Finding the correctness proof of a concurrent program.

(Those who have seen EWD622 will recognize the following as an improved treatment of one of the versions of the concurrent program developed in that report. The main improvement consists of the heuristics for finding the correctness proof: the heuristics effectively buffer the shock of invention which, in EWD622 - 11, was indicated by "A bold guess is to interpret....".

For the benefit of those who have not seen EWD622, this note is written as a self-contained text that fully redescribes the problem. They have furthermore the advantage that they won't be confused by changed notations and meanings of variables.)

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

$$y = f(y) \tag{1}$$

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

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

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

$$< y[i] := fi(y) > \tag{3}$$

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 between them mutual exclusion in time (when they are considered "to take time"). In order to guarantee termination we must assume that the sequence of i-values for which the assignments (3) are carried out must be one of some sort of "fair random order" in which, for instance, a finite upper bound is known to exist for the 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 somehow guaranteed. (He who refuses to make that assumption can read the following as a proof of partial correctness.)

For the purpose of this note it suffices to know that functions f exist such that with a proper value of y equation (1) will be solved by a finite number of assignments (3). How for a given f and initial value y this property can be established is <u>not</u> the subject of this paper. (He who refuses to assume that the function f has that delightful property is free to do so: he can, again, read the following as a proof of partial correctness that states that <u>when</u> our concurrent program has terminated, (1) is satisfied.)

Besides the global vector y there is a global boolean array h, with elements $h[i]$ for $0 \le i < N$, all of which are true to start with. We now consider the following program of N-fold concurrency, in which each atomic action modifies at most one global array element. We give the program first and shall explain the notation afterwards.

The concurrent program we are considering consists of the following N components $(0 \le i < N)$:

```
comp.i:
   L0:  do < (E j: h[j]) > →
   L1:          < if y[i] = fi(y) → h[i]:= false >
                  [] y[i] ≠ fi(y) → y[i]:= fi(y) > ;
   L2j:                              (A j: < h[j]:= true > )
                  fi
        od
```

In line L0, $(\underline{E}\ j: h[j])$ is an abbreviation for $(\underline{E}\ j: 0 \le j < N: h[j])$; for the sake of brevity we shall use this abbreviation throughout this note. By writing $< (\underline{E}\ j: h[j]) >$ in the guard we have made the inspection whether a true $h[j]$ can be found into an atomic action.

The opening angle bracket "<" in L1 has two corresponding closing brackets, corresponding to the two "atomic alternatives"; it means that in the same atomic action the guards are evaluated and either "$h[i]$:= false" or "$y[i]$:= $fi(y)$" is executed. In the latter case, N separate atomic actions follow, each setting an $h[j]$ to true: in line L2j we have used the abbreviation $(\underline{A}\ j: < h[j]:= \text{true} >)$ for the program that performs the N atomic actions $< h[0]:= \text{true} >$ through $< h[N-1]:= \text{true} >$ in some order which we don't specify any further.

Our target state is that y is a solution of (1), or, more explicitly

$$(\underline{A}\ j:\ y[j] = fj(y)) \quad . \qquad (4)$$

We first observe that (4) is an invariant of the repeatable statements: in the alternative constructs, always the first alternative will be chosen, leaving y , and hence (4) unaffected. We can even conclude a stronger invariant

$$\underline{non}\ (\underline{E}\ j:\ h[j])\ \underline{and}\ (\underline{A}\ j:\ y[j] = fj(y)) \qquad (5)$$

or, equivalently

$$(\underline{A}\ j:\ \underline{non}\ h[j])\ \underline{and}\ (\underline{A}\ j:\ y[j] = fj(y)) \qquad (5')$$

for, when (5) holds, no assignment h[i]:= false can destroy the truth of (<u>A</u> j: <u>non</u> h[j]). When (4) holds, the assumption of fair random order implies that within a finite number of steps (5) will hold. But then the guards of the repetitive constructs are false, and all components will terminate nicely with (4) holding. The critical point is: can we guarantee that none of the components terminates too soon? In order to prove that termination implies that (4) holds, we have to prove the universal truth of

$$(\underline{E}\ j:\ h[j])\ \underline{or}\ (\underline{A}\ j:\ y[j] = fj(y)) \quad . \qquad (6)$$

Relation (6) certainly holds when the N components are started because initially we start with all h[j] true. We are only left with the obligation to prove the invariance of (6); the remaining part of this report is devoted to that proof, and to how it can be discovered.

We get a hint of the kind of difficulties we may expect when trying to prove the invariance of (6) as soon as we realize that the first term is a compact notation for

$$h[0]\ \underline{or}\ h[1]\ \underline{or}\ h[2]\ \underline{or}\ \dots\ \underline{or}\ h[N-1]$$

which can become false when, as a result of "h[i]:= false" the <u>last</u> true h[j] disappears. That is ugly!

Proving a mathematical theorem is often only feasible by proving a stronger --but, somehow, more manageable-- theorem instead. In direct analogy: instead of trying to prove the invariant truth of (6) we shall try to prove the invariant truth of a stronger assertion that we get by replacing the conditions $y[j] = fj(y)$ by stronger ones. Because under the universal truth of $(Q \underline{or} R)$, the relation $\underline{non}\ R$ is stronger than Q , we can strengthen (6) into

$$(\underline{E}\ j{:}\ h[j])\ \underline{or}\ (\underline{A}\ j{:}\ \underline{non}\ Rj) \qquad (7)$$

provided

$$(\underline{A}\ j{:}\ y[j] = fj(y)\ \underline{or}\ Rj) \qquad (8)$$

holds universally. (Someone who sees these heuristics presented in this manner for the first time may experience this as juggling, but I am afraid that it is quite standard and that we had better get used to it.)

What have we gained by the introduction of the N predicates Rj ? Well: the freedom to choose them! More precisely: the freedom to define them in such a way that we can prove the universal truth of (8) --which is structurally quite pleasant-- while the universal truth of (7) --which is structurally equally "ugly" as (6)-- follows more or less directly from the definition of the Rj's : that is the way in which we may hope that (7) is more "manageable" than the original (6).

In order to find a proper definition of the Rj's , we analyse our obligation to <u>prove</u> the invariance of (8).

If we only looked at the invariance of (8), one might think, that a definition of the Rj's in terms of y:

$$Rj = (y[j] \neq fj(y))$$

would be a sensible choice. A moment's reflection tells us that that definition does not help: it would make (8) universally true by definition, and the right-hand terms of (6) and (7) would be identical, whereas (7) was intended to be stronger than (6).

For two reasons we are looking for a definition of the Rj's in which the y does <u>not</u> occur: firstly, it is then that we can expect the proof of the universal truth of (8) to amount to something --and, therefore, to contribute to the argument-- , secondly, we would like to conclude the universal truth of (7) --which does not mention y at all-- from the <u>definition</u> of the Rj's. In other words, we propose a definition of the Rj's which does not refer to y at all: only with such a definition the replacement of (6) by (7) and

(8) localizes our dealing with y completely to the proof of the universal truth of (8).

Because we want to define the Rj's independently of y , because initially we cannot assume that for some j-value y[j] = fj(y) holds, and because (8) must hold initially, we must assume that initially

$$(\underline{A}\ j:\ Rj) \qquad (9)$$

holds. Because, initially, all the h[j] are true, the initial truth of (9) is guaranteed if the Rj's are defined in such a way that we have

$$(\underline{E}\ j:\ \underline{non}\ h[j])\ \underline{or}\ (\underline{A}\ j:\ Rj) \quad . \qquad (10)$$

We observe, that (10) is again of the recognized ugly form we are trying to get rid of. We have some slack --that is what the Rj's are being introduced for-- and this is the moment to decide to try to come away with a stronger --but what we have called: "structurally more pleasant" -- relation such as

$$(\underline{A}\ j:\ \underline{non}\ h[j]\ \underline{or}\ Rj) \qquad (11)$$

from which (10) immediately follows. We can already divulge that, indeed, (11) will be one of the defining equations for the Rj's .

From (11) it follows that the algorithm will start with all the Rj's true. From (8) it follows that the truth of Rj can be appreciated as "the equation y[j] = fj(y) need not be satisfied", and from (7) it follows that in our final state we want to have all the Rj's equal to false.

Let us now look at the alternative construct

```
L1:   < if y[i] = fi(y) → h[i]:= false >
         ▯ y[i] ≠ fi(y) → y[i]:= fi(y) > ;
L2j:                  (A j: < h[j]:= true > )
        fi
```

We observe that the first alternative sets h[i] false, and that the second one, as a whole, sets all h[j] true. As far as the universal truth of (11) is concerned, we therefore conclude that in the first alternative Ri is allowed to, and hence <u>may</u> become false, but that in the second alternative as a whole, all Rj's <u>must</u> become true.

Let us now confront the two atomic alternatives with (8). Because, when the first alternative is selected, only $y[i] = fi(y)$ has been observed, the universal truth of (8) is not destroyed by it, provided:

In the execution of the first atomic alternative

$$< y[i] = fi(y) \rightarrow h[i]:= \text{false} >$$

<u>no</u> Rj for $j \neq i$ may change from true to false. (12)

Confronting the second atomic alternative

$$< y[i] \neq fi(y) \rightarrow y[i]:= fi(y) >$$

with (8), and observing that upon its completion <u>none</u> of the relations $y[j] = fj(y)$ needs to hold, we conclude that the second atomic alternative itself must already cause a final state in which all the Rj's are true, in spite of the fact that the subsequent assignments $h[j]:=$ true --which would each force an Rj to true on account of (11)-- have not been executed yet. In short: in our definition for the Rj's we must include besides (11) another reason why an Rj should be defined to be true.

As it stands, the second atomic alternative only modifies y , but we had decided that the definition of the Rj's would not be expressed in terms of y ! The only way in which we can formulate the additional reason for an Rj to be true is in terms of an <u>auxiliary</u> variable (to be introduced in a moment), whose value is changed in conjunction with the assignment to $y[i]$. It has to force each Rj to true until the subsequent assignment $< h[j]:=$ true $>$ does so via (11). Because the second atomic alternative is followed by N subsequent, separate atomic actions $< h[j]:=$ true $>$ --one for each value of j -- it stands to reason that we introduce for comp.i an auxiliary local boolean array ri with elements $ri[j]$ for $0 \leq j < N$. Their initial (and "neutral") value is true. The second atomic alternative of L1 sets them all to false, the atomic statements L2j will reset them to true one at a time.

In the following annotated version of comp.i we have inserted local assertions between braces. In order to understand the local assertions about ri it suffices to remember that ri is local to comp.i . The local assertion Ri in the second atomic alternative of L1 is justified by the guard

$y[i] \neq fi(y)$ in conjunction with (8). We have further incorporated in our annotation the consequence of (12) and the fact that the execution of a second alternative will never cause an Rj to become false: a true Ri can only become false by virtue of the execution of the first atomic alternative of L1 by comp.i itself! Hence, Ri is true all through the execution of the second alternative of comp.i .

```
comp.i:
     L0:   do < (E j: h[j]) > → {(A j: ri[j])}
     L1:         < if y[i] = fi(y) → h[i]:= false > {(A j: ri[j])}
                   ▯ y[i] ≠ fi(y) →
                          {Ri} y[i]:= fi(y);
                          (A j: ri[j]:= false) > {Ri and (A j: non ri[j])};
     L2j:                 (A j:{Ri and non ri[j]} < h[j]:= true; ri[j]:= true > )
                 fi {(A j: ri[j])}
           od
```

On account of (11) Rj will be true upon completion of L2j. But the second atomic alternative of L1 should already have made Rj true, and it should remain so until L2j is executed. The precondition of L2j, as given in the annotation, hence tells us the "other reason besides

$$(\underline{A}\ j:\ \underline{non}\ h[j]\ \underline{or}\ Rj) \qquad (11)$$

why an Rj should be defined to be true":

$$(\underline{A}\ i, j:\ \underline{non}\ Ri\ \underline{or}\ ri[j]\ \underline{or}\ Rj) \qquad . \qquad (13)$$

Because it is our aim to get eventually all the Rj's false, we _define_ the Rj's as the _minimal_ solution of (11) and (13), minimal in the sense of: as few Rj's true as possible.

A second look shows how the minimal solution is found. It is a sort of transitive closure: starting with the set of Rj's forced true by (11) --on account of falsity of _non_ h[j]-- , if necessary we extend this set --possibly in cascades-- with the Rj's forced true by (13) --on account of falsity of _non_ Ri _or_ ri[j]-- .

For a value of i , for which

$$(\underline{A}\ j:\ ri[j]) \qquad (14)$$

holds, the truth of Ri forces no further true Rj's via (13); consequently, when such an Ri becomes false, no other Rj-values are then affected. This, and the fact that the first atomic alternative of L1 is executed under the truth of (14) tells us, that with our definition of the Rj's requirement (12) is, indeed, met.

We have proved the universal truth of (8) by defining the Rj's as the minimal solution of (11) and (13). The universal truth of (7) , however, is now obvious. If the left-hand term of (7) is false, we have

$$(\underline{A}\ j:\ \underline{non}\ h[j]),$$

and (11) and (13) have as minimal solution all Rj's false, i.e.

$$(\underline{A}\ j:\ \underline{non}\ Rj) \qquad ,$$

which is the second term of (7).

Acknowledgements. I would like to express my gratitude to both IFIP WG2.3 and "The Tuesday Afternoon Club", where I had the opportunity to discuss this problem. Those familiar with the long history that led to this note, however, know that in this case I am indebted to C.S.Scholten more than to anyone else.

Reference

EWD622 "On making solutions more and more fine-grained." by Edsger W.Dijkstra.

Plataanstraat 5
5671 AL NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow