CIS 575. Introduction to Algorithm Analysis
Material for February 12, 2024

Proving Correct an Iterative Algorithm on Integers

1 Loop Invariants for Correctness Proofs

In the previous note, we considered programs of the form

P
while G
    B

To prove the correctness of such a program, wrt. a precondition *Pre* and a postcondition *Post*, we need to guess a loop invariant Φ and then verify each item of the checklist

1. (**Establish**) $\{Pre\}\ P\ \{\Phi\}$
2. (**Maintain**) $\{\Phi \wedge G\}\ B\ \{\Phi\}$
3. (**Correctness**) $\Phi \wedge \neg G$ logically implies *Post*
4. (**Terminate**) if a state satisfies Φ then G will eventually become false.

Here $\{Q\}\ C\{R\}$ means that executing C in a state that satisfies Q will result in a state that satisfies R.
In this and subsequent notes we shall illustrate this checklist through several, increasingly advanced, examples.

2 Verifying a Loop Manipulating Integers

Recall from the first week that we considered the *fibonacci* function

$$\begin{aligned} \mathsf{fib}(0) = \mathsf{fib}(1) &= 1 \\ \mathsf{fib}(n+2) &= \mathsf{fib}(n) + \mathsf{fib}(n+1) \text{ for } n \geq 0 \end{aligned}$$

and claimed that it may be implemented by the *iterative* program

$i, j \leftarrow 1, 1$
for $k \leftarrow 1$ **to** $n - 1$
  $i, j \leftarrow j, i + j$
return j

We shall now verify that claim (for $n \geq 1$). For that purpose, we need to come up with a loop invariant; we suggest (perhaps based on simulating the program on small values of n)

$$j = \mathsf{fib}(k) \text{ and } i = \mathsf{fib}(k-1) \text{ and } 1 \leq k \leq n$$

It is useful to observe that a loop

for $q \leftarrow a$ **to** b
  C

(where C does not modify a, b or q) is equivalent to

$q \leftarrow a$
while $q \leq b$
  C
  $q \leftarrow q + 1$

We can then embark on verifying the 4 items.

Establish When the loop test is first evaluated, we have $i = j = k = 1$. We must thus show $1 = \mathsf{fib}(1)$ and $1 = \mathsf{fib}(0)$ which follows from the definition of fib, and $1 \leq 1 \leq n$ which follows from our assumption.

Maintain We must prove that for each iteration of the loop body, if the invariant holds for the initial values of i, j, k then it also holds for the final values of i, j, k. We thus need to distinguish between the "old" value and the "new" value of a variable; for that purpose, it is a convenient notation to use primes to denote the new values. Using that notation, we have

$$i' = j, \; j' = i + j, \; k' = k + 1$$

and our assumption is that the invariant holds for the old values: $j = \mathsf{fib}(k)$ and $i = \mathsf{fib}(k-1)$ and $1 \leq k$; also, as the loop guard is true, we have $k \leq n - 1$. We must prove that the invariant holds for the new values: $j' = \mathsf{fib}(k')$ and $i' = \mathsf{fib}(k' - 1)$ and $1 \leq k' \leq n$ which follows from the calculations

$$\begin{aligned} j' &= i + j = \mathsf{fib}(k-1) + \mathsf{fib}(k) = \mathsf{fib}(k+1) = \mathsf{fib}(k') \\ i' &= j = \mathsf{fib}(k) = \mathsf{fib}(k'-1) \\ k' &= k + 1 \leq (n-1) + 1 = n \end{aligned}$$

Correctness Observe that at loop exit we have $k = n$ (to see this formally, recall that the `for` loop is equivalent to a `while` loop with guard $k \leq n-1$ so at exit we have $k > n-1$ which together with $k \leq n$ from the invariant implies $k = n$). From $j = \mathsf{fib}(k)$ we thus infer $j = \mathsf{fib}(n)$ which justifies that j is returned.

Terminate `for` loops always terminate.