For the record: Yossi Shiloach's Algorithm

We are given a positive integer N and two, say, integer functions A and B on the integers, which have both period N, i.e.

$$A.k = A.(k+N) \text{ and } B.k = B.(k+N) \text{ for all } k,$$

and are asked to design an algorithm determining whether they are the same function but for a possible shift of the argument, more precisely, the value of the boolean variable should be made to satisfy the postcondition

$$R: \quad eq \equiv \langle \exists i :: \langle \forall k :: A.(i+k) = B.k \rangle \rangle \quad .$$

*   *   *

Our first remark is that in the above formalization of the postcondition, the symmetry in A and B has been destroyed by the introduction of i. We can restore the symmetry by introducing a j as well, and rewrite

$$R: \quad eq \equiv \langle \exists i,j :: \langle \forall k :: A.(i+k) = B.(j+k) \rangle \rangle \quad .$$

Our next remark is that thanks to the periodicity of A and B, the universal quantification can be confined to N consecutive values of k:

$$R: \quad eq \equiv \langle \exists i,j :: \langle \forall k: 0 \leq k < N: A.(i+k) = B.(j+k) \rangle \rangle \; .$$

In the rest of this text, the symmetry between the pairs (A, i) and (B, j) will be maintained.

We first analyse the case that

$$eq := true$$

would establish R. In that case, the the algorithm would have to establish for some i, j the truth of

$$R': \quad \langle \forall k: 0 \leq k < N: A.(i+k) = B.(j+k) \rangle \quad ;$$

since the N terms of this quantified expression are independent, their truths have to be verified individually. We adopt the standard solution, i.e. we introduce a variable, h say, that satisfies

$$P: \; \langle \forall k: 0 \leq k < h: A.(i+k) = B.(j+k) \rangle \wedge 0 \leq h$$

and make the (standard) observations that

(i) $h = 0 \Rightarrow P$

(ii) the guarded command

$$A.(i+h) = B.(j+h) \rightarrow h := h+1$$

maintains the truth of P, and

(iii) $P \wedge N \leq h \wedge eq \Rightarrow R$,

which leads to the program skeleton

```
|[ var h, i, j : int
 ; h, i, j := 0, .... {P}
 ; do h < N →
      if A.(i+h) = B.(j+h) → h := h+1 fi
   od {R'}
 ; eq := true
]| {R}
```

If this program skeleton does not abort, it establishes $eq \equiv true$, as it should. If it aborts because of finding

$$A.(i+h) \neq B.(j+h) \quad ,$$

this can be for two reasons: either another i,j-combination is needed to establish R' , or $eq \equiv false$ should hold in the final state. With this in mind we shall try to supply the missing alternative

$$A.(i+h) \neq B.(j+h) \rightarrow \ldots\ldots \quad .$$

At the moment this alternative is selected, the truth of P tells us that h equalities have been established, and the values of i,j determine which. An assignment to i or j in general falsifies P and thereby destroys this information (which was time-wise expensive to collect when h is large). The question is therefore whether we can save some of it, i.e. from the situation pictorially represented by

$$\underbrace{\begin{array}{ccc} A.i & & A.(i+h-1) \\ = & \cdots\cdots & = \\ B.j & & B.(j+h-1) \end{array}}_{h} \quad \begin{array}{c} A.(i+h) \\ \neq \\ B.(j+h) \end{array}$$

Shiloach's invention has been to impose —if not already present— a total order $<$ on the values compared, i.e

$$A.(i+h) \neq B.(j+h) \equiv A.(i+h) < B.(j+h) \vee B.(j+h) < A.(i+h) .$$

Let us focus on the situation in which the left conjunct holds, i.e.

$$\underbrace{\begin{array}{ccc} A.i & & A.(i+h-1) \\ = & \cdots\cdots\cdots & = \\ B.j & & B.(j+h-1) \end{array}}_{h} \quad \begin{array}{c} A.(i+h) \\ < \\ B.(j+h) \end{array} \quad ,$$

for now we see a situation in which the

notion of "the lexical order" of strings is a relevant concept. (For two different strings, their lexical order is defined as the order of their elements in the leftmost position in which they differ.) Defining the string $SA.i$ of length N by

$$SA.i \;=\; A.i \;\; A.(i+1) \;\; ... \;\; A.(i+N-1)$$

(and $SB.j$ similarly), we observe that (i) because of the periodicity of the function A, $SA.i$ defines A completely, and (ii) the situation we were focussing – given by $P \wedge A.(i+h) < B.(j+h)$ – implies in terms of the lexical order between strings

$$\langle \forall k : 0 \leq k < h+1 : SA.(i+k) < SB.(j+k) \rangle \quad .$$

The nice thing about this conclusion about i and j is that it is still useful when simplified and weakened to a conclusion about i only, viz.

$$\langle \forall k : 0 \leq k < h+1 : SA.(i+k) < BB \rangle$$

where BB is the lexical maximum of the SB strings, in formula

$$BB = \langle \uparrow k :: SB.k \rangle \quad .$$

<u>Remark</u> After the introduction of the lexical maxima, the function of the program to be designed can be described by the assignment statement

$$eq := AA = BB \quad .$$

(End of Remark.)

Our last conclusion about i suggests that we consider

$$QA: \quad \langle \forall k : 0 \le k < i : SA.k < BB \rangle \wedge 0 \le i$$

and observe

(i) $\qquad i = 0 \Rightarrow QA$

(ii) the guarded command

$$A.(i+h) < B.(j+h) \rightarrow i := i+h+1$$

maintains the truth of QA , and

(iii) $\qquad QA \wedge N \le i \wedge (eq \equiv false) \Rightarrow R \quad .$

[ad (ii). As given, the guarded command falsifies P , but the assignment $h := 0$ remedies this.
ad (iii). From $QA \wedge N \le i$ we can conclude $AA < BB$, which implies $AA \ne BB$.]

With QB analogously defined by

QB: $\langle \forall k : 0 \le k < j : SB.k < AA \rangle \wedge 0 \le j$,

merging our results now yields the program

```
|[ var h, i, j : int
 ; h, i, j := 0, 0, 0 {inv. P ∧ QA ∧ QB}
 ; do h<N ∧ i<N ∧ j<N →
     if A.(i+h) = B.(j+h) → h:= h+1
     [] A.(i+h) < B.(j+h) → i, h := i+h+1, 0
     [] B.(j+h) < A.(i+h) → j, h := j+h+1, 0
     fi
   od
 ; eq := N≤h
]|
```

This program terminates because the repeatable statement increases the value of $h+i+j$ each time by 1 while the guard of the repetition bounds this value from above. The form of the final assignment to eq is justified by the observation that after initialization at most 1 of the conjuncts of the guard is *false* .

Austin, 16 June 1998

prof. dr Edsger W. Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188 , USA