5 May 1977

In gratitude dedicated to C.A.R.Hoare, D.E.Knuth, and J.F.Traub.

On making solutions more and more fine-grained.

This note deals with a problem that I owe to C.S.Scholten. Today seems an appropriate day to start writing this note, for yesterday evening I completed EWD595' (the second version of EWD595 which, itself, is the n-th version of our joint article on the on-the-fly garbage collection): Scholten's problem was already with us for a few weeks, before we realized that it had, in a way, the same flavour as the collector's problem of detecting that the marking had been completed. Perhaps we shall see one day, that all these solutions, which at present seem disconnected pieces of logical ingenuity --not to say: intricacy-- are all members of the same family.

In the on-the-fly garbage collection the cooperation of mutator and collector ensured during marking that a stable state --all reachable nodes black and all white nodes garbage-- would be reached in a finite number of steps of the collector's marking cycle: the problem was the design of the detection mechanism for the collector that, indeed, the stable state had been reached. Scholten's problem poses such a detection problem for N machines.

In the following y denotes a vector of N components $y[i]$ for $0 \leq i < N$. With the function f we shall denote a vector-valued function of a vector argument, and the algorithms we shall study solve the equation

$$y = f(y) \qquad (1)$$

or, introducing $f0$, $f1$, $f2$, ... for the components of f

$$y[i] = fi(y) \qquad \text{for} \quad 0 \leq i < N \quad . \qquad (2)$$

It is assumed that the initial value of y and the function f are such that the repeated assignments of the form

$$< y[i] := fi(y) > \qquad (3)$$

will lead in a finite number of steps to y being a solution of (1). In (3) we have used Lamport's notation of the angle brackets: they enclose "atomic actions" which can be implemented by ensuring mutual exclusion in time (when they are considered "to take time"). The sequence of i-values for which the assignments are carried out must be one of some sort of "fair random order" in which, for instance, a finite upper bound is given for the maximum number of consecutive assignments --i.e.: i-values-- in which a given j $(0 \leq j < N)$

does not occur: in other words, we assume the absence of individual starvation guaranteed.

Because equation (1) is assumed to have at least one solution, such an initial value of y always exists: start with y equal to a solution! This is, of course, not an interesting case; Scholten has formulated more general conditions (on the domain of the elements of y , on the functions f and on the initial value for y) under which convergence in a finite number of steps, and towards a solution which is uniquely determined by the initial value of y , can be guaranteed. These conditions do not interest us here: we shall study the more general situation in which in a finite number of steps a not-necessarily unique solution of (1) will be reached. (In passing we note that also the marking in the garbage collection had that characteristic of nondeterminacy.)

Note 1. The mechanisms we shall design will even "operate" when no solution of (1) is reached within a finite number of steps: then they will fail to terminate. In this sense our programs can be considered as a multi-dimensional generalization of the Linear Search. (End of note 1.)

We consider solutions consisting of N repetitive processes of the form:

```
prog.i:     do ... → < y[i]:= fi(y) > od                                      PRO
```

but the problem is, of course, what to fill in for the dots. The roughest sketch would be

```
prog.i:     do < E j: 0 ≤ j < N: y[j] ≠ fj(y) > → < y[i]:= fi(y) > od      PR1
```

but this version is rejected for two reasons: firstly, the guard is an unacceptably large grain of action, secondly and more important, we want the construction of prog.i to be independent of fj for $j \ne i$. We can remove the second objection and reduce the first one by introducing a global array e with the boolean elements e[i] for $0 \le i < N$, and maintaining

$$(\underline{A}\ i\colon 0 \le i < N\colon e[i] \Rightarrow (y[i] = fi(y))) \quad . \tag{4}$$

Because (4) is trivially satisfied by all e[i] false, we assume that initialization. With the convention that j ranges over $0 \le j < N$, we can now write (with some more notational liberties that will be explained later)

```
prog.i:     do < E j: non e[j] > →                                    PR2
                 < if y[i] = fi(y) → e[i]:= true >
                    ▯ y[i] ≠ fi(y) → y[i]:= fi(y);
                                    (A j: e[j]:= false) >
                   fi
            od
```

Note 2. I have used the abbreviation (A j: e[j]:= false) for the program that performs the assignments e[0]:= false through e[N-1]:= false in some order. Because here it is part of an atomic action, the undefinedness of the order is still irrelevant. (End of note 2.)

Note 3. In PR2 , the whole alternative construct is effectively a single atomic action. In view of later needs, however, I have given each alternative its own closing angle bracket. (End of note 3.)

Note 4. In the first alternative of PR2 , the superfluous assignment to y[i] has been suppressed. (End of note 4.)

Note 5. In a more abstract version we could have introduced a set E of those processes j for which y[j] = fj(y) is guaranteed to hold. In that case (A j: e[j]:= false) would have been coded as E:= ∅ . Honesty forces me to mention that during more abstract explorations I have, indeed, used such a notation, and to admit that the reason that I don't do so now could very well be, that the symbols of set theory are not on my typewriter. The boolean array can be regarded as the characteristic function for E; the problem, of course, is that we can also regard the value of E as a coding for the value of e . (End of note 5.)

It is clear that both alternatives in PR2 leave (4) invariant. It is also clear that y = f(y) is a stable state as far as y is concerned. Termination of one of the processes implies (A j: e[j]) , from which, together with (4), y = f(y) can be deduced, i.e. that the stable state has been reached, and that all other programs will terminate as well.

Note 6. If we really want to spell this out, we would have to *show* the invariance of, say,

$$(\underline{A}\ j: e[j])\ \underline{and}\ y = f(y) \quad .$$

As we have more difficult problems ahead of us, I shall not waste my time on that demonstration: it is really trivial. (End of note 6.)

* * *

One of the ways in which we could try to chop up the large grain of action in PR2 would be to separate inspection of y , computation of fi , and modification of y[i] . With a local vector vi and a local "scalar", qi we could try:

```
prog.i:     do < E j: non e[j] > →                                      PR3
                 < vi:= y > {y[i] = vi[i]};
                 qi:= fi(vi) {qi = fi(vi)};
                 if vi[i] = qi → < e[i]:= true >
                  ▯ vi[i] ≠ qi → < y[i]:= qi; (A j: e[j]:= false) >
                 fi
            od
```

Note 7. We have allowed ourselves vi:= y as an abbreviation for (A j: vi[j]:= y[j]) . Upon its completion the relation "y[i] = vi[i]" can be regarded as a local assertion of prog.i , in spite of the fact that it contains a reference to the global y[i] : we can do so because for $j \neq i$, prog.j only inspects, but never modifies the value of y[i] . (End of note 7.)

The proof of the invariance of (4) , however, fails for the first alternative in the following manner. The weakest pre-condition for < e[i]:= true > to establish (4) is

$$(4)\ \underline{and}\ y[i] = fi(y) \quad ,$$

but we can only guarantee --see the assertions between braces--

$$(4)\ \underline{and}\ y[i] = vi[i] = qi = fi(vi) \quad ,$$

and in order to conclude the former from the latter we need the further assumption y = vi . Program PR3 is, indeed, wrong, but the failure of its correctness proof indicates how to repair it.

Because the non-destruction of (4) by < e[i]:= true > depends on the truth of y = vi , we can repair program PR3 by replacing < e[i]:= true > by

$$< e[i]{:=}\ (y = vi) >$$

which is a shorthand notation for

$$< e[i]{:=}\ (\underline{A}\ j{:}\ y[j] = vi[j]) > \quad .$$

Because --specially for large N -- this is again a bulky atomic action, we can introduce a global array d with boolean elements $d[i]$ for $0 \le i < N$, such that

$$(\underline{A}\ i{:}\ 0 \le i < N{:}\ d[i] \Rightarrow (y = vi)) \quad . \tag{5}$$

If we can keep (5) invariantly true, replacing $< e[i]{:=}\ \text{true} >$ in PR3 by $< e[i]{:=}\ d[i] >$ ensures that $e[i]$ will not be set to true erroneously, i.e. so as to destroy the truth of (4) . Assuming all the $d[i]$ initialized to false, keeping (5) invariant leads to the following program, that is now derived from PR3 in a straightforward manner:

```
prog.i:     do < E j: non e[j] > →                                          PR4
                  < d[i]:= true; vi:= y > ;
                  qi:= fi(vi);
                  if vi[i] = qi → < e[i]:= d[i] >
                   ▯ vi[i] ≠ qi → < y[i]:= qi;
                                     (A j: d[j]:= false);
                                     (A j: e[j]:= false) >
                  fi
            od
```

* * *

The transition from PR2 to PR4 was motivated by something like the assumption that the fi-computations were time-consuming. Another way of chopping up atomic actions in PR2 would be to separate the modification of $y[i]$ from the false-setting of the $e[j]$'s. In the following program, derived from PR2 , we have introduced a global ghost-variable ef for reasons that will become clear in a moment; ef is assumed to be initialized at false.

```
prog.i:     do < E j: non e[j] > →                                          PR5
                 < if y[i] = fi(y) → e[i]:= true >
                    ▯ y[i] ≠ fi(y) → y[i]:= fi(y); ef:= true > ;
                                      < (A j: e[j]:= false); ef:= false >
                  fi
            od
```

The reason for introducing the ghost-variable ef becomes clear as soon as we realize that $y[i] := fi(y)$ without setting all the $e[j]$'s to false, might cause a violation of (4) as a result of the modification of y . The introduction of ef enables us to express the temporary violation of (4) by replacing it by

$$(\underline{A}\ i\colon 0 \le i < N\colon e[i] \Rightarrow (y[i] = fi(y)))\ \underline{or}\ ef \qquad (6)$$

Note 8. The name "ef" is for me a mnemonic for "e-implication false". (End of note 8.)

Thanks to the introduction of ef , (6) is now clearly an invariant; it is, however, all by itself too weak to conclude that upon termination $y = f(y)$ holds. As it stands we can only conclude upon termination

$$y = f(y)\ \underline{or}\ ef \quad ,$$

a conclusion that suffices, if we can also show the invariance of

$$ef \Rightarrow (\underline{E}\ j\colon \underline{non}\ e[j]) \quad , \qquad (7)$$

for then ef is guaranteed to be false upon termination. It is indeed possible to show that (7) is inveriant as well, and that, therefore, program PR5 is correct.

Without the introduction of more elaborate ghost-variables, we need for the demonstration of the invariance of (7) a somewhat different argument. Consider an atomic action that causes for ef a transition from false to true; let this be performed by prog.k . Then, prior to that atomic action we can assert

$$(6)\ \underline{and}\ \underline{non}\ ef\ \underline{and}\ y[k] \ne fk(y)$$

from which $\underline{non}\ e[k]$ can be concluded. Because prog.k is the only one that can reset $e[k]$ to true and cannot cause this resetting to take place before resetting ef to false, $e[k]$ must remain false --and, hence, $(\underline{E}\ j\colon \underline{non}\ e[j])$ must remain true-- as long as ef remains true.

The operational argument in the preceding paragraph is highly unattractive; it does, however, show the way out. Introducing a global variable k $(0 \le k \le N)$ we can represent $\underline{non}$ ef by $k = N$, and ef by $0 \le k < N$. (In particular: when $k < N$, it has been prog.k that lastly caused ef to become true, i.e. that lastly caused k to become different from N .)

```
prog.i:      do < E j: non e[j] > →                                    PR5'
                  < if y[i] = fi(y) → e[i]:= true >
                   [] y[i] ≠ fi(y) → y[i]:= fi(y);
                                     if k < N → skip
                                      [] k = N → k:= i
                                     fi > ;
                                     < (A j: e[j]:= false); k:= N >
                  fi
             od
```

The program has been called PR5' because it only differs from PR5 by the ghost-variable . The ghost-variable k is assumed to have been initialized = N . It is then easy to prove the invariance of

$$k < N \Rightarrow \underline{\text{non}}\ e[k] \qquad (7')$$

(or, if we don't like undefined righthand sides of implications, $k = N\ \underline{\text{cor}}\ \underline{\text{non}}\ e[k]$). To complete the treatment, relation (6) must be rewritten as

$$(\underline{A}\ i: 0 \le i < N: e[i] \Rightarrow (y[i] = fi(y)))\ \underline{\text{or}}\ k < N \quad . \qquad (6')$$

* * *

The above three stars stand for as many days of vain struggle, when I tried to merge the two achievements embodied in PR4 and PR5' . Eventually I had some success when I started from the rejected correction of PR3 . In the text below, the e[i]'s have been renamed for reasons that will become clear later; initially, all the g[i]'s are false.

```
prog.i:    do < E j: non g[j] > →                                      PR6
                < vi:= y > {vi[i] = y[i]};
                qi:= fi(vi) {qi = fi(vi)};
                if vi[i] = qi → < g[i]:= (y = vi) >
                 [] vi[i] ≠ qi → < y[i]:= qi; (A j: g[j]:= false) >
                fi
           od
```

I won't repeat its correctness proof, but proceed immediately to chop up its last atomic action as in PR5'.

Initially, $k = N$; for the reformulation of (7') we can assume $g[N]$ to be constantly false.

```
prog.i:   do < E j: non g[j] > → {k ≠ i}                                PR7
L0:            < vi:= y > {vi[i] = y[i]};
               qi:= fi(vi) {qi = fi(vi)};
L1:            if vi[i] = qi → < g[i]:= (y = vi) >
L2:             ▯ vi[i] ≠ qi → < y[i]:= qi;
                                 if k < N → skip
                                  ▯ k = N → k:= i
                                 fi > ;
L3:                             < (A j: g[j]:= false); k:= N >
               fi {k ≠ i}
          od
```

In the following correctness proof the atomic actions are referred to by the label on the line of their opening angle bracket.

We first observe that $\{k \neq i\}$ is a local assertion for prog.i in isolation, valid everywhere except between L2 and L3 : L0 and L1 don't assign to k , L2 may destroy it, but, because $N \neq i$, L3 will restore $\{k \neq i\}$. But, although k is a global variable, $\{k \neq i\}$ also remains true in combination with the other prog.j's , because neither their assignment $k:= j$ $(j \neq i\ !)$, nor their assignment $k:= N$ $(N \neq i\ !)$ can destroy it.

We next observe the invariance of

$$(\underline{A}\ j\colon g[j] \Rightarrow (y[j] = fj(y)))\ \underline{or}\ k < N\ . \qquad (8)$$

Action L0 does not assign to its variables. Action L1 can only affect the implication for $j = i$; the weakest precondition of L1 for that implication is, according to the Axiom of Assignment,

$$(y = vi) \Rightarrow (y[i] = fi(y))$$

which follows, indeed, from the local assertions and the guard, for

$$y[i] = vi[i] = qi = fi(vi)\ .$$

Action L2 establishes (8) on account of its term $k < N$, and action L3 also establishes (8) because it makes all implications vacuously true.

The next invariance to be established is

$$(\underline{A}\ j\colon g[j] \Rightarrow (vj = y))\ \underline{or}\ k < N \quad . \tag{9}$$

It is, like (8), initially true because then all the $g[j]$ are false; actions L0 and L1 can affect in (9) only the implication for $j = i$, but make that implication true, action L2 establishes the truth of (9) on account of its term $k < N$, and action L3, again, makes all implications vacuously true.

The next invariant relation is

$$k < N \Rightarrow \underline{non}\ g[k] \quad . \tag{10}$$

Action L0 does not affect its variables, action L1 does not do so on account of the local assertion $\{k \neq i\}$, action L3 makes (10) vacuously true. Action L2 leaves (10) clearly invariant if, initially, $k < N$; only if initially $k = N$, we need for L2 a more elaborate argument, for we have to show that then, initially, $\underline{non}\ g[i]$ holds. We shall demonstrate this by deriving a contradiction from the assumption $k = N\ \underline{and}\ g[i]$. From this assumption and (8) we conclude $y[i] = fi(y)$, from this assumption and (9) we conclude $vi = y$, hence $y[i] = fi(vi)$: from the local assertions and the guard, however, we derive $y[i] = vi[i] \neq qi = fi(vi)$, which gives the required contradiction. This concludes the demonstration of the invariance of (10).

On account of (10), $(\underline{A}\ j\colon g[j]) \Rightarrow k = N$, and hence, on account of (8), (9) we can conclude that $(\underline{A}\ j\colon g[j]) \Rightarrow (\underline{A}\ j\colon y[j] = fj(y)\ \underline{and}\ vj = y)$. This concludes our treatment of PR7.

* * *

We now introduce $d[i]$'s and $e[i]$'s, for the time being considered as ghost-variables. They are initialized as false.

```
prog.i:  do <E j: non g[j]> →                                          PR8
L0:             <d[i]:= true; vi:= y>;
                qi:= fi(vi);
L1:             if vi[i] = qi → <g[i]:= (y = vi); e[i]:= d[i]>
L2:              ▯ vi[i] ≠ qi → <y[i]:= qi; (A j: d[j]:= false);
                                    if k < N → skip ▯ k = N → k:= i fi>;
```

```
L3:                                   < (A j: g[j]:= false; e[j]:= false);
                                        k:= N >
                        fi
        od
```

In addition to the invariance of (8), (9), and (10) we establish the invariance of

$$(\underline{A}\ j:\ d[j] \Rightarrow (vj = y)) \quad . \tag{11}$$

Relation (11) is true to start with, L0 leaves it invariant, and so do L1, L2, and L3 .

But now we are in a position to establish

$$(\underline{A}\ j:\ e[j] \Rightarrow g[j]) \quad , \tag{12}$$

because L0, L2, and L3 leave it trivially invariant, and L1 does so on account of (11) .

From (12) we deduce that $(\underline{A}\ j:\ e[j]) \Rightarrow (\underline{A}\ j:\ g[j])$. Hence, the program is still correct if we turn the e's and the d's into non-ghost-variables, and replace the outer guard by $< \underline{E}\ j:\ \underline{non}\ e[j] >$. After that replacement, however, we can regard the g's as ghost-variables! Removing the operations on the g's and on k , we get

```
prog.i:     do < E j: non e[j] > →                                        PR9
                  < d[i]:= true ; vi:= y > ;
                  qi:= fi(vi);
                  if vi[i] = qi → < e[i]:= d[i] >
                   ▯ vi[i] ≠ qi → < y[i]:= qi; (A j: d[j]:= false) > ;
                                   < (A j: e[j]:= false) >
                  fi
            od
```

* * *

(The above three stars stand for an interval of about two weeks, during which I wrote EWD623 through EWD626, while C.S.Scholten continued to think about his problem. As I have seen his work in the meantime, the following is unavoidably heavily influenced by his results.)

In my next refinement, I start again from PR5 (or PR5') , but wish this time to replace the last line, which is effectively

$$\langle (\underline{A}\ j:\ e[j]:= \text{false}) \rangle$$

by
$$(\underline{A}\ j:\ \langle e[j]:= \text{false} \rangle)$$

i.e. the single grain that sets all the $e[j]$'s false should be broken up into N little grains, each setting a single $e[j]$. The single global ghost-boolean is no longer sufficient, nor is the single global ghost-integer from PR5' . We propose to introduce for each prog.i a boolean ghost-array ri , with elements ri[0] through ri[N-1] , all initialized at false, and each ri[j] representing prog.i's "obligation" to set e[j] to false.

```
prog.i:   $\underline{do}\ \langle \underline{E}\ j:\ \underline{non}\ e[j] \rangle \to \{(\underline{A}\ j:\ \underline{non}\ ri[j])\}$                    PR10
L0:              $\langle \underline{if}\ y[i] = fi(y) \to e[i]:= \text{true} \rangle$
L1:                $[]\ y[i] \neq fi(y) \to \{Ri\}\ y[i]:= fi(y);$
                                    $(\underline{A}\ j:\ ri[j]:= \text{true}) \rangle ;$
L2j:                                $(\underline{A}\ j:\ \langle e[j],\ ri[j]:= \text{false},\ \text{false} \rangle )$
                 $\underline{fi}$
          $\underline{od}$
```

The first atomic action has two labels, labelling its alternative courses of action; on the last line we have condensed N labels. It is clear that $(\underline{A}\ j:\ \underline{non}\ ri[j])$ is an invariant of prog.i's repeatable statement. (Remember that the ghost-variable ri is local to prog.i .) Again we have to prove that

$$(\underline{A}\ j:\ e[j]) \Rightarrow (\underline{A}\ j:\ y[j] = fj(y)) \qquad (13)$$

This conclusion (13) is justified, provided we can find N predicates Rj , such that

$$(\underline{A}\ j:\ (y[j] \neq fj(y)) \Rightarrow Rj) \qquad (14)$$

and
$$(\underline{A}\ j:\ e[j]) \Rightarrow (\underline{A}\ j:\ \underline{non}\ Rj) \quad . \qquad (15)$$

Intuitively --that is what (14) says-- Rj may be interpreted as "it is uncertain whether the j-th equation of (2) is satisfied. We shall, however, define Rj quite differently --as will be shown in a moment, in a way such that (15) is obviously satisfied-- and then prove the invariance of (14) .

Because (15) can be rewritten as

$$(\underline{E}\ j\colon Rj) \Rightarrow (\underline{E}\ j\colon \underline{non}\ e[j]) \quad ,$$

an analogy with the marking process of the on-the-fly garbage collection, indeed, presents itself. In the latter we had relations like "the existence of a white reachable node implies the existence of a grey node", or more precisely "for each white reachable node, there exists a grey node from which it can be reached via (what we called) a propagation path." In other words, (15) is trivially satisfied if we can define Rj to be true if and only if node j is in some sort of transitive closure starting from the nodes with a false e . (If all the e's are true, the set of starting points, and therefore the whole transitive closure is empty.)

A bold guess is to interpret the truth of $ri[j]$ as the presence of an arrow from node nr.i to node nr.j and to interpret Rj as $\underline{non}\ e[j]$ or reachable via a directed path from another e that is false. In formula

$$Rj = (\underline{non}\ e[j]\ \underline{or}\ (\underline{E}\ k\colon Rk\ \underline{and}\ rk[j])) \qquad \text{(see EWD622-18)} \tag{16}$$

from which (15) follows. Because initially all $e[j]$'s are false, all Rj's are initially true; we have thus established the initial truth of (14), the invariance of which will be demonstrated now.

The choice L0 leaves (14) invariant: its implications for $j \neq i$ are left unaffected because their antecedents remain (trivially) unaffected, and because their consequents are left unaffected on account of (16) and the fact that L0 is executed under the circumstance that node nr.i has no outgoing arrows (remember $(\underline{A}\ j\colon \underline{non}\ ri[j])$). The implication for $j = i$ is and remains vacuously true on account of the falsity of its antecedent, as implied by the guard.

The choice L1 leaves (14) invariant. On account of the guard and the initial truth of (14) we conclude that it can only be chosen when Ri holds. Because the truth of Ri is not destroyed by the creation of arrows, and because of (16)

$$(\underline{A}\ k, j\colon (Rk\ \underline{and}\ rk[j]) \Rightarrow Rj) \tag{17}$$

L1 establishes Rj for all j , i.e. upon completion each implication of (14) holds on account of its true consequent.

Also each of the individual actions L2j leaves (14) invariant, because on account of (16), removal of an incoming arrow of node j , together with e[j]:= false can never cause for Rj --and hence for any other Rk-- the transition from true to false.

This could complete our treatment of PR10. There is, however, a little bit more that might be worth observing. If it is the sole purpose of the arrows to propagate from nodes with non e the property R , and, no obviously redundant arrows are retained, we may hope that even

$$(\underline{A}\ k,\ j:\ rk[j] \Rightarrow (Rk\ \underline{and}\ Rj)) \qquad (18)$$

is invariantly true.

We have already observed that choice L0 cannot affect Rj for $j \neq i$. If, initially, node nr.i has an incoming arrow, i.e. there exists a k such that rk[i] holds , then $k \neq i$ because of non ri[i] ; then (18) tells us, that initially Rk is true. We have just established that Rk then remains true, and on account of (17), Ri remains true. If node nr. i has no incoming arrows, Ri becoming false can do no harm to (18) , because it has no outgoing arrows either when L0 is executed.

L1 does not violate (18) because it is only executed under the truth of Ri and all Rj are certainly true upon completion.

L2j does not violate (18) either. Because the ri[j] are local ghost-variables of prog.i , the initial truth of ri[j] is obvious; therefore (18) tells us that Rj holds initially and the assignment e[j]:= false ensures that Rj holds upon completion. Hence we can conclude that any act L2j leaves all Rj unchanged. Therefore, all right-hand sides of (18) are constant; only one antecedent is strengthened, and thus (18) is indeed an invariant.

Having established that any act L2j leaves all Rj unchanged, that L1 can only cause for Rj a transition from false to true, and that L0 can only affect Ri , we see that the truth of Ri is not destroyed by any prog.j for $j \neq i$, and that only L0 of prog.i can set Ri to false.

* * *

(The above three stars stand for a two-hour failure to prove the correctness of the next version without the introduction of more ghost-variables, followed by a restless night.)

Encouraged by the success of the ri's and the Ri's I shall now try to combine the introduction of the vi from PR9 with the chopping up of the false-setting of the e[j]'s from PR10 . I think that this text should not become too repetitive and that I should make a larger jump : in addition I shall also separate the false-setting of the d[j]'s from the assignment to y[i], and furthermore the false-setting of the d[j]'s will be chopped up. Analogous to the ri[j]'s we introduce qi[j]'s to record prog.i's "obligation" to set d[j] to false.

In my treatment of PR10 I dislike that the nice relation (18) could only be derived at the end. In order to derive it earlier, I shall try a new proof experiment: I intend to strengthen guards of the alternative construct by adding "ghost-constraints" and show eventually that the strengthening was ineffective because the truth of the added term is implied by the truth of the guard it was supposed to strengthen. The choice of the strengthening is inspired by my desire to keep the initial proof of the invariance of (18) simple. (Because the strengthened guards contain ghost-variables, I have placed them between (temporary) angle brackets.) We consider the following program, where Ri is defined as by (16) .

```
prog.i:      do < E j: non e[j] > → {A j: non ri[j]}                          PR11
L0:                < d[i]:= true; vi:= y > {y[i] = vi[i]};
                   qi:= fi(vi) {qi = fi(vi)};
L1:                if vi[i] = qi → {y[i] = fi(vi)} < e[i]:= d[i] >
                    ▯ < vi[i] ≠ qi and Ri > → {y[i] ≠ fi(vi)}
L2:                          < y[i]:= qi; (A j: qi[j], ri[j]:= true, true ) > ;
L3j:                         (A j: {ri[j]} < d[j], qi[j]:= false, false > );
L4j:                         (A j: {ri[j]} < e[j], ri[j]:= false, false > )
                   fi
             od
```

L0 and L3j can trivially not affect any Rj. L4j , although it removes incoming arrows for node nr.j , can never cause for Rj a transition from true to false, as it leaves Rj true on account of the final $\underline{\text{non}}\ e[j]$. Action L2, that only adds arrows, cannot effectuate for Rj a transition from true to false either. Hence, L1 is the only action that can do so. But because L1 is executed under absence of outgoing arrows, it can only do so for Ri ; hence all through the second alternative Ri which occurs in the guard is invariantly true; hence --on account of (17)-- action L2 makes all Rj true, and --as $Ri\ \underline{\text{and}}\ ri[j]$ is a pre-condition for L4j - actions L4j find and leave the Rj's true.

Now we are ready to prove for PR11 the invariance of

$$(\underline{A}\ k,\ j\colon\ rk[j] \Rightarrow (Rk\ \underline{\text{and}}\ Rj)) \qquad (18)$$

L0 and L3j trivially don't affect (18), L4j leaves the consequents unaffected and only strengthens an antecedent, L2 makes all consequents true and L1 does not violate (18) either, as it can only set Ri false in the absence of incoming arrows --as the existence of a $Rk\ \underline{\text{and}}\ rk[i]$ will keep it true-- and L1 is executed under the absence of outgoing arrows.

The next step is to draw as quickly as possible the relevant conclusion for which we need the qi[j]'s, and to eliminate them from then onwards from our consideration. We prove the invariance of

$$(\underline{A}\ j\colon\ (vj \neq y\ \underline{\text{and}}\ d[j]) \Rightarrow (\underline{E}\ k\colon\ qk[j])) \qquad (19)$$

L0 can only affect the i-th implication, but leaves its antecedent false, action L1 does affect none, L2 leaves all consequents true, L3j can only affect the j-th implication, but it leaves its antecedent false, and L4j affects none. Initially all antecedents are false, and the universal validity of (19) has been established.

Because --remember that the ri and qi are local variables of prog.i !-- it is easily established that $(\underline{A}\ k,\ j;\ qk[j] \Rightarrow rk[j])$, we can deduce from (19)

$$(\underline{A}\ j\colon\ (vj \neq y\ \underline{\text{and}}\ d[j]) \Rightarrow (\underline{E}\ k\colon\ rk[j])) \quad . \qquad (20)$$

From now on we won't refer to the qi's anymore; we shall need (20) once.

In order to prove the invariance of (14) we may expect --because such a circumstance is not unusual at all-- to have to strengthen it. I propose to do so by weakening the antecedents $y[j] \neq fj(y)$, because in view of the local assertions in the alternative clause of PR11 it seems attractive to replace them by

$$y[j] \neq fj(vj)\ \underline{\text{or}}\ y \neq vj$$

(from the negation of which $y[j] = fj(y)$ duly follows). Because we also expect $d[j]$ to hold eventually, it seems safe to weaken the antecedents still further by adding the term "$\underline{\text{or}}\ \underline{\text{non}}\ d[j]$" . Thus we arrive, inspired by (14), at our tentative invariant relation --initially trivially true--

$$(\underline{A}\ j\colon (y[j] \neq fj(vj)\ \underline{\text{or}}\ y \neq vj\ \underline{\text{or}}\ \underline{\text{non}}\ d[j]) \Rightarrow Rj) \tag{21}$$

Action L2 , which sets all consequents true, is harmless, action L3j can only affect the j-th implication, but is harmless because L3j is executed under the invariant truth of $ri[j]$ and on account of (18) under the invariant truth of its consequent Rj . Any action L4j is trivially harmless now we have already established that it leaves the Rj's unaffected. We are left with L0 and L1 .

Action L0 leaves the consequents unchanged and can only affect the antecedent for $j = i$: in that case it suffices to show that a false antecedent remains false, i.e. with P the negation of the antecedent

P: $$y[i] = fi(vi)\ \underline{\text{and}}\ y = vi\ \underline{\text{and}}\ d[i]$$

we have to show that

$$P \Rightarrow wp(\text{"}< d[i]{:=}\ \text{true};\ vi{:=}\ y >\text{"},\ P) \quad .$$

The Axiom of Assignment defines this weakest pre-condition as

$$y[i] = fi(y)\ \underline{\text{and}}\ y = y\ \underline{\text{and}}\ \text{true} \quad .$$

The last two terms are true all by themselves, the truth of the first term is implied by the first two terms of P , and hence L0 leaves (21) invariant.

But what about L1 ? We have established that L1 does not affect Rj for $j \neq i$; for $j \neq i$, it cannot affect the antecedents either, so we only need to worry about the i-th implication of (21). The assignment $< e[i]{:=}\ d[i] >$, which leaves its antecedent unaffected, can only violate the implication by making the consequent Ri false while the antecedent remains true . A necessary initial condition for $< e[i]{:=}\ d[i] >$ to make Ri false --see

(16) and (18)-- is

$$d[i]\ \underline{\text{and}}\ \underline{\text{non}}\ (\underline{E}\ k\colon\ rk[i])\quad .$$

Combined with the truth of the antecedent, we derive

$$(y[i] \neq fi(vi)\ \underline{\text{or}}\ y \neq vi)\ \underline{\text{and}}\ d[i]\ \underline{\text{and}}\ \underline{\text{non}}\ (\underline{E}\ k\colon\ rk[i])\quad .$$

Combined with the local assertion $y[i] = fi(vi)$ as derived from the guard, we get

$$y \neq vi\ \underline{\text{and}}\ d[i]\ \underline{\text{and}}\ \underline{\text{non}}\ (\underline{E}\ k\colon\ rk[i])\quad .$$

But on account of (20) this is false: also L1 does not destroy the validity of (21), whose invariance has now been established.

We are left with the obligation to show that the ghost-guard Ri can be omitted. The local assertion $y[i] \neq fi(vi)$ as derived from the guard implies Ri with the help of (21), of which we regard the invariance as established. And this completes the correctness proof of

```
prog.i:      do < E j: non e[j] > →
                   < d[i]:= true; vi:= y > ;                       PR12
                   qi:= fi(vi);
                   if vi[i] = qi → < e[i]:= d[i] >
                    ▯ vi[i] ≠ qi → < y[i]:= qi > ;
                                   (A j: < d[j]:= false > );
                                   (A j: < e[j]:= false > )
                   fi
             od
```

Remark. C.S.Scholten's proof allows for the further chopping up of the second line into

$$< d[i]:= \text{true} > ;\ (\underline{A}\ j\colon < vi[j]:= y[j] >);$$

I think that at this stage I leave that last proof as an exercise for the reader. (End of remark.)

Concluding remarks.

In one respect I consider the way in which this report has developed as a little bit disappointing: the constructive flavour of its beginning has largely disappeared from PR10 onwards. Rather than to verify a posteriori I prefer to merge and synthesize proof and program developments. In sequential programming this art has been raised to a considerable height; when I was halfway this report I saw the same merge and synthesis emerging during multiprogram development. This observation excited me, as it would raise the Gries/Owicki theory more clearly to the status of a tool for construction. Perhaps I should not allow myself to be too much disappointed by the disappearance of the constructive flavour: there wasn't much program to be invented anymore, and, besides that, I was of course biased by having seen Scholten's work.

In other respects I am extremely pleased with it. I have discovered at least two tricks that were new for me: the change of ghost-variables into non-ghost-variables and vice versa and --probably more generally applicable than the first trick-- the temporary strengthening of guards by adding "ghost-constraints". I feel that the latter has done a great deal in smoothing the correctness proof for PR12; in any case it seems a very neat way for preventing circular arguments.

Furthermore we have now at least a workable --be it partial-- grip on a canonical problem that I have shunned for at least four years, since the moment I designed self-stabilizing systems, and that is the general problem of the detection that in such distributed system the stabilization towards the legitimate states has been completed.

The development of this report was not easy: quite regularly it has strained my agility in the propositional calculus, but I guess that I can learn it. (It was certainly a good training.) In any case it shows --to my taste even convincingly-- the feasibility of departing from the usual operational arguments, in which one tries to visualize classes of computational histories; furthermore it shows the vast superiority of the non-operational arguments --once they have been found!-- over the traditional ones.

Acknowledgements. I am greatly indebted to C.S.Scholten for drawing my attention again to this problem, and for contributing so much to its solution. (He was the first to see clearly the analogy with the garbage collector, and to transfer the notion of "reachability via a path" into the solution of this problem.) Further I am --as usual-- indebted to the regular members of the "Tuesday Afternoon Club". (End of acknowledgements.)

Explanation. This was the first project I embarked upon, shortly after Hoare, Knuth and Traub had given me reason to be grateful to them. Hence the dedication, in great gratitude and not without some pride. (End of explanation.)

26 May 1977
Plataanstraat 5
5671 AL NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow

Note added later concerning (16) on page EWD622 - 11.

Relation (16) is correct in as far as that it certainly holds. If we want to use it to define the Rj as a solution of (16), we must add the remark that the Rj's then must be the minimal solution, i.e. the solution with as few Rj's true as possible; this, because the arrows may form cyclic paths. (End of note added later.)