The Lexicographic Minimum of a Cyclic Array

We consider a cyclic array $b\,(i: 0 \leq i \wedge i < N)$ of integers, $1 \leq N$. It being cyclic means that for any i we have $b.i = b.(i+N)$. In terms of it we define, for any h, sequence $X.h$ of length N, as follows

$$X.h \;=\; b\,(i: h \leq i \wedge i < h+N) \;.$$

As a result we have $X.h = X.(h+N)$, so that there are at most N different sequences X. Our purpose is to identify a lexicographically minimal one. More precisely, with sequence M given by

(0) $\quad (\underline{E}h: 0 \leq h \wedge h < N: M = X.h)$

(1) $\quad (\underline{A}h: 0 \leq h \wedge h < N: M \leq X.h)$,

we wish to construct a program that, for some variable r, has

R: $\quad M = X.r$

as a postcondition. Moreover, we aim at a program with a "time complexity" that is linear in N.

* * *

Our target program will contain a repetitive construct and we therefore must propose an invariant. A number of candidate invariants readily present themselves. One of them is

$$(\underline{A}h:\ 0 \le h \wedge h < n:\ X.r \le X.h)\ .$$

It is easily initialized, and conjoined with $n = N$ – the negation of an acceptable guard – it implies R.

Another possible invariant is

$$(\underline{E}h:\ r \le h \wedge h \le s:\ M = X.h)\ ;$$

together with $r = s$ it implies R.
Now someone may come up with a third or a fourth proposal, but meanwhile we get saddled with the problem of which invariant to choose

In view of the relatively high demand on the required efficiency of the program, we do not want to choose an invariant too lightheartedly, because the price to be paid for making too airy a choice could become pretty high. Not only could it hamper us in attaining the efficiency goal, but – much worse – it could prevent us from keeping the design simple. Therefore, we rather analyze our programming problem a little more cautiously, and then design our invariant from our findings.

* * *

Our first observation is that, whatever algorithm we may compose, carrying out a lexicographic comparison between two distinct sequences $X.p$ and $X.q$ seems

to be an indispensable ingredient. Since such sequences are to be compared element-wise and "from left to right", we will somehow need to investigate invariant relation

$$P: \quad 0 \le d \;\wedge\; (\underline{A}\, i : 0 \le i \wedge i < d : X.p.i = X.q.i),$$

for $p \neq q$. Two obvious ways of handling it are by

$$d := 0,$$

which establishes P, and by

$$X.p.d = X.q.d \;\rightarrow\; d := d+1,$$

which maintains P. The latter immediately raises the question of what we can say in the case $X.p.d \neq X.q.d$

Here, the notion of lexicographic order forces us to distinguish between the cases

(2) $X.p.d < X.q.d$

and $X.q.d < X.p.d$. Fortunately, relation P is symmetric in p and q so that the case analysis does not hurt.

By the definition of lexicographic order, P conjoined with (2) implies that sequence $X.p$ lexicographically precedes sequence $X.q$, i.e.

$$X.p < X.q \quad .$$

Can this be of any significance for our purpose of identifying sequence M ? From (1) - one of the two things we know about M - we have $M \leq X.p$, so that together with $X.p < X.q$ we conclude

$$M \neq X.q \ .$$

As a result, we have identified one "non-M" sequence.

But the result is not nearly strong enough, since we have identified only one "non-M" sequence at the expense of d+1 element comparisons, viz. those recorded in P and in (2) . If we cannot perform substantially better, we will simply arrive at an algorithm that is quadratic in N . Perhaps, this is the best we can do if nothing more specific is known about the sequences X . But in our example they are tightly related.

Relation P attracts our attention to common prefixes of X.p and X.q . Let H be such a prefix, and let h be its length. From P we then conclude that for any H such that $0 \leq h \wedge h < d+1$,

(i) $X.p = H \# Y$ and

(ii) $X.q = H \# Z$,

for some Y and Z . (Symbol # stands for concatenation.) Now let us see how we can manipulate $X.p < X.q$ (, which

was the conclusion that we drew from P and (2)):

$$
\begin{array}{ll}
 & X.p < X.q \\
= & \quad \{ (i) \text{ and } (ii) \} \\
 & H \# Y < H \# Z \\
= & \quad \{ \text{property of lexicographic order} \} \\
 & Y < Z \\
= & \quad \{ \text{property of lexicographic order} \} \\
 & Y \# H < Z \# H \ .
\end{array}
$$

It is here where the tight relationship between the sequences X enters the picture. From (i) and the definition of X we can derive

(iii) $\quad X.(p+h) = Y \# H \ ,$

and with the aid of (ii)

(iv) $\quad X.(q+h) = Z \# H \ .$

Now we continue the above calculation:

$$
\begin{array}{ll}
 & Y \# H < Z \# H \\
= & \quad \{ (iii) \text{ and } (iv) \} \\
 & X.(p+h) < X.(q+h) \\
\Rightarrow & \quad \{ \text{with } (1) \} \\
 & M \neq X.(q+h) \ .
\end{array}
$$

In summary, we have derived

$$
\begin{array}{l}
P \wedge \ X.p < X.q \\
\Rightarrow \ (\underline{A}h : 0 \leq h \wedge h < d+1 : M \neq X.(q+h)) \ ,
\end{array}
$$

or, by dummy transformation $h := h-q$,

(3) $P \wedge X.p < X.q$
$\Rightarrow$
$(\underline{A}h: q \leq h \wedge h < q+d+1: M \neq X.h)$.

As a result, we have identified $d+1$ "non-M" sequences at the expense of $d+1$ element comparisons. And this looks much more promising!

Finally, we take advantage of the symmetry between p and q to conclude

(4) $P \wedge X.q < X.p$
$\Rightarrow$
$(\underline{A}h: p \leq h \wedge h < p+d+1: M \neq X.h)$.

This concludes our exploration of a lexicographic comparison between two sequences $X.p$ and $X.q$, $p \neq q$, be it that we have not yet addressed the remaining case $X.p = X.q$. We return to that in a moment.

* * *

The consequents of the lemmata (4) and (3) are – by their very shapes – calling for invariants of the form

Q_p: $(\underline{A}h: p_0 \leq h \wedge h < p: M \neq X.h)$ and

Q_q: $(\underline{A}h: q_0 \leq h \wedge h < q: M \neq X.h)$,

and the question that then remains is

how to choose p_0 and q_0 - which is our only freedom left - .

Because X is periodic - recall, $X.h = X.(h+N)$ - there is no restriction involved in choosing $p_0 = 0$. At the same time we adopt invariant

$Z: \quad p < q$

in order to ensure $p \neq q$. For deciding on q_0 we now turn our attention to the as yet undiscussed case $X.p = X.q$.

The analysis given so far has only yielded "non - M" sequences. In view of our ultimate goal to isolate a sequence M , the analysis of the case $X.p = X.q$ will (have to) be driven by the desire to find a sequence M. From $X.p = X.q$ we conclude that X has $q-p$ as a period. (This may be shown by showing that for any h , $X.p = X.q \Rightarrow X.(p+h) = X.(q+h)$, which follows in a straightforward manner from the connections (i) - (iii) and (ii) - (iv) , given before.) In conjunction with (0) - which has not been used yet - we then obtain

$$(\exists h: p \leq h \wedge h < q: M = X.h) \ .$$

Now we confront this with invariant Qq

$$(\forall h: q_0 \leq h \wedge h < q: M \neq X.h) \ .$$

and here we see that with the choice

$q_0 = p+1$ the conjunction of these two conditions implies $M = X.p$. Adopting that choice we therefore can rely on

(5) $$Qq \wedge X.p = X.q \Rightarrow M = X.p \quad .$$

* * *

This completes our analysis of the problem and our design of a hopefully suitable invariant for the repetitive construct. In summary, it runs

$$P: \quad 0 \leq d \wedge (\underline{A}i: 0 \leq i \wedge i < d: X.p.i = X.q.i)$$

$$Z: \quad p < q$$

$$Qp: \quad (\underline{A}h: 0 \leq h \wedge h < p: M \neq X.h)$$

$$Qq: \quad (\underline{A}h: p+1 \leq h \wedge h < q: M \neq X.h) \quad ,$$

with as a little accompanying theory the lemmata (3), (4), and (5). This and the givens (0) and (1) is, in fact, all we need to know for carrying out the forthcoming construction of a program text establishing postcondition $M = X.p$

* * *

After the above has been said and done, the construction of the program text offers no further surprises, except for the guard of the repetitive construct which we shall discuss in isolation. Here is the text, lavishly annotated at some crucial places.

```
p, q, d := 0, 1, 0
  {Inv :  P ∧ Z ∧ Qp ∧ Qq }
; do ... →
    if  X.p.d = X.q.d   →  d := d+1   { Inv }
    [] X.p.d < X.q.d   →
            {By P, we have X.p < X.q,
             and hence, with the aid of
             lemma (3),
             (Ah: q≤h ∧ h<q+d+1 : M ≠ X.h) .
             Conjoined with Qq this yields
             (Ah: p+1≤h ∧ h<q+d+1 : M ≠ X.h),
             so that }
        q, d := q+d+1, 0
            {maintains Qq, and of course
             the rest of Inv . }
    [] X.q.d < X.p.d   →
            { By P, we have X.q < X.p,
             and with the aid of lemma
             (4) and Qp,
             (Ah: 0≤h ∧ h<p+d+1 : M ≠ X.h).
             Conjoined with Qq - using
             0≤d - this yields
```

$(\underline{A}h: 0 \leq h \wedge h < (p+d+1) \underline{max}\ q : M \neq X.h)$,
so that }

$p, d := (p+d+1) \underline{max}\ q\ ,\ 0$

{maintains Qp .}

; $q := p+1$ {Inv}

fi

od .

What remains to be done is to construct a guard and to find a suitable variant function. As for the latter, choice $p+q+d$ is not too far-fetched, since we may observe that the first two alternatives each increase $p+q+d$ by exactly 1, and the third alternative by at least 1 (even by at least 2). So, this is okay. But is $p+q+d$ bounded from above?

From Qp and datum (0),

(0) $(\underline{E}h: 0 \leq h \wedge h < N : M = X.h)$,

we conclude that $p < N$ is an invariant of the repetition. This gives us an upperbound on p that is linear in N. We now could proceed finding or imposing linear upperbounds on q and on d, but we can do better in one fell swoop.

The invariance of $p<N$ implies that the precondition of statement

$p := (p+d+1) \max q$

in one way or another has to imply

$$(p+d+1) \max q < N \ .$$

or, equivalently,

(6) $\quad p+d+1 < N \ \wedge \ q < N \ .$

This precondition does not simply follow from the program text as is, unless (6) can be taken as a guard of the repetition. If we can take it as a guard, the precondition of the step implies

$$p + q + d \leq 2 \times N - 3 \ ,$$

and as a result – $p+q+d$ being equal to 1 initially – the number of steps of the repetitive construct will be at most $2 \times N - 3$. This, then, nicely complies with our efficiency requirements.

Next we shall show that (6) can be taken as a guard indeed, by demonstrating that its negation conjoined with the invariant implies $M = X.p$. We argue by case analysis.

Case $N \leq q$

$$\begin{array}{ll} & N \leq q \\ \Rightarrow & \{\text{using } Qp \text{ and } Qq\} \\ & (\underline{A}h: 0 \leq h \wedge h < p: M \neq X.h) \\ & \quad \wedge \ (\underline{A}h: p+1 \leq h \wedge h < N: M \neq X.h) \end{array}$$

$\Rightarrow$ {using (0)}
$M = X.p$.

Case $N \leq p+d+1$

Subcase $X.p \leq X.q$

$X.p < X.q$
$\Rightarrow$ {using P and (3)}
$(\underline{A}h: q \leq h \wedge h < q+d+1 : M \neq X.h)$
$\Rightarrow$ {using Z, i.e. $p<q$}
$(\underline{A}h: q \leq h \wedge h < p+d+1 : M \neq X.h)$
$\Rightarrow$ {using $N \leq p+d+1$}
$(\underline{A}h: q \leq h \wedge h < N : M \neq X.h)$
$\Rightarrow$ {conjoining with Qp and Qq}
$(\underline{A}h: 0 \leq h \wedge h < p : M \neq X.h)$
$\wedge$ $(\underline{A}h: p+1 \leq h \wedge h < N : M \neq X.h)$
$\Rightarrow$ {using (0)}
$M = X.p$.

Subcase $X.q < X.p$

$X.q < X.p$
$\Rightarrow$ {using P and (4)}
$(\underline{A}h: p \leq h \wedge h < p+d+1 : M \neq X.h)$
$\Rightarrow$ {conjoining with Qp}
$(\underline{A}h: 0 \leq h \wedge h < p+d+1 : M \neq X.h)$
$\Rightarrow$ {using $N \leq p+d+1$ and (0)}
false
$\Rightarrow$ {pred. calc.}
$M = X.p$

Subcase $X.p = X.q$

$$
\begin{array}{ll}
 & X.p = X.q \\
\Rightarrow & \quad \{ \text{using } \mathbb{Q}q \text{ and } (5) \} \\
 & M = X.p
\end{array}
$$

(End of Cases.)

So in any case we have $M = X.p$ as a postcondition of the repetition

We conclude this derivation by depicting the raw code, phrased in terms of the original cyclic array $b\,(i: 0 \leq i \wedge i < N)$.

```
p, q, d := 0, 1, 0
; do  p+d+1 < N  ∧  q < N
  →  if  b.(p+d) = b.(q+d)  →  d := d+1
     [] b.(p+d) < b.(q+d)
            →  q, d := q+d+1, 0
     [] b.(q+d) < b.(p+d)
            →  p, d := (p+d+1) max q, 0
               ; q := p+1
     fi
od
{M = X.p}
```

× × ×

Final Remarks

a) There are some technicalities concerning the above algorithm that the reader may wish to know, to check, to appreciate, or to ignore.

- From the guard $p+d+1 < N$ and the invariance of $0 \leq p$, it follows that $d+1 < N$ is a valid precondition of the step of the repetition. Hence, $d < N$ is an invariant as well. In the light of the invariance of P this means that no execution of the program will ever "find" $X.p = X.q$. If this sounds inconceivable, the conclusion must be that operational reasoning is impossible to comprehend.

- The repetitive construct can do with the much simpler and weaker guard

  $$d < N \wedge q < N ,$$

  but the price to be paid is a non-negligible loss of efficiency, viz. from at most $2*N-3$ to at most $3*N-4$ element comparisons. For us, the reason to "tolerate" the stronger guard is not only a matter of efficiency, but that its incorporation requires no extensive justification and that we can handle it in terms of the theory already developed.

- The latter remark is related to the experience that all sorts of efforts to improve on the program's efficiency were all attended by a serious complification of the argument. In this respect Yossi Shiloach has pushed the exercise to a limit — see his incomprehensible "Fast Canonization of Circular Strings" in the Journal of

Algorithms 2 (1901) - .

- The weaker guard

$$d < N \;\wedge\; q < N$$

is attractive however, if the algorithm is to deliver the (least) period of X as well, because this can then be achieved by postfixing the repetitive construct with

```
if N ≤ d → period := q - p
[] N ≤ q → period := N
fi .
```

b) Our main purpose in the foregoing presentation has been to get the design process across. In general, the corner stones of a design process are formed by a number of outstanding design decisions (which usually are prompted by heuristic considerations and by the desire to keep a design simple). It is hoped and believed that each time a design decision has been taken, a more or less exhaustive exploration of its consequences will lead to a point where the need for a next design decision presents itself. In our present example the corner stones are formed by

- the mere decision to attach the name M to the final answer
- the decision to focus on a lexicographic comparison between two sequences

- the adoption of the invariants Qp and Qq, but more in particular the choice of q_0.

When armed with such a limited number of corner stones, we should be able to reconstruct the design without much effort, not only today or tomorrow but even next year. (There is something completely wrong if a University Professor of Programming needs to bring notes to guide his lectures.) In this respect, I was pleasantly surprised when some time ago Richard Bird sent me a note containing the algorithm that we just developed. The surprise was that the development had been explained to him by Netty van Gasteren without the use of pencil and paper (during a rail journey). I am very grateful to them for having done that experiment.

c) I am also indebted to Edsger W. Dijkstra and Carel S. Scholten for a considerable embellishment, back in 1979, of a first draft describing the development of this program.

W.H.J. Feijen,

29 November 1990,

Sterksel.