ECS 240: Homework 3

Sugeerth Murugesan

May 20, 2014

1 Abstract interpretation

1.1 Program analysis

The result can be improved by changing the factorial program rather than the analysis.

By removing the minus operator, the analysis is more precise: One way to do this is when we have the counter variable starting from bottom up.

$y := 1, i := 1$
while $i \leq x$ **do**
  $i := i + 1$
  $y := y * i$
end while

Since every statement of the algorithm is positive it is easier to prove that value of y is positive.

1.2 Lattice structure

To get a precise result we just change the lattice structure,

In the new lattice, the configuration for $\top$, $+$, and $\perp$ are the same as before, but ≥ 0 is a gamma function where $\gamma(\geq 0) = \{x | x \geq 0\}$. Since that is a superset of the positive integers, we must have $\geq 0 \ > \ +$.

Tracing the program we get:

Line 1:$A(x) = +$, $A(y) = \top$

Line 2:$A(x) = +$, $A(y) = +$

Line 3:$A(x) = +$, $A(y) = +$ (We know that it sums upto the same constraints in point 2, also in the point 4 where multiplying + leaves us with these signs)

Line 2:(second iteration):$A(x) = \geq 0$, $A(y) = +$(After executing i=i-, we should increase the number of values that is possible for x.This tracnsition moves from + to ≥ 0)

Line 3:(Abstract values) $A(x) = +$, $A(y) = +$

Line 5: $A(x) = \geq 0$, $A(y) = +$ (At this point we have i==0 which means that we dont distinguish betweeen x == 0 x ≥ 0)

Thus we have proved that we have reached a fixed point and the output y is positive if the program terminates.

2 Abstract interpretation

Given γ function we can write the definition of α and prove that α is monotonic and forms a Galois connection with γ.

The solution for α involves a gratest lower bound operation. To prove the first condition means that we will have to satisfy the first requirement of the Galois connection. γ must be monotonic. We take into account

γ : A $\rightarrow$ P(C) The restriction is that α and γ are monotonic. So if $S_1 \subseteq S_2$, α(S_1) $\subseteq$ α(S_2)
To prove that α and γ follow a Galois connection -

$\alpha(\gamma$(a)) =a for all a $\in$ A.

For an arbitrary a, by the definition of γ and α-
$\alpha(\gamma$(a))=glb {b | $\gamma(b) \supseteq \gamma(a)$ }
= glb {b | b $\geq$ a} =a
Thus the properties of the Galios connection are proved.

3 Hoarre's Logic

3.1 Loop invariant

Simple Rule for loop invariants:

To Prove the following property using Hoares Logic:

{ x mod 2 $\neq$ 0} while (b) do x := x+2 { x mod 2 $\neq$ 0}

The Rules of inference for the looping variable would be:

Proving the loop invariant using the following Hoare's triplet:

$$\frac{\{\ \mathrm{A} \wedge \mathrm{b}\ \}\ \mathrm{c}\ \mathrm{A}}{\{\ \mathrm{A}\ \}\ \text{while (b) do x := x+2}\ \{\ \neg\ \mathrm{b} \wedge \mathrm{A}\ \}}$$

Hoare's triple:
{ A } while (b) do x := x+2 { $\neg$ b $\wedge$ A }

3.2 Loop invariants

{ n $\geq$ 0 }
t := n;
{ n $\geq$ 0 $\wedge$ t = n }
r:= 1
{ n $\geq$ 0 $\wedge$ t = n $\wedge$ r = 1 }

while t $\neq$ 0 **do**
{ r =n!/t! $\wedge$ n > t > 0 }
r:= r * t;
{ r =n!/(t-1)! $\wedge$ n > t > 0 }
t:=t-1;
end while
{ r = n! }

Proof Obligations

{ n $\geq$ 0 $\wedge$ t=n } r : = 1 { P_1 }
{ P_1 } $\rightarrow$ { P_2 }
{ P_2 $\wedge$ (t $\neq$ 0) } r : = r * t; { P_3 }
{ P_3 } t= t-1; { P_2 }
{ P_2 $\wedge$ $\neg$(t $\neq$ 0)} $\rightarrow$ r : = n!

3.3 GCD

{ a > 0 $\wedge$ b > 0 }
x:= a y:=b
{ Q $\equiv$ x > 0 $\wedge$ y > 0 $\wedge$ gcd(a,b) $\equiv$ gcd (x,y)}
{ VA $\equiv$ x + y }
while x$\neq$ y **do**
{ Q $\wedge$ x$\neq$y $\wedge$ (x + y = z) }
if x $\geq$ y **then**
x:=x-y
else
y:=y-x;
{ Q $\wedge$ x$\neq$y $\wedge$ (x + y < z) }
end if
end while
{ Q $\wedge$ (x = y) }
{ gcd(a,b) = x }

3.4 Command and assertion

{ A } c { true }

Let us take for example the idea of having a structural induction over the command c.We need to prove it for 4 different cases which is SKIP, WHILE, ASSIGNMENT, ; ; .

4 Alternate rules

4.1 Hoare rule

Using the rule of consequence the new rule can be derived from the original rule.

While b do c

If $\vdash$ A $\wedge$ b $\Longrightarrow$ C and $\vdash$ {A $\wedge$ b } c { A } then { C } c { A }

If $\vdash$ A $\wedge$ $\neg$b $\Longrightarrow$ B and $\vdash$ {A } *While* b *do* c { A $\wedge$ $\neg$b } then { A } *While* b *do* c { B }

4.2 Hoare rule

4.3 Hoare rule

{ true } while true do skip { false }

This counterexample relies on the fact that true would never change into false. This gives a stronger post-condition.

4.4 Hoare rule

The system of axioms remain complete if we replace the old while with the new rule.

{ x=0} while false x:=x+1 {x=0}

is a counter example for this case. It means that as long as the preconditions are equal to the post conditions the command "c" would not matter.

References

[1] http://www.cis.upenn.edu/ bcpierce/sf/HoareAsLogic.html *Hoare's Logic*

[2] http://www.cis.upenn.edu/ bcpierce/sf/HoareAsLogic.html *Hoare's Logic*

[3] http://cs.au.dk/ amoeller/talks/hoare.pdf *hoare logic*

[4] http://www.cs.cmu.edu/ aldrich/courses/654-sp07/slides/7-hoare.pdf *Factorial Hoare rules*