Week 14 - Day 2 (Conditions / Invariants)

Table of Contents

[CS 403 - 001 Spring 2016](/CS403-001/)

[About](/CS403-001/about/)

# Week 14 - Day 2 (Conditions / Invariants)

Apr 21, 2016

Download Word (docx):

## Navigate using audio

# Final Exam

* May 2nd
* 11 am - 2 pm
  + He’s going to draw pictures of tables and environments and ask us questions about them
    - You have to accurately draw an environment / picture
    - Which environments are garbage collected?
    - Which closures are garbage collected?
  + Next Thursday is optional
* Drop dead project date is Saturday at 12 am

// precondition  
// Audio 0:08:06.024352   
<<x > 0 && y == 0>>  
while (x > 0) {   
 x = x - 1;  
 y = y + 1;  
}  
// #x is original value of x  
<<x==0 && y==#x>>

We need to find invariants

* Invariants
  + x + y = #x && x >= 0 && y >= 0
  + Turns out we don’t need the last one
    - Audio 0:10:31.969762
    - Turns out we don’t need the last one
* Want to make these as simple as possible
  + Audio 0:09:05.838218

P (Precondition)

Q (Postcondition)

I (Invariant)

S (Function body)

* Given our facts of the precondition
  + We need to show that it implies the invariant is true
* What if you had a precondition of false?
  + You could have anything as the invariant because all hypotheses are true
    - Or something like that
* Finding invariant (step 0) Audio 0:12:36.331760
* Step 1 P -> I
* Step 2<<I>> E <<I>>
  + I push back through E, I should imply the invariant through E
* Step 3 `«I && E» S «I»
  + After I pushback through S, I and E should imply the invariant after being pushed back through S
* Setp 4 I && !E -> Q

## Running through it

* Audio 0:18:33.355059
* Step 1:
  + x >=0 && y==0 -> x + y == #x && x >=0
    - is their a difference between x an #x right now?
      * No, so we can simplify
  + x >=0 && y==0 -> x + y == x && x >=0

x + 0 = x  
x = x (check)

* Step 2:
  + E causes no changes of state ✓
* Step 3:
  + Audio 0:21:13.875265
    - I && E -> I ⨂ S

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

I ⨂ S x + y + 1 == #x && x >= 0  
 x - 1 + y + 1 == #x && x - 1 >= 0

Audio 0:23:36.306506

x + y ==#x && x >= 0 && x > 0 -> x + y == #x && x >= 1

Audio 0:24:05.223440

We have two conditions on x. Can it be simplified?  
x + y ==#x && x >= 0 && x > 0 -> x + y == #x && x >= 1  
 ---------------  
 ^ these are both true when x > 0, meaning x >= 1  
  
We have shown 3 is true ✓

* Step 4
  + Audio 0:25:59.289925
  + x + y == #x && x >= 0 && x <= 0

x == 0 && y == #x  
-----  
 ✓

* Audio 0:27:43.042313
  + Humans don’t determine if programs terminate
* I’m sorry about bringing up Formal languages.

## Another example

* Audio 0:33:01.950746

<<x % 2==0 && y==0 && x >= 0>>  
while (x > 0) {  
 x = x - 2;  
 y = y + 1;  
}  
<<x==0 && y== #x / 2>>

### What are our invariants? (Things that are always true)

* x > 0
* x + 2y == #x
* x % 2 == 0
  + It starts out even and we just subtract 2 from it
* y >= 0
  + Audio 0:37:35.525937
* The above are true, but they might not be minimal
  + We have to show at the end that I && !#E -> Q
    - So I’m dumb, we need x + 2y == #x
* Audio 0:40:49.999159
* Re-write the second term ( x + 2y == #x )?
  + x + 2(#x/2) == #x
  + No, doesn’t help us prove y == #x / 2
* Only one we don’t need is y >= 0
  + Audio 0:42:37.440301

## IF statement example

* Audio 0:47:07.386269

<< a >= 0 && b > 0 && t == 1>>  
while (b > 0) {  
 if (b % 2 == 0) {  
 a = a \* a;  
 b = b / 2;  
 }  
 else {  
 t = t \* a;  
 b = b - 1;  
 }  
}  
<<b == 0 && t == #a^#b>>

### What are the invariants?

* Audio 0:50:20.104086
* b >= 0
  + b is counting down and eventually 1 will be subtracted from 1 and the loop will fail
    - b will be equal to 0 at that point
* a >= 0
* t >= 0
  + starts out at 1, t can go to zero if a is 0 to begin with
* how many steps to calculate #a^#b
  + log(b)
* what value is being preserved?
  + trying to compute #a^#b
  + storing stuff in t and if I’m not, I’m making a bigger and bigger
    - Audio 0:56:19.321853
    - (a^2)^(b/2) -> a^b
  + t + a^b = #a^#b

Ex: Test 2, b = 2

a b t  
- - - 1 + 2^3 = 9  
2 3 1  
  
2 2 2 2 + 2^2 = 6

* t \* a^b = #a^#b

a b t  
- - - 1 \* 2^3 = 8  
2 3 1  
  
2 2 2 2 \* 2^2 = 8  
  
4 1 2 2 \* 4^1 = 8  
  
4 0 8 8 \* 4^0 = 8

### What is necessary?

* b >= 0
* t \* a^b == #a^#b
  + Figuring out what was preserved helped us figure this out

# Next time we’ll go over finding the weakest precondition for a while loop

## CS 403 - 001 Spring 2016

* CS 403 - 001 Spring 2016
* [jmbeach1@crimson.ua.edu](mailto:jmbeach1@crimson.ua.edu)
* jmbeach

Website for notes and study material for CS 403 (Programming Languages) at The University of Alabama Spring 2016