

Hoare Logic, continued Reasoning About Loops

Announcements

- HW0 due today
 - Commit to SVN then Submit in the HW Server
- HW1 will be up after class
 - Check Homeworks for announcement. Then **Update** src in Eclipse
- **QUIZ 1 today. You have 10 minutes**

2

Backward Reasoning: Rule for Assignment

```
{ 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**

3

Backward Reasoning: 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 }
```

4

Simple Example

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

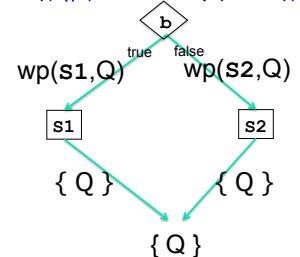
Spring 15 CSCI 2600, A Milanova

5

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

6

Proving Correctness

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

- Backward reasoning
- Forward reasoning:

$\{P\}$	$\{P\}$
derive $\{P'\}$	code
code	derive $\{Q'\}$
$\{Q\}$	$\{Q\}$

Then show $P \Rightarrow P'$ Then show $Q' \Rightarrow Q$

Fall 15 CSCI 2600, A Milanova

7

What Happens When There Is a Loop

Does the postcondition hold?

Precondition: $x \geq 0$;

```

i = x;           { x >= 0 && i = x }
z = 0;           { x >= 0 && i = x && z = 0 }
while (i != 0) {
    z = z+1;      ???
    i = i-1;      ???
}
Postcondition: x = z;
    
```

Yes.

The key is to guess the **loop invariant**.

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

8

Outline

- Reasoning about loops
- Total correctness = **partial correctness** + **termination**
- Loop invariants**
- Computation induction (to prove partial correctness)
- Decrementing function (to prove termination)

Fall 15 CSCI 2600, A Milanova

9

Reasoning About Loops

- Reasoning about loops is difficult
 - Unknown number of iterations and unknown number of paths. Recursion poses similar issues
 - We cannot enumerate all paths
 - Key is to guess a **loop invariant**
 - Two things to prove about loops
 - It computes correct values (**partial correctness**)
 - It **terminates** (does not go into infinite loop)
- total correctness = partial correctness + termination

10

Example: Partial Correctness + Termination

Precondition: $x \geq 0$;

```

i = x;
z = 0;
while (i != 0)
    inv i+z = x
{
    z = z+1;
    i = i-1;
}
    
```

Postcondition: $x = z$;

I. If the loop terminated does $x = z$ hold?

II. Does loop terminate?

- $i=x$ and $z=0$ give us that $i+z = x$ holds at 0th iteration of loop // **Base case**
- Assuming that $i+z = x$ holds after k^{th} iteration, we show it holds after $(k+1)^{\text{st}}$ iteration // **Induction**
 $z_{\text{new}} = z + 1$ and $i_{\text{new}} = i - 1$ thus
 $z_{\text{new}} + i_{\text{new}} = z + 1 + i - 1 = z + i = x$
- If loop terminated, we know $i = 0$. Since $z+i = x$ holds, we have $z = x$

- Loop terminates. The precondition $x \geq 0$ guarantees $i \geq 0$ before loop. At every iteration i decreases by 1, thus it eventually reaches 0.

11

Reasoning About Loops by Induction

- $i+z = x$ is a **loop invariant** (a fact that holds true before and after each loop iteration)
 - Even though i and z change, $i+z=x$ stays true
- We just made an inductive argument over the number of iterations of the loop
 - Computation induction
 - Established that loop invariant holds at 0th iteration
 - Assuming that loop invariant holds after k^{th} iteration, show that it holds after $(k+1)^{\text{st}}$ iteration

12

Reasoning About Loops by Induction

1. Partial correctness

- Establish and prove **loop invariant** using computation induction
- Loop exit condition** and **loop invariant** must imply the desired postcondition
 - $i = 0$ (loop exit condition) and $i+z = x$ imply $z = x$

2. Termination

- (Roughly) Establish “decrementing function” D . $D \geq 0$ before loop, each iteration decrements D , loop invariant and $D = 0$ imply loop exit condition

- We will discuss Partial Correctness first

13

Another Example Partial Correctness + Termination

Let us prove that **sum** is correct:

Precondition: $len \geq 0 \ \&\& \ arr.length = len$

```
int sum = 0;
int i = 0;
while (i < len) {
    sum = sum + arr[i];
    i = i+1;
}
Postcondition: sum = arr[0]+...+arr[arr.length-1]
```

Fall 15 CSCI 2600, A Milanova

14

Another Example: Partial Correctness

$\sum_{j: 0 \text{ to } i-1} arr[j]$ stands for $arr[0]+arr[1]+\dots+arr[i-1]$

Loop invariant: $i \leq len \ \&\& \ sum = \sum_{j: 0 \text{ to } i-1} arr[j]$

- $i \leq len \ \&\& \ sum = \sum_{j: 0 \text{ to } i-1} arr[j]$ holds before loop
- Assume $sum = \sum_{j: 0 \text{ to } i-1} arr[j]$ holds after k^{th} iteration


```
sum_new = sum + arr[i];
i_new = i+1;
```

Thus $sum = \sum_{j: 0 \text{ to } i-1} arr[j]$ holds after $(k+1)^{st}$ iteration
 $i \leq len$ also holds (had we had $i = len$ after k^{th} iteration, there wouldn't have been a $(k+1)^{st}$ iteration!)

15

Another Example: Partial Correctness

$\sum_{j: 0 \text{ to } i-1} arr[j]$ stands for $arr[0]+arr[1]+\dots+arr[i-1]$

- $\neg (i < len)$ which is $i \geq len$ plus loop invariant
 $i \leq len \ \wedge \ sum = \sum_{j: 0 \text{ to } i-1} arr[j]$
 imply
 $sum = \sum_{j: 0 \text{ to } len-1} arr[j]$, which with precondition
 $arr.length=len$ gives us desired postcondition
 $sum = arr[0]+...+arr[arr.length-1]$

We still must argue termination!

16

Another Example: Termination

Precondition: $len \geq 0 \ \&\& \ arr.length = len$

```
int sum = 0.0;
int i = 0;
while (i < len) {
    sum = sum + arr[i];
    i = i+1;
}
```

- We now argue termination (informally).
 At each iteration i increases by one, while len stays the same.
 Thus, eventually i reaches len .
 $i = len$ implies loop exit condition.

More on termination later!

Fall 15 CSCI 2600, A Milanova

17

Partial Correctness, More Formally

$\{P\} \text{ while } (b) \text{ s } \{Q\}$

We need to “guess” a **loop invariant** Inv such that

- $P \Rightarrow Inv$ // Inv holds before loop. Base case
- $\{b \ \&\& \ Inv\} \text{ s } \{Inv\}$ // Assuming Inv held after k^{th} iteration and execution took a $(k+1)^{st}$ iteration, then Inv holds after $(k+1)^{st}$ iteration
- $(\neg b \ \&\& \ Inv) \Rightarrow Q$ // The exit condition and loop invariant imply desired postcondition

Fall 15 CSCI 2600, A Milanova

18

Choosing Loop Invariant

What is a suitable loop invariant?

Precondition: $x \geq 0$;

$i = x$;

$z = 0$;

while ($i \neq 0$) {

$z = z+1$;

$i = i-1$;

}

Postcondition: $x=z$

Inv: $i+z = x$

Fall 15 CSCI 2600, A Milanova

19

Choosing Loop Invariant

What is a suitable loop invariant?

Precondition: $x \geq 0 \ \&\& \ y = 0$

while ($x \neq y$) {

$y = y+1$;

}

Postcondition: $x=y$

Inv: $y \leq x$

Fall 15 CSCI 2600, A Milanova

20

Choosing Loop Invariant

What is a suitable loop invariant?

Precondition: $len \geq 0$

$\&\& \ arr.length = len$

int sum = 0;

int i = 0;

while ($i < len$) {

 sum = sum + arr[i];

 i = i+1;

}

Postcondition: sum = arr[0]+...+arr[arr.length-1]

Inv: $i \leq len \ \&\& \ sum = \sum_{j: 0 \text{ to } i-1} arr[j]$

Fall 15 CSCI 2600, A Milanova

21

Choosing Loop Invariant

What is a suitable loop invariant?

Precondition: $n \geq 0$

i = 0;

r = 1;

while ($i < n$) {

 i = i+1;

 r = r*i;

}

Postcondition: $r = n!$

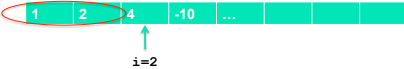
Inv: $i \leq n \ \&\& \ r = i!$

Fall 15 CSCI 2600, A Milanova

22

A "tedious" Invariant

Inv: sum = a[0]+a[1]



Precondition: $len \geq 0 \ \&\& \ a.length = len$

int sum = 0;

int i = 0;

while ($i < len$) {

 invariant sum = a[0]+...+a[i-1] $\&\& \ i \leq len$

 sum = sum + a[i];

 i = i+1;

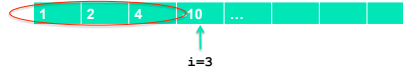
}

Postcondition: sum = a[0]+...+a[a.length-1]

23

A "tedious" Invariant

Inv: sum = a[0]+a[1]+a[2]



Precondition: $len \geq 0 \ \&\& \ a.length = len$

int sum = 0;

int i = 0;

while ($i < len$) {

 invariant sum = a[0]+...+a[i-1] $\&\& \ i \leq len$

 sum = sum + a[i];

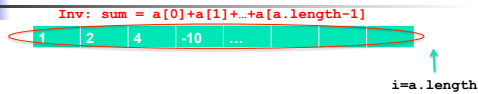
 i = i+1;

}

Postcondition: sum = a[0]+...+a[a.length-1]

24

A “tedious” Invariant



As the induction variable i moves through the array, sum contains the correct partial result.

At the end of the array, sum contains the correct total

Similar invariants for min, max, avg, sorting algorithms, etc.
These invariants are easy to guess.

Fall 15 CSCI 2600, A Milanova

25

An “elegant” Invariant

What is a suitable loop invariant?

Precondition: $x \geq 0$;

$i = x$;

$z = 0$;

Inv: $i+z = x$

while ($i \neq 0$) {

$z = z+1$;

$i = i-1$;

}

Postcondition: $x=z$

Fall 15 CSCI 2600, A Milanova

26

An “elegant” Invariant

Precondition: $x \neq 0$

$zeros = 0$;

$y = x$;

while ($y \% 10 == 0$) {

$y = y/10$;

$zeros = zeros + 1$;

}

Postcondition: $x = y \cdot 10^{zeros} \ \&\& \ y \% 10 \neq 0$

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

27

Termination, More Formally

- We must “guess” a **decrementing function D**

- Maps state to natural numbers: 0, 1, 2, ...

{ P } while (b) s { Q }

We need D, with range in the natural numbers, such that

- $\{ \text{Inv} \ \&\& \ b \} \ s \ \{ D_{\text{after}} < D_{\text{before}} \}$ // Execution of loop reduces the value of D
- $(\text{Inv} \ \&\& \ D=0) \Rightarrow \neg b$ // Inv + D reaching 0 (D’s minimal value), must imply the loop exit condition

Fall 15 CSCI 2600, A Milanova

28

Choosing Decrementing Function

What is a suitable decrementing function?

Precondition: $x \geq 0$;

$i = x$;

$z = 0$;

D = i

while ($i \neq 0$) {

$z = z+1$;

$i = i-1$;

}

Postcondition: $x=z$

Fall 15 CSCI 2600, A Milanova

29

Choosing Decrementing Function

What is a suitable decrementing function?

Precondition: $x \geq 0 \ \& \ y = 0$

while ($x \neq y$) {

$y = y+1$;

}

Postcondition: $x=y$

D = x - y

Fall 15 CSCI 2600, A Milanova

30

Choosing Decrementing Function

What is a suitable decrementing function?

Precondition: $\text{len} \geq 0$

$\wedge \text{arr.length} = \text{len}$

$D = \text{len} - i$

double sum = 0.0;

int i = 0;

while (i < len) {

sum = sum + arr[i];

i = i+1;

}

Postcondition: $\text{sum} = \text{arr}[0] + \dots + \text{arr}[\text{arr.length}-1]$

Fall 15 CSCI 2600, A Milanova

31

Choosing Decrementing Function

What is a suitable decrementing function?

Precondition: $n \geq 0$

i = 0;

r = 1;

while (i < n) {

i = i+1;

r = r*i;

}

Postcondition: $r = n!$

Fall 15 CSCI 2600, A Milanova

32

Choosing Decrementing Function

Precondition: $x \neq 0$

zeros = 0;

y = x;

while (y%10 == 0) {

y = y/10;

zeros = zeros + 1;

}

Postcondition: $x = y \cdot 10^{\text{zeros}}$ && $y\%10 \neq 0$

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

33

Reasoning About Loops is Difficult

- For non-looping code, weakest precondition enables proof
- For loops, we have to guess
 - The loop invariant
 - The decrementing function
- Then use the proof techniques we discussed
- If the proof doesn't work
 - Maybe you chose wrong invariant/function: Fix
 - Maybe the loop is incorrect!

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

34

Why Study Formal Proofs

- Helps us write **correct** programs!
- Most of the time, our loops don't need proofs

```
for i in seq: print i
```
- Sometimes we have to write more complex loops
 - Establish precondition and postcondition
 - Establish a loop invariant
 - Reason about correctness and termination

Fall 15 CSCI 2600, A Milanova

35

Reasoning About Loops, Recap

total correctness = partial correctness + termination

1. Partial correctness

- "Guess" and prove **loop invariant**
- Loop exit condition** and **loop invariant** must imply the postcondition
- This gives us: "If the loop terminated then the postcondition holds". But does the loop terminate?

2. Termination

- (Roughly) "Guess" **decrementing function** D. Each iteration decrements D, D at its minimum value (usually 0) must imply loop exit condition

36

Example: Another Factorial

Precondition: $n \geq 0$

```
r = 1;
n = t;
while (n != 0) {
    r = r*n;
    n = n-1;
}
```

Postcondition: $r = t!$

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

37

Example: Integer Division x/y

Precondition: $x \geq 0 \ \&\& \ y \geq 0$

```
r = x;
q = 0;
while (y <= r) {
    r = r-y;
    q = q+1;
}
```

Postcondition: $x = y*q + r \ \&\& \ r < y$

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

38

Example: Greatest Common Divisor

Precondition: $x1 > 0 \ \&\& \ x2 > 0$

```
y1 = x1;
y2 = x2;
while (y1 != y2) {
    if (y1 > y2) {
        y1 = y1-y2;
    } else {
        y2 = y2-y1;
    }
}
```

Postcondition: $y1 = \text{gcd}(x1, x2)$

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

39

Fall 15 CSCI 2600, A Milanova

40