Examples for Loop Invariants

September 5, 2007

We can use loop invariants as a proof technique to prove an algorithm involving a loop. In Goodrich and Tamssia's book (page 27), the technique is summarized as follows.

To prove some statement $\mathcal{S}$ about a loop is correct, define in terms of a series of smaller statement $\mathcal{S}_0$, $\mathcal{S}_1$, ..., $\mathcal{S}_k$, where:

- The initial claim, $\mathcal{S}_0$, is true before the loop begins.
- If $\mathcal{S}_{i-1}$ is true before iteration i begins, then one can show that $\mathcal{S}_i$ will be true after iteration i is over or at the beginning of loop $i+1$.
- The final statement $\mathcal{S}_k$ implies the statement S that we wish to justify as being true.

This is essentially an induction proof. The proof is for a loop iterating from 1 to k. It's trivial to expand this argument to other loop bounds. In class, I described $\mathcal{S}_{i-1}$ as a loop invariant, a property that holds at the beginning of each loop iteration i. Our text book (Cormen et al.) names the three steps as *initialization*, *maintenance*, and *termination* (page 17-18).

Example 1.

```
int calSum(int n)
{
  int i, sum;

  sum = 0;
  for (i=1; i <= n; i++)
    sum += i;
  return sum;
}
```

We like to show that this loop returns $\sum_{i=0}^{n} i$. The loop invariant for this loop is that $sum = \sum_{k=0}^{i-1} k$ at the beginning of each loop.

- initial claim/initialization. The claim is trivially true at the beginning of the first loop when i=1. Now *sum* is initialized to 0 before the loop and $\sum_{k=0}^{0} k = 0$.
- induction step/maintenance. Assume that the claim is true at the beginning loop i, we show that it holds at the beginning of loop $i+1$. If at the beginning of loop i, $sum = \sum_{k=0}^{i-1} k$, we have $sum = \sum_{k=0}^{i-1} k + i = \sum_{k=0}^{i} k$ at the end of loop i. Then $sum = \sum_{k=0}^{i} k$ at the beginning of loop $i+1$.
- final claim/termination. The loop terminates when i is incremented to $n+1$, at which point the loop invariant property $sum = \sum_{k=0}^{n} k$ holds.

Example 2.

The Fibonacci sequence is defined as follows.

$$f_n = \begin{cases} n, & n = 0, 1 \\ f_{n-1} + f_{n-2}, & n \geq 2 \end{cases}$$

We design an iterative algorithm to calculates f_n given n.

```
double fibIterativeint n
{
  int i;
  double F_n, F_{n-1}, F_{n-2};

  if (n < 2) return n;

  F_{n-2} = 0;
  F_{n-1} = 1;
  for (i = 2; i≤n; i++) {
     F_n = F_{n-1} + F_{n-2};
     F_{n-2} = F_{n-1};
     F_{n-1} = F_n;
  }

  return F_n;
}
```

We want to prove this algorithm returns f_n. It is trivially true when $n < 2$ assuming the input parameter $n \geq 0$. If $n \geq 2$, we use loop invariant technique to show that the loop calculates f_n. We observe that the loop invariant is that $F_{n-1} = f_{i-1}$ and $F_{n-2} = f_{i-2}$ at the beginning of each loop iteration i. This claim is proved as following.

- initial claim/initialization. The claim is true at the beginning of the first loop iteration when i=2. F_{n-1} is initialized to 1 which equals to f_1 and F_{n-2} is initialized to 0 which equals to f_0

- induction step/maintenance. Assume that the claim is true at the beginning loop i, we show that it holds at the beginning of loop $i+1$. If at the beginning of loop i, $F_{n-1} = f_{i-1}$ and $F_{n-2} = f_{i-2}$, when executing the loop body we get $F_n = F_{n-1} + F_{n-2} = f_{i-1} + f_{i-2} = f_i$, $F_{n-2} = F_{n-1} = f_{i-1}$, and $F_{n-1} = F_n = f_i$. Therefore, at the end of this loop iteration $F_n = f_i, F_{n-1} = f_i$ and $F_{n-2} = f_{i-1}$, the last two of which are the property we want to prove for the beginning of loop $i+1$.

- final claim/termination. Based on the step 1 and step 2, we know that at the beginning of loop n, $F_{n-1} = f_{i-1}$ and $F_{n-2} = f_{i-2}$. Using the same argument in Step 2, we know when the final iteration finishes, $F_n = f_n$.