CIS 313 Intermediate Data Structures

Sample Proof of Correctness by Loop Invariant

We wish to prove the correctness of the following piece of code, which converts an integer n to its representation in binary b:

```
input: integer n>=0
output: integer k, array b of k bits

convert_to_binary(n)   {

-- initialization
int k=0
int t=n
array b = [] of bit

--loop
while t>0 do
     b[k] = (t mod 2)
     k = k+1
     t = t div 2

--end
return b, k
}
```

An invariant α for a general loop `<init> while` γ `do` $\mathcal{L}$ must satisfy the following three properties:

(i) *(initialization)* α is true after the initialization phase `<init>`.

(ii) *(maintenance)* Suppose γ is true so that the loop can be entered. If α is true, then after one execution of the body of the loop $\mathcal{L}$, the invariant α will still be true.

(iii) *(termination)* Eventually $\neg\gamma$ will occur, so the loop will halt. The desired outcome is $\neg\gamma \wedge \alpha$.

For `convert_to_binary`, the loop condition is clearly $\gamma =$ "$t > 0$" and the loop invariant α we will use is

- $t \geq 0$, and
- Let $m = \sum_{i=0}^{k-1} b[i] \cdot 2^i$ be the number represented by b. Then $n = 2^k \cdot t + m$.

Let's work through the three steps of the process of using a loop invariant.

(i) *(initialization)* Initially, $t = n \geq 0$, so part 1 of α holds. Also, $k = 0$ and $b = []$, so $m = 0$. Part 2 of α holds now since $2^k \cdot t + m = 2^0 \cdot n + 0 = n$.

(ii) *(maintenance)* Suppose that both γ and α are true. Then $t > 0$ and $n = 2^k \cdot t + m$ (where m is defined as above). The new values of t, k, m we will call t', k', m', and clearly $k' = k + 1$. Also, $t' = \lfloor t/2 \rfloor$.

We need to show that α holds for these new values t', k', m'. Part 1 of α is easy: $t > 0$ so $t' = \lfloor t/2 \rfloor \geq 0$. For part 2, it remains to show that $n = 2^{k'} \cdot t' + m'$ There are two cases, depending on whether t is even or odd.

(t is even) Here $b[k] = 0$, $m' = m$, and $t' = \frac{t}{2}$. Now

$$2^{k'} \cdot t' + m' = 2^{k+1} \cdot \frac{t}{2} + m = 2^k \cdot t + m = n$$

(the last step follows by hypothesis) and part 2 is true.

(t is odd) In this case $b[k] = 1$, $m' = 2^k + m$, and $t' = \frac{t-1}{2}$. Substituting as above,

$$2^{k'} \cdot t' + m' = 2^{k+1} \cdot \frac{t-1}{2} + (2^k + m) = 2^k \cdot t - 2^k + (2^k + m) = 2^k \cdot t + m = n$$

and again part 2 of α holds.

(iii) *(termination)*

The loop terminates, since $t > \lfloor t/2 \rfloor$.

At termination, both α and $\neg\gamma$ are true. Part 1 of α tells us that $t \geq 0$ while $\neg\gamma$ says that $t \leq 0$. From these we get $t = 0$.

Now let's look at part 2 of α at termination. Remember that $m = \sum_{i=0}^{k-1} b[i] \cdot 2^i$ is the number represented by the bits stored in the array b. Part 2 says that $n = 2^k \cdot t + m$. But we know that $t = 0$, so $n = m$. That is what we wanted to prove, namely that b stores the binary representation of the number n.