$$

Invariant I is (a = (7-i)*K) & i >= 0

Checking I:

- i = 7 & a = 0 => (a = (7-i)*K) & i >= 0
- (a = (7-i)*K) & i >= 0 & ~(i > 0) => a = 7*K

$$\frac{i = 7 \;\&\; a = 0 \Rightarrow I \qquad \{i > 0 \;\&\; I\}\ a = a+K;\ i = i-1;\ \{I\} \qquad I \;\&\; \sim(i > 0) \Rightarrow a = 7*K}{\{i = 7 \;\&\; a = 0\}\ \text{while } (i > 0) \text{ do } a = a+K;\ i = i-1;\ \{a = 7*K\}}$$

While Statements

P => I { e > 0 & I } C { I } I & ~(e > 0) => Q

{ P } while (e > 0) do C { Q }

Invariant I is (a = (7-i)*K) & i >= 0

Checking I:

- i = 7 & a = 0 => (a = (7-i)*K) & i >= 0
- (a = (7-i)*K) & i >= 0 & ~(i > 0) => a = 7*K

i=0

i = 7 & a = 0 => I {i > 0 & I} a = a+K; i = i-1; {I} I & ~(i > 0) => a = 7*K

{i = 7 & a = 0} while (i > 0) do a = a+K; i = i-1; {a = 7*K}

While Statements

$$\frac{P \Rightarrow I \qquad \{ e > 0 \,\&\, I \}\ C\ \{ I \} \qquad I \,\&\, \sim(e > 0) \Rightarrow Q}{\{ P \}\ \text{while } (e > 0) \text{ do } C\ \{ Q \}}$$

Invariant I is (a = (7-i)*K) & i >= 0

Checking I:

- i = 7 & a = 0 => (a = (7-i)*K) & i >= 0
- (a = (7-i)*K) & i >= 0 & ~(i > 0) => a = 7*K
- validate the triple: {i > 0 & I} a = a+K; i = i-1; {I}

$$\frac{i = 7 \,\&\, a = 0 \Rightarrow I \qquad \{i > 0 \,\&\, I\}\ a = a+K;\ i = i-1;\ \{I\} \qquad I \,\&\, \sim(i > 0) \Rightarrow a = 7*K}{\{i = 7 \,\&\, a = 0\}\ \text{while } (i > 0) \text{ do } a = a+K;\ i = i-1;\ \{a = 7*K\}}$$

While Statements

$$\frac{P \Rightarrow I \qquad \{ e > 0 \ \& \ I \}\ C\ \{ I \} \qquad I \ \& \ \sim(e > 0) \Rightarrow Q}{\{ P \}\ \text{while } (e > 0) \text{ do } C\ \{ Q \}}$$

Invariant I is (a = (7-i)*K) & i >= 0

$$\frac{i = 7 \ \& \ a = 0 \Rightarrow I \qquad \dfrac{\{i > 0 \ \& \ I\}\ a = a+K\ \{ P \} \qquad \{ P \}\ i = i-1;\ \{I\}}{\{i > 0 \ \& \ I\}\ a = a+K;\ i = i-1;\ \{I\}} \qquad I \ \& \ \sim(i > 0) \Rightarrow a = 7*K}{\{i = 7 \ \& \ a = 0\}\ \text{while } (i > 0) \text{ do } a = a+K;\ i = i-1;\ \{a = 7*K\}}$$

While Statements

P => I { e > 0 & I } C { I } I & ~(e > 0) => Q

{ P } while (e > 0) do C { Q }

Invariant I is (a = (7-i)*K) & i >= 0

a = (7-(i-1))*K & i-1 >= 0

{i > 0 & I} a = a+K { P } { P } i = i-1; {I}

---

i = 7 & a = 0 => I {i > 0 & I} a = a+K; i = i-1; {I} I & ~(i > 0) => a = 7*K

{i = 7 & a = 0} while (i > 0) do a = a+K; i = i-1; {a = 7*K}

While Statements

$$\frac{P \Rightarrow I \qquad \{ e > 0 \,\&\, I \}\ C\ \{ I \} \qquad I \,\&\, \sim(e > 0) \Rightarrow Q}{\{ P \}\ \text{while } (e > 0) \text{ do } C\ \{ Q \}}$$

Invariant I is (a = (7-i)*K) & i >= 0

a + K = (7-(i-1))*K & i-1 >= 0

a = (7-(i-1))*K & i-1 >= 0

$$\frac{i > 0 \,\&\, I \Rightarrow Q \qquad \dfrac{}{\{Q\}\ a = a+K\ \{ P \}}}{\{i > 0 \,\&\, I\}\ a = a+K\ \{ P \}}$$

$$\frac{\{i > 0 \,\&\, I\}\ a = a+K\ \{ P \} \qquad \{ P \}\ i = i-1;\ \{I\}}{\{i > 0 \,\&\, I\}\ a = a+K;\ i = i-1;\ \{I\}}$$

$$\frac{i = 7 \,\&\, a = 0 \Rightarrow I \qquad \{i > 0 \,\&\, I\}\ a = a+K;\ i = i-1;\ \{I\} \qquad I \,\&\, \sim(i > 0) \Rightarrow a = 7*K}{\{i = 7 \,\&\, a = 0\}\ \text{while } (i > 0) \text{ do } a = a+K;\ i = i-1;\ \{a = 7*K\}}$$

While Statements

$$\frac{P \Rightarrow I \qquad \{ e > 0 \,\&\, I \}\ C\ \{ I \} \qquad I \,\&\, \sim(e > 0) \Rightarrow Q}{\{ P \}\ \text{while } (e > 0) \text{ do } C\ \{ Q \}}$$

Invariant I is (a = (7-i)*K) & i >= 0

$$i > 0 \,\&\, a=(7-i)*K \,\&\, i >=0 \Rightarrow a + K = (7-(i-1))*K \,\&\, i-1 >= 0$$

$$\frac{\dfrac{i > 0 \,\&\, I \Rightarrow Q \qquad \dfrac{}{\{Q\}\ a = a+K\ \{ P \}}}{\{i > 0 \,\&\, I\}\ a = a+K\ \{ P \}} \qquad \{ P \}\ i = i-1;\ \{I\}}{\{i > 0 \,\&\, I\}\ a = a+K;\ i = i-1;\ \{I\}}$$

$$\frac{i = 7 \,\&\, a = 0 \Rightarrow I \qquad \{i > 0 \,\&\, I\}\ a = a+K;\ i = i-1;\ \{I\} \qquad I \,\&\, \sim(i > 0) \Rightarrow a = 7*K}{\{i = 7 \,\&\, a = 0\}\ \text{while } (i > 0) \text{ do } a = a+K;\ i = i-1;\ \{a = 7*K\}}$$

While Statements

$$\frac{P \Rightarrow I \qquad \{ e > 0 \,\&\, I \}\ C\ \{ I \} \qquad I \,\&\, \sim(e > 0) \Rightarrow Q}{\{ P \}\ \text{while } (e > 0) \text{ do } C\ \{ Q \}}$$

Invariant I is (a = (7-i)*K) & i >= 0

i > 0 & a=(7-i)*K & i >=0 => a + K = (7-(i-1))*K & i-1 >= 0

== a + K = (7-i)*K + K & i-1 >= 0

== a = (7-i)*K & i-1 >= 0

$$\frac{\dfrac{i > 0 \,\&\, I \Rightarrow Q \qquad \dfrac{}{\{Q\}\ a = a+K\ \{ P \}}}{\{i > 0 \,\&\, I\}\ a = a+K\ \{ P \} \qquad \{ P \}\ i = i-1;\ \{I\}}}{\{i > 0 \,\&\, I\}\ a = a+K;\ i = i-1;\ \{I\}}$$

$$\frac{i = 7 \,\&\, a = 0 \Rightarrow I \qquad \{i > 0 \,\&\, I\}\ a = a+K;\ i = i-1;\ \{I\} \qquad I \,\&\, \sim(i > 0) \Rightarrow a = 7*K}{\{i = 7 \,\&\, a = 0\}\ \text{while } (i > 0) \text{ do } a = a+K;\ i = i-1;\ \{a = 7*K\}}$$

Another Example

- What is the loop invariant?

{x = 0} while (x < 15) { x = x + 2 } { even(x) }

Another Example

- What is the loop invariant?

{x = 0} while (x < 15) { x = x + 2 } { even(x) }

Invariant: even(x)

- x = 0 => even(x)
- even(x) & x >= 15 => even(x)
- { even(x) & x < 15 } x = x + 2 {even (x) }

Another Example

- What is the loop invariant?

$\{x = 0 \ \& \ y = N \ \& \ N > 0\}$ while $(y > 0)$ $\{ y = y - 1; x = x + 1 \}$ $\{ x = N \}$

Another Example

- What is the loop invariant?

{x = 0 & y = N & N > 0} while (y > 0) { y = y - 1; x = x + 1 } { x = N }

Invariant: x + y = N

Another Example

- What is the loop invariant?

{x = 0 & y = N & N > 0} while (y > 0) { y = y - 1; x = x + 1 } { x = N }

Invariant: x + y = N

- x = 0 & y = N & N > 0 => x + y = N

Another Example

- What is the loop invariant?

{x = 0 & y = N & N > 0} while (y > 0) { y = y - 1; x = x + 1 } { x = N }

Invariant: x + y = N

- x = 0 & y = N & N > 0   =>   x + y = N
- y <= 0 & x + y = N   =>   x = N

nope!

Another Example

- What is the loop invariant?

$\{x = 0 \;\&\; y = N \;\&\; N > 0\}$ while $(y > 0)$ $\{ y = y - 1; x = x + 1 \}$ $\{ x = N \}$

Invariant: $x + y = N \;\&\; y >= 0$

- $x = 0 \;\&\; y = N \;\&\; N > 0 \quad => \quad x + y = N \;\&\; y >= 0$
- $y <= 0 \;\&\; x + y = N \;\&\; y >=0 \quad => \quad x = N$

add constraint to invariant

more difficult to establish invariant initially

easier to establish postcondition

Another Example

- What is the loop invariant?

$\{x = 0 \;\&\; y = N \;\&\; N > 0\}$ while $(y > 0)$ $\{ y = y - 1; x = x + 1 \}$ $\{ x = N \}$

Invariant: $x + y = N \;\&\; y >= 0$

- $x = 0 \;\&\; y = N \;\&\; N > 0 \quad => \quad x + y = N \;\&\; y >= 0$
- $y <= 0 \;\&\; x + y = N \;\&\; y >=0 \quad => \quad x = N$

add constraint to invariant

easier to establish postcondition

more difficult to establish invariant initially

Another Example

- What is the loop invariant?

{x = 0 & y = N & N > 0} while (y > 0) { y = y - 1; x = x + 1 } { x = N }

Invariant: x + y = N & y >= 0

- x = 0 & y = N & N > 0 => x + y = N & y >= 0
- y <= 0 & x + y = N & y >=0 => x = N
- { x+y = N & y >= 0 & y > 0 } y = y-1; x = x+1 {x+y = N & y >= 0 }

While Statements: Summary

- Given a Hoare triple for a while loop:
  - { P } while (e > 0) do C { Q }
- We prove it correct by:
  - guessing an invariant I (this is the hard part)
  - proving I holds initially: P => I
  - showing the loop body preserves I:
    - { e > 0 & I } C { I }
  - showing the postcondition holds on loop termination:
    - I & ~(e > 0) => Q
- As a rule:

$$\frac{P \Rightarrow I \qquad \{\, e > 0 \,\&\, I \,\}\ C\ \{\, I \,\} \qquad I \,\&\, {\sim}(e > 0) \Rightarrow Q}{\{\, P \,\}\ \text{while}\ (e > 0)\ \text{do}\ C\ \{\, Q \,\}}$$

- Note: one often adds I as an annotation on the loop:
  - while [I] (e > 0) do C

FRAMING & MODULARITY

Another Issue: Framing

- Another valid triple:

$\{ x = 9 \ \& \ y = 7 \ \& \ z = 23\} \ x = x + 1 \ \{ x = 10 \ \& \ y = 7 \ \& \ z = 23\}$

- Proving it using the rules:

Another Issue: Framing

- Another valid triple:

{ x = 9 & y = 7 & z = 23} x = x + 1 { x = 10 & y = 7 & z = 23}

- Proving it using the rules:

(1) { x + 1 = 10 & y = 7 & z = 23 } x = x + 1 { x = 10 & y = 7 & z = 23} (assignment rule)

(2) x = 9 & y = 7 & z = 23 => x + 1 = 10 & y =7 & z = 23 (implication)

(3) { x = 9 & y = 7 & z = 23} x = x + 1 { x = 10 & y = 7 & z = 23} (by (1), (2), consequence)

Another Issue: Framing

- Another valid triple:

$$\{x = 9 \ \& \ y = 7 \ \& \ z = 23\} \ x = x + 1 \ \{x = 10 \ \& \ y = 7 \ \& \ z = 23\}$$

- Proving it using the rules:

(1) $\{x + 1 = 10 \ \& \ y = 7 \ \& \ z = 23\} \ x = x + 1 \ \{x = 10 \ \& \ y = 7 \ \& \ z = 23\}$ (assignment rule)

(2) $x = 9 \ \& \ y = 7 \ \& \ z = 23 \ \Rightarrow \ x + 1 = 10 \ \& \ y = 7 \ \& \ z = 23$ (implication)

(3) $\{x = 9 \ \& \ y = 7 \ \& \ z = 23\} \ x = x + 1 \ \{x = 10 \ \& \ y = 7 \ \& \ z = 23\}$ (by (1), (2), consequence)

- Note: Formulae not involving x are just propagated
- More generally, conjuncts not involving variables that are not *modified* are just propagated
- Can we factor those expressions out of most of the proof?

The Simple Frame Rule

- The Simple Frame Rule (also called the rule of constancy)

$$\frac{\{\ P\ \}\ C\ \{\ Q\ \} \qquad \text{C does not modify the (free) variables of } R}{\{\ P\ \&\ R\ \}\ C\ \{\ Q\ \&\ R\ \}}$$

- What counts as "modifying"?
  - In our simple language, the only way a variable may be modified is if it appears on the left in an assignment statement
  - In languages with functions or methods, calling one of them may have a modification effect
  - In C, you might be able to intentionally modify variables on the stack
  - In C, you might also have a buffer overflow ... yikes!
- The frame rule is a way of *simplifying* proofs
- Why are Haskell proofs so easy? Nothing is modified!

The Simple Frame Rule

- The Simple Frame Rule (also called the rule of constancy)

$$\frac{\{ P \}\ C\ \{ Q \} \qquad \text{C does not modify the (free) variables of } R}{\{ P \ \&\ R \}\ C\ \{ Q \ \&\ R \}}$$

- Example:

$$\frac{}{\{ x = 6 \ \&\ y = 7 \ \&\ z = 23\}\ x = x + 1;\ x = x * 2;\ x = x - 4;\ \{ x = 10 \ \&\ y = 7 \ \&\ z = 23\}}$$

The Simple Frame Rule

- The Simple Frame Rule (also called the rule of constancy)

$$\frac{\{\, P \,\}\; C\; \{\, Q \,\} \qquad \text{C does not modify the (free) variables of } R}{\{\, P \,\&\, R \,\}\; C\; \{\, Q \,\&\, R \,\}}$$

- Example:

$$\frac{\{\, x = 6 \,\}\; x = x + 1;\, x = x * 2;\, x = x - 4;\; \{\, x = 10 \,\} \qquad \begin{array}{l} x = x + 1;\, x = x * 2;\, x = x - 4; \\ \text{does not modify } y \text{ or } z \end{array}}{\{\, x = 6 \,\&\, y = 7 \,\&\, z = 23 \,\}\; x = x + 1;\, x = x * 2;\, x = x - 4;\; \{\, x = 10 \,\&\, y = 7 \,\&\, z = 23 \,\}}$$

SUMMARY!

Summary

- States map variables to values
- Formulae describe states:
  - semantics in Haskell: fsem :: State -> Form -> Maybe Bool
  - semantics in Math: [[f]]s
  - formulae and states we deal with are well-formed
    - well-formedness is a very simple syntactic analysis
  - P => Q means P describes a subset of the states that Q does
- Hoare Triples characterize program properties
  - { P } C { Q } – know when it is valid
  - know the statement rules you can use to conclude { P } C { Q }
  - understand the structural rules:
    - rule of consequence
    - frame rule
  - know how to build formal proofs using inference rules