Chapter 37 : The starting pit problem.

In which we tackle a difficult problem.

There are N pits located along a circular race track. They are numbered 1..N. At pit i there are p.i litres of fuel available. To race from pit i to its clockwise neighbour we require q.i litres of fuel. We are asked to find a pit from which it is possible to race a complete lap starting with an empty fuel tank.

To guarantee the existence of such a pit we are given

* (0) $\langle + i : 1 \le i \le N : p.i \rangle = \langle + i : 1 \le i \le N : q.i \rangle$

We introduce some notation.

* (1) D.i.j $\quad = \quad \langle + k : i \le k < j : p.k - q.k \rangle$

This is the difference between the number of litres available and the number of litres required when racing from pit i to pit j. [1]

Here are a few properties of D

- (2) D.i.k $\quad = \quad$ D.i.j + D.j k $\quad$,i, j, k ε {1..N}

- (3) D.i.i $\quad = \quad$ 0

- (4) D.i.j + D.j.i $\quad = \quad$ 0

Towards using the symmetric linear search in our solution, we now define F

* (5) F.x $\quad \equiv \quad \langle \forall\, i :: 0 \le D.x.i \rangle$

We can now specify our program

Pre : $\langle \exists\, k : 1 \le k \le N : F.k \rangle$

Post : F.x

We now calculate our guards

F.a ⇒ F.b

= {definition of F}

$\langle \forall\, i :: 0 \le D.a.i \rangle \Rightarrow \langle \forall\, i :: 0 \le D.b.i \rangle$

---

[1] As the race track is circular we can have D.2.1 which is of course D.2.N + D.N.1. We will not complicate our notation by introducing modular arithmetic.

$=$ {(2)}

$\langle \forall\ i :: 0 \le D.a.b + D.b.i \rangle \Rightarrow \langle \forall\ i :: 0 \le D.b.i \rangle$

$\Leftarrow$ {arithmetic}

$D.a.b \le 0$

Symmetrically, $(F.b \Rightarrow F.a) \Leftarrow D.b.a \le 0$

As $D.b.a = -D.a.b$ we can rewrite this as $(F.b \Rightarrow F.a) \Leftarrow 0 \le D.a.b$

We now arrive at our program

```
a, b := 1, N
;do a ≠ b →    {M ≤ a < b ≤ N}

               if D.a.b ≤ 0 → a := a + 1
               [] 0 ≤ D.a.b → b := b – 1
               fi

od
; x := a
```

Evaluating the guards could be expensive, so we strengthen our invariant as follows

P2 : $d = D.a.b$

We now have to determine the appropriate assignments to d which will establish and maintain P2.

Clearly we need $d := D.1.N$. But recall that $D.1.N = -D.N.1$

So, the assignment

$a,b,d := 1, N, q.N-p.N$

establishes the invariants.

Let us consider one of the branches

$(a, d := a+1, E).P2$

$=$ {text sub.}

$E = D.(a+1).b$

$=$ {arithmetic}

$E = D.a.b + q.a – p.a$

Thus we have

```
if d ≤ 0 → a, d := a + 1, d + q.a – p.a
```

We leave it to the reader to complete the remainder of the work. This should lead to the the final version of our program as follows

```
a, b, d  := 1, N, q.N – p.N
;do a ≠ b →     {M ≤ a < b ≤ N}

                if d ≤ 0 → a, d := a + 1, d + q.a – p.a
                [] 0 ≤ d → b, d := b – 1, d + q.(b-1) – p.(b-1)
                fi

od
; x := a
```