EECS 3101 M: Design and Analysis of Algorithms

Suprakash Datta
Office: LAS 3043

Course page: `http://www.eecs.yorku.ca/course/3101M`
Also on Moodle

More on Correctness of Algorithms

Let us prove the correctness of the following algorithm for computing the n-th power of a given real number.

$\text{POWER}(y, z)$

// return y^z where $y \in R, z \in \mathbb{N}$
$x = 1$
while $z > 0$
    if $\text{ODD}(z)$
        $x = x * y$
    $z = \lfloor z/2 \rfloor$
    $y = y^2$
return x

Formulating the Loop Invariant

```
POWER(y, z)
x = 1
while z > 0
    if ODD(z)
        x = x * y
    z = ⌊z/2⌋
    y = y^2
return x
```

How does the algorithm proceed?

The loop invariant for this code segment needs to capture the entire state of the program, viz., variables x, y, z. Otherwise proving the invariant may be difficult.

Formulating the Loop Invariant

```
POWER(y, z)
x = 1
while z > 0
    if ODD(z)
        x = x * y
    z = ⌊z/2⌋
    y = y^2
return x
```

Since y, z are changed in the program, let y_0, z_0 denote the initial values of y, z respectively. We want to express the fact that in each iteration the loop stores in x, y_0 raised to the power "the last $i-1$ digits of z_0". The part in quotes can be compactly expressed as $z_0 \mod 2^{i-1}$.

Formulating the Loop Invariant

- Suppose the number of bits in z is n. Suppose also that the initial values of x, y, z are x_0, y_0, z_0 respectively. We see that the while loop goes from $i = 1$ to n. After studying the program the following loop invariant seems reasonable:
  LI: Before iteration i, $z = \lfloor \frac{z_0}{2^{i-1}} \rfloor$, $x = y_0^{z_0 \mod 2^{i-1}}$ and $y = y_0^{2^{i-1}}$.

- We prove Initialization, correctness and termination

Proving Correctness: Initialization

Initialization: Before the first iteration, the invariant yields $z = \lfloor \frac{z_0}{2^0} \rfloor = z_0$, $x = y_0^{z_0 \mod 2^0} = y_0^0 = 1$ and $y = y_0^{2^0} = y_0$. All these values match the code – x is initialized to 1 in line 1 and y, z are unchanged.

Proving Correctness: Maintenance

Assume that the loop invariant holds at the beginning of iteration i. We want to show that it holds at the beginning of iteration $i+1$. So before the current iteration, we have $z = \lfloor \frac{z_0}{2^{i-1}} \rfloor$, $x = y_0^{z_0 \mod 2^{i-1}}$ and $y = y_0^{2^{i-1}}$.
Lines 4 and 5 (potentially) change x. Notice that $z_0 \mod 2^i = 2^i + z_0 \mod 2^{i-1}$ if the i^{th} bit of z_0 is a 1; otherwise $z_0 \mod 2^i = z_0 \mod 2^{i-1}$. Notice also that the i^{th} bit of z_0 is a 1 iff $z = \lfloor \frac{z_0}{2^{i-1}} \rfloor$ is odd. Therefore if the i^{th} bit of z_0 is a 0 x is unchanged. This is what lines 4,5 do.
Otherwise,
$x = x * y = y_0^{z_0 \mod 2^{i-1}} * y_0^{2^{i-1}} = y_0^{2^i + z_0 \mod 2^{i-1}} = y_0^{z_0 \mod 2^i}$,
which is exactly the loop invariant for y at the beginning of the next iteration.

Proving Correctness: Maintenance

Line 6 changes z to $\lfloor z/2 \rfloor = \lfloor \lfloor \frac{z_0}{2^{i-1}} \rfloor /2 \rfloor = \lfloor \frac{z_0}{2^i} \rfloor$, and line 7 changes y to $(y_0^{2^{i-1}})^2 = y_0^{2^i}$. Thus the maintenance proof is complete.

Proving Correctness: Termination and Correctness

The loop terminates with $i = n + 1$.

Plugging this value of i into the invariant we get
$x = y_0^{z_0 \mod 2^{n+1-1}} = y_0^{z_0}$.

This is what the program was meant to do and it is therefore proven correct.

Note that the final values of y, z are not really important, since x is what the program returns.

Correctness of Insertion Sort

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

- What is a good loop invariant?
- It is easy to write a loop invariant if you understand what the algorithm does.

Forming Loop Invariants

- LI for outer loop:

Forming Loop Invariants

- LI for outer loop: LI1: at the start of for loop iteration j, $A[1..j-1]$ consists of elements originally in $A[1..j-1]$ but in sorted order

Forming Loop Invariants

- LI for outer loop: LI1: at the start of for loop iteration j, $A[1\,..\,j-1]$ consists of elements originally in $A[1\,..\,j-1]$ but in sorted order
- The inner while loop moves elements finds the highest position k so that $A[k] \leq key$ and then moves $A[k\,..\,j-1]$ one position right without changing their order.
  Then, in the outer loop, the *key* element is inserted into $A[k]$ so that $A[k-1] \leq A[k] \leq A[k+1]$.

Forming Loop Invariants

- LI for outer loop: LI1: at the start of for loop iteration j, $A[1..j-1]$ consists of elements originally in $A[1..j-1]$ but in sorted order
- The inner while loop moves elements finds the highest position k so that $A[k] \leq key$ and then moves $A[k..j-1]$ one position right without changing their order.
  Then, in the outer loop, the *key* element is inserted into $A[k]$ so that $A[k-1] \leq A[k] \leq A[k+1]$.
  LI2: at the start of for loop iteration i, key contains $A[j]$ and $A[i+2..j]$ consists of elements originally in $A[i+1..j-1]$ but moved one spot to the right and the original $A[i+1] \geq key$

Correctness of LI2: Initialization

Putting $i = j - 1$ in LI2, we get the statement
key contains $A[j]$ and $A[j + 1 \,..\, j]$ consists of elements originally in $A[j \,..\, j - 1]$ but moved one spot to the right and the original $A[j] \geq key$
This is true, because of line 2.

Correctness of LI2: Maintenance

LI2: at the start of for loop iteration i, key contains $A[j]$ and $A[i+2\,..\,j]$ consists of elements originally in $A[i+1\,..\,j-1]$ but moved one spot to the right and the original $A[i+1] \geq key$

We assume that LI2 holds before iteration i, the loop body executes and will show that LI2 holds before iteration $i-1$ (the loop index is decreasing).
Since the loop body executes we know that $i > 0$ and $A[i] > key$. Also, key contains $A[j]$. The loop body moves (copies) $A[i]$ to $A[i+1]$, and decrements i. So LI2 holds before iteration $i-1$.

Correctness of LI2: Termination and Correctness

The loop terminates when $i = 0$ or $A[i] \leq key$

Case 1: $i = 0$: Plugging $i = 0$ into LI2 we get:
at the start of for loop iteration i, key contains $A[j]$ and $A[2\,..\,j]$ consists of elements originally in $A[1\,..\,j-1]$ but moved one spot to the right and the original $A[1] \geq key$
Thus in this case the loop found the correct place $k = 0$.

Case 2: $A[i] \leq key$. Plugging in this value of i, we get $A[i+2\,..\,j]$ consists of elements originally in $A[i+1\,..\,j-1]$ but moved one spot to the right and the original $A[i+1] \geq key$
So in both cases, the loop does what it was meant to do and it is therefore correct.

Proving Correctness of LI1: Initialization

LI1: at the start of for loop iteration j, $A[1 \,..\, j-1]$ consists of elements originally in $A[1 \,..\, j-1]$ but in sorted order
Before the first iteration, $j = 2$, LI1 trivially holds because $A[1 \,..\, 1]$ is a sorted array

Proving Correctness of LI1: Maintenance

Since the inner loop is correct, we know it moves elements finds the highest position k so that $A[k] \leq key$ and then moves $A[k \mathinner{.\,.} j-1]$ one position right without changing their order.

Then, in line 8 in the outer loop, the *key* element is inserted into $A[k]$ so that $A[k-1] \leq A[k] \leq A[k+1]$.

Since LI1 held before the current iteration, $A[1 \mathinner{.\,.} j-1]$ was sorted and $A[j]$ was inserted in the correct place, so $A[1 \mathinner{.\,.} j]$ consists of elements originally in $A[1 \mathinner{.\,.} j]$ but in sorted order.

Proving Correctness of LI1: Termination and Correctness

The loop terminates with $j = A.length + 1$

Plugging this value of j into LI1 we get $A[1 \mathinner{..} A.length]$ consists of elements originally in $A[1 \mathinner{..} A.length]$ but in sorted order

This is what the loop (program) was meant to do and it is therefore proven correct.