A bagatelle (for the files)

Given a boolean sequence $f.(i: 0 \leq i \wedge i < 2 \cdot N)$ we are requested to design a (sequential) program that, by swapping elements of $f.$, establishes relation R given by

$$R: \quad R0 \vee R1 \ ,$$

where $R0$ and $R1$ are given by

$$R0: \quad (\underline{A}i: 0 \leq i \wedge i < N: f.(2 \cdot i))$$

$$R1: \quad (\underline{A}i: 0 \leq i \wedge i < N: \neg f.(2 \cdot i + 1)) \ .$$

*

We have to design a program that has to establish a disjunction. Programs being what they are, for instance constructive existence proofs, it will be nearly unavoidable to establish R without revealing which of the disjuncts is established.

A patented way to conclude $R0$ is by the conjunction of $P0$,

$$P0: \quad (\underline{A}i: 0 \leq i \wedge i < p: f.(2 \cdot i))$$

and $p = N$. Likewise, $R1$ can be concluded from

$$P1: \quad (\underline{A}i: 0 \leq i \wedge i < q: \neg f.(2 \cdot i + 1))$$

and $q = N$. Therefore we propose that our program establishes $R0 \vee R1$ by establishing

$p = N \vee q = N$ under invariance of $P0 \wedge P1$.
Thus we obtain

$\{ 0 \leq N \}$
  $p, q := 0, 0$
$\{$invariant $P0 \wedge P1 \wedge 0 \leq p \wedge p \leq N \wedge 0 \leq q \wedge q \leq N \}$
  ; do $p \neq N \wedge q \neq N$
    → increase $p+q$ maintaining
      the invariant
  od
$\{ P0 \wedge P1 \wedge (p = N \vee q = N)$, hence $R \}$.

The simplest way of increasing $p+q$ is to increase p or to increase q. In view of the guard $p \neq N$ and the required invariance of $p \leq N$, the simplest increase of p is $p := p+1$. Therefore we investigate

  wp. $p := p+1$. $P0$
= { definition of wp, $p := p+1$ and of $P0$ }
  $(\underline{A} i : 0 \leq i \wedge i < p+1 : f.(2 \cdot i))$
= { predicate calculus, using $0 \leq p$ }
  $(\underline{A} i : 0 \leq i \wedge i < p : f.(2 \cdot i)) \wedge f.(2 \cdot p)$
= { definition of $P0$ }
  $P0 \wedge f.(2 \cdot p)$
= { using $P0$ }
  $f.(2 \cdot p)$.

Because $p := p+1$ vacuously maintains $P1$, we have derived

(0) $\{ P0 \wedge P1 \wedge f.(2 \cdot p) \}$ $p := p+1$ $\{ P0 \wedge P1 \}$.

Thus we obtain for the repeatable statement

```
{P0 ∧ P1}
   if f.(2·p) → p := p+1   {P0 ∧ P1}
   ▯ ¬f.(2·q+1) → q := q+1  {P0 ∧ P1. by the same
                              token}
   ▯ ¬f.(2·p) ∧ f.(2·q+1)
        → f: swap.(2·p, 2·q+1)
          {f.(2·p) ∧ ¬f.(2·q+1)}
          {P0 ∧ P1 has not been affected
           by the swap}
          p, q := p+1, q+1
          {P0 ∧ P1}
   fi
{P0 ∧ P1}.
```

We still can simplify the code a little bit by applying the coordinate transformation

$$m = 2 \cdot p \quad \wedge \quad n = 2 \cdot q + 1 .$$

We obtain

```
m, n := 0, 1
; do m ≠ 2·N ∧ n ≠ 2·N+1
  → if f.m → m := m+2
    ▯ ¬f.n → n := n+2
    ▯ ¬f.m ∧ f.n → f: swap.(m,n)
                   ; m, n := m+2, n+2
    fi
  od .
```

*

The program being as nice as it is, I feel strangely annoyed by the above rendering

of the design process on paper. The annoyance is caused by the annotated program texts mainly. They are the result of laziness -- and sloppiness in its wake -- inflicted on me by the "linearity" of the paper. As soon as the guard f.(2·p) has been calculated for the benefit of the invariance of P0 under p:=p+1 , a nice blackboard performance would presumably have displayed a nascent program text reading

$$\{P0\}$$
$$\underline{\text{if}}\ \ f.(2\cdot p) \ \rightarrow\ \ p := p+1 \quad \{P0\}\ .$$

A little bit later the blackboard would have displayed

$$\{P0\}\,\{P1\}$$
$$\underline{\text{if}}\ \ f.(2\cdot p) \ \rightarrow\ \ p := p+1 \quad \{P0\}\{P1\}\ .$$

In this respect formula (0) is already too entangled, doing insufficient justice to a separation of -- be it trivial -- concerns.

Next, the text, as I wrote it, makes an enormous shortcut by displaying the complete program text at once; and there the sloppiness creeps in: vide "by the same token" and the worse "has not been affected by the swap", thus delegating proof obligations to the reader.

One (well-known) moral is that in writing we have to be permanently aware of the fact that the convenience of the reader counts more than the convenience of the writer, and that therefore

laziness, sloppiness, and incompleteness on the writer's part are always out of place. Another moral is that it would do no harm did we have a much more explicit formulation of the limitations and opportunities presented by the use of paper in our rendering of mathematical designs. (A third, and completely different, moral might be that the way we derive program texts and the way we write them down are notationally too far apart to admit for real smooth presentations of program development, be it on paper or on blackboard.)

W.H.J. Feijen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 - 1188
U.S.A.

Austin, 13 February 1988