Proving the Partial Correctness of a Simple Algorithm with a While Loop

Solutions for a Suggested Exercise

In this exercise you were asked to consider the following computational problem.

Sum of Array Elements

Precondition: An integer array A with some positive length n is given as input.

Postcondition: The value

$$\sum_{i=0}^{n-1} \mathtt{A}[i]$$

is returned as output.

You were also asked to consider the following algorithm (as an algorithm that can be used to solve the above problem).

```
integer arraySum ( integer[] A ) {
1. integer sum := A[0]
2. integer i := 0
3. while (i < A.length − 1) {
4.   i := i + 1
5.   sum := sum + A[i]
   }
6. return sum
}
```

1. You were asked to prove that the following is a ***loop invariant*** for the `while` loop in this algorithm.

   Loop Invariant:

   (a) `A` is an input integer array with some positive length n.

   (b) `i` is an integer variable such that $0 \leq \texttt{i} \leq n-1$.

   (c) `sum` is an integer variable such that

   $$\texttt{sum} = \sum_{j=0}^{i} \texttt{A}[\, j\,].$$

   Solution: Loop Theorem #1 will be used to prove this.

   - The execution of the loop test at line 3, "$\texttt{i} < \texttt{A.length} - 1$", has no side-effects. That is, it does not modify the value of any input, variable, or global data.
   - Consider an execution of this algorithm beginning with the precondition of the "Sum of Array Elements" problem satisfied, so that an integer array `A` with some positive length n has been given as input. The execution of the algorithm begins with the executions of steps 1 and 2 before the `while` loop is reached. Note that this loop is only executed one: Either the first execution of the loop fails to terminate at all, or step 6 is executed and the execution of the algorithm also ends after the first execution of the loop ends.

     Since neither step 1 nor step 2 modifies the array `A` (or its length, n) part (a) of the claimed "loop invariant" holds at the beginning of the execution of the loop, because it is implied by the precondition for the computational problem being solved.

     Since `i` is declared to be an integer variable $0 \leq n-1$ when step 5 is executed, part (b) of the claimed "loop invariant" holds at the beginning of the execution of the loop as well.

     Since `sum` is declared to be an integer variable with value

     $$\texttt{A}[0] = \sum_{j=0}^{\texttt{i}} \texttt{A}[j]$$

     at line 1, and the value of `sum` is not changed by the execution of the step at line 2, part (c) is also satisfied at the beginning of the execution of the loop.
   - Consider an execution of the body of the loop when the claimed "loop invariant" is satisfied at the beginning of this execution of the loop body.

     It follows by part (a) of the claimed "loop invariant" that `A` is an integer array with some positive length n at the beginning of this execution of the body of the loop.

Neither A nor its length, n, are changed when the steps at lines 4 and 5, so this is still true, and part (a) of the claimed "loop invariant" is still satisfied, when this execution of the body of the loop ends.

It follows by part (b) of the claimed "loop invariant" that i is an integer variable such that $0 \leq \texttt{i} \leq n-1$ when this execution of the body of the loop begins. Furthermore, since the loop test at line 3 has just been checked and passed, $\texttt{i} < n-1$ and, since the value of i and n are both integers, $0 \leq \texttt{i} \leq n-2$ at this point.

The value of i is increased by one at line 4, so that $1 \leq \texttt{i} \leq n-1$ after this step has been executed. Since step 5 does not change the value of i, $1 \leq \texttt{i} \leq n-1$ — and part (b) of the claimed "loop invariant" is satisfied — at the end of this execution of the loop body.

Part (c) of the claimed "loop invariant"' implies that sum is an integer variable with value

$$\sum_{j=0}^{\texttt{i}} \texttt{A}[j]$$

at the beginning of this execution of the loop body. Since the value of i is increased by one when the step at line 4 is executed, sum has value

$$\sum_{j=0}^{\texttt{i}-1} \texttt{A}[j]$$

after the execution of this step. However, since the value of sum is increased by A[i] when the step at line 5, sum is an integer variable with value

$$\left(\sum_{j=0}^{\texttt{i}-1} \texttt{A}[j] \right) + \texttt{A}[\texttt{i}] = \sum_{j=0}^{\texttt{i}} \texttt{A}[j],$$

and part (c) of the claimed "loop invariant" is also satisfied, at the end of this execution of the loop body.

Thus the claimed "loop invariant" is satisfied, once again, at the end of this execution of the body of the loop.

It now follows by Loop Theorem #1 that the claimed "loop invariant" is correct. That is, it really *is* a loop invariant for the while loop in this algorithm.

2. You were then asked to use this, as needed, to prove that the arraySum is partially correct (when considered as an algorithm for the "Sum of Array Elements" problem).

   Solution: Consider an execution of this algorithm that begins with the precondition for the "Sum of Array Elements" problem satisfied — so that A is an integer array with some positive length n.

One can see, by inspection of the algorithm, that the algorithm has no extra inputs, does not access any undocumented global data, and makes no undocumented changes to data: The signature of the algorithm can be examined to confirm that the algorithm has no extra inputs, and the executable statements at lines 1–6 can be examined to confirm that these other properties are also satisfied.

The execution of the algorithm begins with the execution with the steps at lines 1 and 2 before the `while` loop is reached. Either the execution of the `while` loop terminates or it does not.

If the execution of the loop ends then it follows by part (b) of the loop invariant that `i` is an integer variable such that $0 \leq \texttt{i} \leq n-1$ before the step at line 6 is reached and executed. On the other hand, since the loop test at line 3 was just checked and failed, $\texttt{i} \geq \texttt{A.length} - 1 = n - 1$ as well. Thus $\texttt{i} = n - 1$.

It now follows by part (c) of the loop invariant that `sum` is an integer variable with value

$$\sum_{j=0}^{\texttt{i}} \texttt{A}[j] = \sum_{j=0}^{n-1} \texttt{A}[j]$$

immediately before the execution of the step at line 6. The value

$$\sum_{i=0}^{n-1} \texttt{A}[i]$$

is therefore returned as output, when the step at line 6 is executed and the execution of the algorithm ends. The postcondition for the "Sum of Array Elements" problem has therefore been satisfied, as needed to establish condition (a) in the definition of "partial correctness".

On the other hand, if the execution of the loop does not terminate then the execution of the algorithm certainly does not terminate, either — as needed to establish condition (b) in the definition of "partial correctness".

It follows that this algorithm is partially correct.