Chapter 14 : Fusc.

In which we apply the techniques we learned in Chapter 13.

Suppose we are given the fusc function defined as follows.

fusc : natural → natural

* (0) fusc.0 = 0

* (1) fusc.1 = 1

* (2) fusc.(2*n) = fusc.n , 0 < n

* (3) fusc.(2*n+1) = fusc.n + fusc.(n+1) , 0 < n

We are given a natural number N and asked to write a program to compute fusc.N

Choosing invariants.

We propose as invariants

P0 : α * fusc.n + β * fusc.(n+1) = fusc.N
P1 : $0 \le n \le N$

Establishing the Invariants.

The following assignment establishes the invariants

n, α, β := N, 1, 0

Termination.

We note the following.

P0 ∧ P1 ∧ n=0
= {definitions of P0, P1}
α * fusc.n + β * fusc.(n+1) = fusc.N ∧ $0 \le n \le N$ ∧ n=0
⇒ {leibniz}
α * fusc.0 + β * fusc.1 = fusc.N ∧ true
= {predicate calculus}
α * fusc.0 + β * fusc.1 = fusc.N

= {defn. fusc}

$\alpha * 0 + \beta * 1 = fusc.N$

= {arithmetic}

$\beta = fusc.N$

Calculate the loop body.

P0

= {definition P0}

$\alpha*fusc.n + \beta*fusc.(n+1) = fusc.N$

= {case analysis, even.n i.e. n = 2*p}

$\alpha*fusc.(2*p) + \beta*fusc.(2*p+1) = fusc.N$

= {definition fusc (2) (3)}

$\alpha * fusc.p + \beta*(fusc.p + fusc.(p+1)) = fusc.N$

= {*/+}

$\alpha * fusc.p + \beta*fusc.p + \beta*fusc.(p+1) = fusc.N$

= {gather like terms together}

$(\alpha+\beta)*fusc.p + \beta*fusc.(p+1) = fusc.N$

= {WP}

$(n, \alpha, \beta := n \text{ div } 2, \alpha + \beta, \beta).P0$

Giving us the program fragment

if even.n → n, α, β := n div 2, $\alpha + \beta$, β

Let us analyse the other case

P0

= {definition P0}

$\alpha*fusc.n + \beta*fusc.(n+1) = fusc.N$

= {case analysis, odd.n i.e. n = 2*p+1}

$\alpha*fusc.(2*p+1) + \beta*fusc.(2*p+2) = fusc.N$

= {definition fusc (2) (3)}

$\alpha*(fusc.p + fusc.(p+1)) + \beta*fusc.(p+1) = fusc.N$

= {*/+}

$\alpha*fusc.p + \alpha*fusc.(p+1) + \beta*fusc.(p+1) = fusc.N$

= {gather like terms together}

$\alpha*fusc.p + (\alpha+\beta)*fusc.(p+1) = fusc.N$

= {WP}

$(n, \alpha, \beta := (n-1) \text{ div } 2, \alpha , \alpha + \beta).P0$

Which gives us the program fragment

```
if odd.n → n, α, β := (n-1) div 2, α , α + β
```

Finished program.

And so the entire program is as follows.

```
n, α, β := N, 1, 0 {P0 ∧ P1}
;do n ≠ 0  →    {P0 ∧ P1 ∧ n ≠ 0}

                if even.n → n, α, β := n div 2, α + β, β
                [] odd.n → n, α, β := (n-1) div 2, α , α + β
                fi

                {P0 ∧ P1}
od
{β = fusc.N}
```

We note that this solution has temporal complexity O(log.N)