Software Specifications
Loop Verification Examples

Cain Susko

Queen's University
School of Computing

March 14, 2022

Example

```
x, y, n, int

ASSERT(n >= 0)
i = 0; y = 1;

while (i != n){
         y = y * x;
         i++;
} // end while
ASSERT(y == power(x,n))
```

to verify this assertion we must find the invariant within the loop. that is, a value that does not change within the loop and gives the post condition after the loop. the value that satisfies these conditions is:

```
y == power(x, i) && 0 <= i <= n
```

where `y` is the value and its bounds are specified to the right.

Example

consider the following program:

```
y, k, n int;

ASSERT(n <= 0)
k = n; y = 1;

while (k > 0){
         y = y * x;
         k = k - 1;
 } // end while
 ASSERT(y == power(x, n))
```

thus the invariant is:

```
0 <= k <= n && FINISH THIS WITH SLIDE
```

Proof Tableau

this is the proof tableau for the above program:

```
y, k, n int;

ASSERT(n <= 0)
ASSERT(0 <= n <= n && 1 == power(x, 0))
k = n;
ASSERT(0 <= k <= n && 1 == power(x, n-k)
y = 1;
ASSERT(I)

while (k > 0){ ASSERT(I && k > 0)
// k > 0 implies 0 <= k-1 when k is int
// y == power(x, n-k) implies y * x == power(x, n-k+1)
         ASSERT(0 <= k-1 <= n && y * x == power(x, n-k+1)
         y = y * x;
         ASSERT(0 <= k-1 <= n && y * x == power(x, n-k+1)
         k = k - 1;
         ASSERT(I)
 } // end while
 ASSERT(I && k <= 0)
 // k <= 0 && 0 <= k implies k == 0, k == 0 && I implies post-condition
 ASSERT(y == power(x, n))
```

Note, the invariant `I` is `FUCK YOU WITH THE GODDAMN PAPER GET SOME FUNCING SLIDES LIKE WERE IN THE 21st CENTURY`
Termination: according to the invariant $k \leq n$ and each iteration of the loop should FINISH WITH SLIDES

Program Termination

checking wether a while loop terminates is a n algorithmically unsolvable problem. At the end of the this course we will discull unsolvability.
for a "well structured" while loop we can argue termination by including suitable bounds for the loop variable in the invariant.

Algorithm for Verification

Finally, the algorith for verifying the proof tableau of a while loop is as follows:

i Select an invariant and show that the loop body preserves the invariant.

ii Show that the invariant holds before entering into the loop

iii Show that invariant and negation of loop condition implies the post contidion–possibly after some termiation assignments.

iv Argue that the loop terminates–typically done by including the bounds for the loop variable in the invariant

Guidelines for Invariant

- often the invariant can be an generalization of the post-condition – replace some constant in the post-condition by one in the loop variable.
- normally we add bounds for the loop variable in the invariant (for arguing termination)

More Example

consider the following:

```
ASSERT(0 <= n <= max)
{ int i;
present = false;
i = 0;

while (i != n){
        if(A[i] == x) present = true; i++;
} // end while
ASSERT(present iff x in A[0:n-1])
```

Thus, the invariant is just a generalization of the post-condition:

```
present iff x in A[0:i-1] && 0 <= i <= n
```

where the part on the right are the bounds for termination.