Concurrent Zipping (for our files)

dedicated to Johan L. Lukkien

When Johan Lukkien saw our problem of "Concurrent Vector Writing", he concocted a much more intriguing programming exercise which he solved in his own way. The purpose of this note is to show him a purely formal derivation, rendered in the formalism that we (WF+AvG) happen to be familiar with. (But we will not explain this formalism here.)

* * *

The computation proper that we start with is the two-component multiprogram given by

<table>
<tr><td colspan="2">Pre: $i=0 \wedge j=0$ $(\wedge\ 0 \le N)$</td></tr>
<tr><td>A: <u>do</u> $i \ne N$
→ $x.i := 0$
; $i := i+1$
<u>od</u></td><td>B: <u>do</u> $j \ne N$
→ $x.j := 0$
; $j := j+1$
<u>od</u></td></tr>
<tr><td colspan="2">Post: ? $\langle \forall k : (0 \le) k < N : x.k = k \ \underline{\text{mod}}\ 2 \rangle$</td></tr>
</table>

Approximation 0 (= Specification)

We are asked to synchronize the two components such that postcondition Post will be established.

* * *

To that end we choose

PA: $\langle \forall k : \text{even}.k \wedge k < i : x.k = 0 \rangle$, and

PB: $\langle \forall k : \text{odd}.k \wedge k < j : x.k = 1 \rangle$

to become system invariants. Then the result follows, if both components terminate. By symmetry we will focus on just PA, in what follows.

Condition PA can only be violated by $i := i+1$ in A and by $x.j := 1$ in B.

<u>Re "$i := i+1$"</u> in A

$(i := i+1).PA$
$\equiv$ {substitution}
$\langle \forall k : \text{even}.k \wedge k < i+1 : x.k = 0 \rangle$
$\equiv$ { predicate calculus }
$PA \wedge (\text{odd}.i \vee x.i = 0)$

Therefore we require

$\{?\ \text{odd}.i \vee x.i = 0\}\ i := i+1$

<u>Re "$x.j := 1$"</u> in B

$(x.j := 1).PA$
$\Leftarrow$ { j outside range of k in PA}
$PA \wedge (\text{odd}.j \vee i \leq j)$.

Therefore we require

$\{?\ \text{odd}.j \vee i \leq j\}\ x.j := 1$

<u>End</u> Re's

Thus we arrive at our next approximation, viz.

```
Pre:   i = 0  ∧  j = 0
-------------------------------------------------
A:     do i ≠ N →
              x.i := 0
            ; {? odd.i  ∨  x.i = 0,  Note 0 }
              i := i + 1
       od
-------------------------------------------------
B:     do j ≠ N →
            {? odd.j  ∨  i ≤ j,  Note 1 }
              x.j := 1
            ; j := j + 1
       od
-------------------------------------------------
Inv:   PA :  ⟨∀k: even.k ∧ k < i : x.k = 0⟩
```

Approximation 1

* * *

Next we tackle the two remaining (queried) assertions.

Note 0 "$odd.i \vee x.i=0$" in A

L: correct from the preceding $x.i:=0$

G: $(x.j:=1).(odd.i \vee x.i=0)$

$\equiv$ { substitution }

$odd.i \vee (i \neq j \wedge x.i=0)$

$\Leftarrow$ { target assertion $odd.i \vee x.i=0$ itself }

$odd.i \vee i \neq j$

$\Leftarrow$ { calculus }

$odd.i \vee i<j$

We supply our target assertion with co-assertion

$$\text{odd}.i \vee i<j \;.$$

Note 1 "$\text{odd}.j \vee i \le j$" in B

We decide to establish this assertion by requiring it to be a system invariant.[F]

End of Notes.

Thus, we have arrived at

Pre:	$i=0 \wedge j=0$
A:	<u>do</u> $i \ne N \rightarrow$ $\quad x.i := 0$ $\quad ; \{\text{odd}.i \vee x.i=0\}\{?\ \text{odd}.i \vee i<j$, Note 0$\}$ $\quad i := i+1$ <u>od</u>
B:	<u>do</u> $j \ne N \rightarrow$ $\quad x.j := 1$ $\quad ; j := j+1$ <u>od</u>
Inv:	PA: $\langle \forall k : \text{even}.k \wedge k<i : x.k=0 \rangle$ Z: $?\ \text{odd}.j \vee i \le j$, Note 1

Approximation 2

* * *

F This is no rabbit. The decision is imposed on us because disjunct $i \le j$ is an invariant of B

We just proceed.

Note 0 "odd.i ∨ $i<j$" in A

G: correct, by Widening

L: by prefixing statement $x, i := 0$ in A with guarded skip

if odd.i ∨ $i<j$ → skip fi

Note 1 Invariant "Z: odd.j ∨ $i \le j$"

Initially: ok

Re "$i:=i+1$" in A

$$
\begin{array}{ll}
 & (i:=i+1).\ Z \\
\equiv & \quad \{\text{substitution}\} \\
 & \text{odd}.j \ \vee\ i<j \\
\Leftarrow & \quad \{\text{a bit of fancy calculus}\} \\
 & (\text{odd}.i \vee i<j) \wedge (\text{odd}.j \vee i \le j) \\
\equiv & \quad \{\text{definition } Z\} \\
 & (\text{odd}.i \vee i<j) \wedge Z
\end{array}
$$

The first conjunct is pre-assertion of $i:=i+1$. So, we are done

Re "$j:=j+1$" in B

$$
\begin{array}{ll}
 & (j:=j+1).\ Z \\
\equiv & \quad \{\text{substitution}\} \\
 & \text{even}.j \ \vee\ i \le j+1 \\
\Leftarrow & \quad \{\text{predicate calculus}\}^{F} \\
 & i \le j+1
\end{array}
$$

and we decide $i \le j+1$ to be a system invariant.

F This move is a little rabbitish.

End of Notes.

We summarize the above design, and add a number of "outdented" assertions in A for demonstrating the invariance of $i \le j+1$

Pre: $i = 0 \wedge j = 0$

A: <u>do</u> $i \ne N \rightarrow$

$\{\text{even}.i \vee i \le j\}$

  <u>if</u> $\text{odd}.i \vee i < j \rightarrow \text{skip}$ <u>fi</u>

$\{i \le j\}$

  ; $x.i := 0$

  ; $\{\text{odd}.i \vee x.i = 0\}\{\text{odd}.i \vee i < j\}$

$\{i \le j\}$

  $i := i+1$

$\{\text{even}.i \vee i \le j$, locally following from pre-assertion $\text{odd}.i \vee i < j\}$

<u>od</u>

B: <u>do</u> $j \ne N \rightarrow$

  $x.j := 1$

  ; $j := j+1$

<u>od</u>

Inv: PA : $\langle \forall k : \text{even}.k \wedge k < i : x.k = 0 \rangle$,

Z : $\text{odd}.j \vee i \le j$,

$i \le j+1$, see below

Approximation 3

The outdented assertions are all globally correct — Widening —. Their local correctness is readily established. The invariance of $i \leq j+1$ follows from pre-assertion $i \leq j$ of $i := i+1$

* * *

Finally, we have to handle PB, the other original target invariant. By symmetry, this is done at once by replacing in A the quadruple $(i, j, \text{odd}, 0)$ with $(j, i, \text{even}, 1)$. The final, unannotated version reads

Pre: $i = 0 \wedge j = 0$

A: <u>do</u> $i \neq N \rightarrow$
    <u>if</u> odd.$i \vee i < j \rightarrow$ skip <u>fi</u>
  ; $x.i := 0$
  ; $i := i+1$
<u>od</u>

B: <u>do</u> $j \neq N \rightarrow$
    <u>if</u> even.$j \vee j < i \rightarrow$ skip <u>fi</u>
  ; $x.j := 1$
  ; $j := j+1$
<u>od</u>

Post: $\langle \forall k :: x.k = k \ \underline{\text{mod}}\ 2 \rangle$

Solution

* * *

Do both components terminate? They do, because

- there is no danger for the two components to get stuck simultaneously in their guarded skips:

$$\begin{array}{ll} & odd.i \vee i<j \;\vee\; even.j \vee j<i \\ \equiv & \quad \{\text{algebra}\} \\ & i \neq j \;\vee\; odd.i \;\vee\; even.j \\ \equiv & \quad \{\text{modern algebra}\} \\ & true \;. \end{array}$$

- thus at least one component terminates its repetition. Let this be A. Then the state satisfies $i = N$ so that the guard in B's guarded skip equivales true.

* * *

We think that

- Johan's problem is a nice, but not a trivial one
- With our assertion-driven method for deriving multiprograms, we cannot do much shorter, crisper, or better than what has been recorded in this note.

W.H.J. Feijen
A.J.M. van Gasteren

15 October 1997