15-122: Principles of Imperative Computation

Recitation 11 Josh Zimmerman

Iterative vs. recursive factorial

Consider the following implementations of the factorial function, and try to prove that it satisfies its postcondition.

```
int factIter(int n)
//@requires n >= 0;
{
   // You can assume that this function is correctly implemented.
   // That is, you can assume factIter(n) is equal to n!
}

int factRec(int n)
//@requires n >= 0;
//@ensures \result == factIter(n);
{
   if (n == 0) {
      return 1;
   }
   else {
      return n * factRec(n - 1);
   }
}
```

Solution:

Partial correctness.

Base case First, we consider the base case. When `n == 0`, we know that we return 1, which is $0!$, so it's equal to `factIter(0)`.

Inductive hypothesis Next, we assume that `factRec(k)` satisfies the postcondition for some `int k` where `k >= 0`, or in other words that the result of `factRec(k)` is equal to `factIter(k)`.

Inductive step Now, we consider `factRec(k + 1)`. Since `k >= 0`, we know `k + 1 > 0`.

Therefore, we'll be in the `else` case and will return `(k + 1) * factRec(k + 1 - 1)`, which is equal to `(k + 1) * factRec(k)`. We're allowed to make this call since we know that `k + 1 > 0` and so `k >= 0`.

By the inductive hypothesis, `factRec(k)` is equivalent to `factIter(k)` and by the definition of factorial (and the assumption that `factIter` is correct) `(k + 1) * factIter(k)` is equal to `factIter(k + 1)`.

Thus, the function has partial correctness.

Termination:

We've shown that if the function terminates, it is correct, but we need to show that the function terminates.

By the precondition, we know that `n >= 0`.

Base case We also know that if `n == 0` then we terminate immediately.

Inductive hypothesis Assume that `factRec(k)` terminates for some `k >= 0`, where `k` is an `int`.

Inductive step Then, consider `factRec(k + 1)`. We recurse and call `factRec(k)`. By our inductive hypothesis, `factRec(k)` terminates, so therefore `factRec(k + 1)` terminates as well.

Thus, for all `n >= 0`, this function terminates.