Software Specifications
Verifying If Statement

Cain Susko

Queen's University
School of Computing

March 18, 2022

Verifying If Statements

If Part

```
ASSERT(I && i != n && A[i] != B[i])
// i <= n && i != n implies i + 1 <= n when i is int
// A[i] != B[i] implies that
ASSERT(0 <= i+1 <= n && false iff ForAll(k=0, k<i+1)A[b] = -B[k])
        equal = false
ASSERT(0 <= i + 1 < = n && eual iff FofAll(k=0; k < i+1)A[k]==B[k])
```

Else Part

```
I && i != n && A[i] == B[i] implies (*) 0 <= i+1 <= n &&
        equal iff ForAll(k=0, k<i+1) A[k] == B[k]

(*) i <= n && i != n implies i+1 <= n when i,n are ints
        (equal iff ForAll(k=0, k<i) A[k] == B[k]) && A[i] == B[i]
        implies equal iff ForAll(k = 1, n < i+1)A[k] == B[k]
```

Verifying For Loops

For Loops are written in C like so: `for(i=n; i<N; i++)C` . The general form of a for loop is represented as:

$$for(A_o; B; A_1)$$

A verification works like so:

```
ASSERT(P) // pre-condition
        ...
A_o;
ASSERT(I) // I invariant
while(B){
        ASSERT(I&&B)
                CA_1;
        }
ASSERT(I&&!B)
```

Consider the following example:

```
ASSERT(0 <= n < max){
        int i;
        for (i = 0; A[i] != x &&i < n; i++)
                {}
        present = (i<n)
ASSERT(present iff x in A[0:n-1])
```

to verify it we must frist re write it as a while loop:

```
ASSERT(0 <= n < max){
        int i;
        for (i = 0; A[i] != x &&i < n; i++)
                {i++;}
        ASSERT(i < n iff x in A[0:n-1]) // while loop post condition
        present = (i<n)
ASSERT(present iff x in A[0:n-1])
```

The Invariant `I` is bounded by `0 <= i <= n` from the while loop bounds but what is the invariant it's self. The idea is we continue the execution of the loop untill we find the target element, thus the first `i` entiries are not equal to `x`, therefore the invariant is:

$$0 \geq i \geq n \wedge \{\forall_k k > i\} \;\; x \neq A[k]$$