Derek McCarthy B00007439
Week 9 CA Derivation of Algorithms

Question

Given an integer array f[0..N), where {N > 0}. Write a specification and hence derive a solution that will find the largest value in f and the frequency of its occurrence.

Step 1 – Program Specification

|[

con N: int; {N > 0}
f: array [0..N) of int;
var
freq: int;
largest: int;

S

{freq = #(largest = ∀i : 0 ≤ i < N : f.i ≥ largest)}

]|

Step 2 - Write down invariants P0 and P1

Replacing N with n.

P0 : {freq = #(largest = ∀i : 0 ≤ i < n : f.i ≥ largest)}

P1 : 0 ≤ n ≤ N

Step 3 – Write out an outlier solution

S0;

{P, Bound function N-n}

do n ≠ N →

{P ⋀ n ≠ N}

S1;

{P}

od

{P ⋀ n = N}

{freq = #(largest = ∀i : 0 ≤ i < n : f.i ≥ largest}

Step 4 – Derive S0, variable initialisation.

Suggest n, largest := 1, 0;

$(freq = \#(largest = \forall i : 0 \leq i < n : f.i \geq largest \wedge 0 \leq n \leq N)(n, largest := 1, 0)$

$\equiv$ [Substitution]

$freq = \#(0 = \forall i : 0 \leq i < 1 : f.i \geq 0 \wedge 0 \leq 1 \leq N)$

$\equiv$ [Only value that can satisfy range for i is 0, therefore i = 0]

$freq = \#(0 = f.0 \geq 0 \wedge 0 \leq 1 \leq N)$

$\equiv$ [Exclude the middle, arithmetic]

$true = \#(0 = f.0 \geq 0 \wedge 0 \leq 1 \leq N)$

$\equiv$ [Arithmetic on f.0]

$true = 0 = true \wedge 0 \leq 1 \leq N$

$\equiv$ [Exclude the middle, Arithmetic]

$true = true = true \wedge 0 \leq 1 \leq N$

$\equiv$ [Constants]

$1 \leq N$

$\Leftarrow \{N > 0\}$

<u>true</u>

Therefore S0 ; given by,

freq := 0;
largest :=0;
n := 1;

Step 5 – Derive S1; loop body

Suggest n := n + 1

$(0 \leq n \leq N)(n := n + 1)$

$\equiv$ [Substitution]

$freq = \#(largest = \forall i : 0 \leq i < n : f.i \geq largest)$

$\equiv$ [Split off i = n]

$freq = \#(largest = \forall i : 0 \leq i < n : f.i \geq largest) + \#(f.n \geq largest)$

$\equiv$ [$\Leftarrow$ P]

$freq = freq + \#(f.n \geq largest)$

≣ [Case Analysis]

freq := freq + 1, if f.n ≥ largest

freq := freq + 0, if f.n < largest

≣ [Case Analysis]

[if...fi]

Logic behind case analysis

≣ [if..fi]
if f.n ≥ largest →
freq := freq + 1
largest := f.n
[] f.n < largest →
Skip;
fi

Therefore S1; becomes
if f.n ≥ largest →
freq := freq + 1
largest = f.n
[] f.n < largest →
Skip;
fi
n : n + 1

Step 6 – Prove termination

Initialisation
(N – n ≥ 0) (n := 1)

≣ [Substitute]

N – 1 ≥ 0

≣ [Arithmetic]

N ≥ 1

⇐ {Given N > 0}

Loop Body
(N - n) (n := n + 1)

≣ [Substitute]

N – (0 + 1)

≣ [Arithmetic]

N – n - 1

<

N - n

Therefore Decreasing

Step 6 - Complete Solution

```
|[ Con N : int; {N > 0}
        f: array [0..N) of int;
        var
        freq, n, largest : int;
        freq, n, largest := 0, 1, 0;
        do n ≠ N →
                if f.n ≥ largest →
                        freq := freq + 1;
                        largest := f.n;
                [] f.n < largest →
                        Skip;
                fi
                n := n + 1;
        od
        {freq = #(largest = ∀i : 0 ≤ i < N : f.i ≥ largest)}
]|
```