Software Specifications
Verification of Conditional Statements in Specifications

Cain Susko

Queen's University
School of Computing

March 11, 2022

If Else Verification

there are 2 parts to verification, given the example:

```
ASSERT(y == y0 && z == z0)

if(y <= z)   {y = z+1; z = z+2;} else
             {z = y+2; y = y+1;}

ASSERT(y == y0 && z == z0 && y <= z)
```

If Part FINISH WITH SLIDES ON ONQ

If Verification

there is also a case of the if statement when there is no else part. this is represented as:

$$P\ if\ (B)\ C\}\ Q$$

the inference rule for this statement is:

$$\frac{P\ \&\&\ B\ \{C\}\ Q \qquad P\ \&\&!\ B\{\ \}\ Q}{P\ \{if\ (B)\ C\}\ Q}$$

Example

```
ASSERT(z <= y)
if (w > z || y > x) {w = z-1; x = y;}
ASSERT(w <= z <= y <= x)
```

The If Part of the Verification looks like so:

```
ASSERT(z <= y && (w > z || y > x))
// z <= implies
 FINISH USING SLIDE ON ONQ
```

The non-existant else part would go like so:

```
z <= y && !(w > z || y > x)
z <= y && w <= z && y <= x
```

in which the last line implies the post-condition in the if part.

Verifying While Loops

To verify while loops a special assertion must be used. it is called the *loop-invariant*. The loop-invariant has the following formula:

$$\frac{I \;\&\&\; B \;\{C\}\; I}{I \;\{while(b)\; C\;\}\; I \;\&\&\; !B}$$

verifying whie loops has the following proof tableau scheme:

```
ASSERT(I)
        while(B){
                ASSERT(I && B)
                        C
                ASSERT(I)
                } // end while
        ASSERT(I && !B)
```

Observe: the inner assertion block (I && B) -¿ (I) is the permise for the proof. This proof also has a side condition that the evaluation of B should not change the state

What Should we use as Invariant there is a certian art to choosing which part of the code fragment to use as invariant, which must be done in order to verify the while loop

Example The best way to get gud is through doing so here is an example:

```
ASSERT(true)
i = 0; j = 100;
while(i <= 100){

        i = i + 1;
        j = j - 1;} // end while
ASSERT(i == 101 && j == -1)
```

Here, the invariant is `i + j == 100` as this statement is always true within the loop. the syntax for writing this within our above code would be:

```
INVAR(i + j == 100 && 0 <= i <= 101)
```

Note: `0 <= i <= 101` is also invariant as it is always true within the loop