A DISCIPLINE OF PROGRAMMING

EDSGER W. DIJKSTRA

Burroughs Research Fellow,
Professor Extraordinarius,
Technological University, Eindhoven

PRENTICE-HALL, INC.

ENGLEWOOD CLIFFS, N.J.

17 AN EXERCISE ATTRIBUTED TO R.W. HAMMING

The way the problem reached me was: "To generate in increasing order the sequence *1, 2, 3, 4, 5, 6, 8, 9, 10, 12*, ... of all numbers divisible by no primes other than *2, 3*, or *5*." Another way of stating which values are in the sequence is by means of three axioms:

Axiom *1*.	The value *1* is in the sequence.
Axiom *2*.	If x is in the sequence, so are $2 * x$, $3 * x$, and $5 * x$.
Axiom *3*.	The sequence contains no other values than those that belong to it on account of Axioms *1* and *2*.

(We leave to the number theorists the task of establishing the equivalence of the two above definitions.)

We include this exercise because its structure is quite typical for a large class of problems. Being interested only in terminating programs, we shall make a program generating only the, say, first *1000* values of the sequence. Let

$PO(n, q)$ mean: the value of "q" represents the ordered set of the first "n" values of the sequence.

Then Axiom *1* tells us that *1* is in the sequence and, as $2 * x$, $3 * x$, and $5 * x$ are functions whose value is $> x$ for $x > 0$, Axiom *2* tells us that *1* is the minimum value whose membership of the sequence can be established on account of the first two axioms. Axiom *3* then tells us that *1* is the minimum value occurring in the sequence and therefore $PO(n, q)$ is easily established for $n = 1$: "q" then contains the value *1* only. The obvious program structure is:

"establish $PO(n, q)$ for $n = 1$";
do $n \neq 1000 \longrightarrow$
    "increase n by 1 under invariance of $PO(n, q)$"
od

Under the assumption that we can extend a sequence with a value "*xnext*", provided that the value "*xnext*" is known, the main problem of "increase n by 1 under invariance of $PO(n, q)$" is how to determine the value "*xnext*". Because the value 1 is already in q, $xnext > 1$, and *xnext*'s membership of the sequence must therefore rely on Axiom *2*. Calling the maximum value occurring in q "*q.high*", *xnext* is the minimum value $> q.high$, that is, of the form $2 * x$ or $3 * x$ or $5 * x$ such that x occurs in the sequence. But because $2 * x$, $3 * x$, and $5 * x$ are all functions whose value is $> x$ for $x > 0$, that value of x must satisfy $x < xnext$; furthermore, x cannot satisfy $x > q.high$, for then we would have

$$q.high < x < xnext$$

which would contradict that *xnext* is the minimum value $> q.high$. Therefore we have $x \leq q.high$, i.e. x must already occur in q, and we can sharpen our definition of *xnext*: *xnext* is the minimum value $> q.high$, that is of the form $2 * x$ or $3 * x$ or $5 * x$, such that x occurs in q. (It is for the sake of the above analysis that we have initialized $PO(n, q)$ for $n = 1$; initialization for $n = 0$ would have been just as easy, but then *q.high* would not be defined.)

A straightforward implementation of the above analysis would lead to the introduction of the set *qq*, where *qq* consists of all values $xx > q.high$, such that xx can be written as

$$xx = 2 * x, \qquad \text{with } x \text{ in } q,$$

or as

$$xx = 3 * x, \qquad \text{with } x \text{ in } q,$$

or as

$$xx = 5 * x, \qquad \text{with } x \text{ in } q$$

The set *qq* is nonempty and *xnext* would be the minimum value occurring in it. But upon closer inspection, this is not too attractive, because the adjustment of *qq* would imply (in the notation of the previous chapter)

$$qq := (qq \simeq \{xnext\}) + \{2 * xnext, 3 * xnext, 5 * xnext\}$$

where the "+" means "forming the union of two sets". Because we have to determine the minimum value occurring in *qq*, it would be nice to have the elements of q ordered; forming the union in the above adjustment would then require an amount of reshuffling, which we would like to avoid.

A few moments of reflection, however, will suffice for the discovery that we do not need to keep track of the whole set *qq*, but can select *xnext* as the minimum value occurring in the much smaller set

$$qqq = \{x2\} + \{x3\} + \{x5\},$$

where

> $x2$ is the minimum value $> q.high$, such that $x2 = 2 * x$ and x occurs in q,
> $x3$ is the minimum value $> q.high$, such that $x3 = 3 * x$ and x occurs in q and
> $x5$ is the minimum value $> q.high$, such that $x5 = 5 * x$ and x occurs in q.

The above relation between q, $x2$, $x3$, and $x5$ is denoted by $P1(q, x2, x3, x5)$.

A next sketch for our program is therefore:

```
"establish P0(n, q) for n = 1";
do n ≠ 1000 →
    "establish P1(q, x2, x3, x5) for the current value of q";
    "increase n by 1 under invariance of P0(n, q), i.e.
        extend q with min(x2, x3, x5)"
od
```

A program along the above lines would be correct, but now "establish $P1(q, x2, x3, x5)$ for the current value of q" would be the nasty operation, even if —what we assume— the elements of the ordered set q are as accessible as we desire. The answer to this is a standard one: instead of computing $x2$, $x3$, and $x5$ as a function of q afresh when we need them, we realize that the value of q only changes "slowly" and try to "adjust" the values, which are a function of q, whenever q changes. This is such a standard technique that it is good to have a name for it; let us call it "taking the relation outside (the repetitive construct)". Its application is reflected in the program of the following structure:

```
"establish P0(n, q) for n = 1";
"establish P1(q, x2, x3, x5) for the current value of q";
do n ≠ 1000 →
    "increase n by 1 under invariance of P0(n, q), i.e.
        extend q with min(x2, x3, x5)";
    "re-establish P1(q, x2, x3, x5) for the new value of q"
od
```

The re-establishment of $P1(q, x2, x3, x5)$ has to take place after extension of q, i.e. after increase of $q.high$; as a result, the adjustment of $x2$, $x3$, and $x5$ is either the empty operation, or an increase, viz. a replacement by the corresponding multiple of a higher x from q. Representing the ordered set q by means of an array *aq*, i.e. as the values *aq*(*1*) through *aq*(*n*) in monotonically increasing order, we introduce three indices *i2*, *i3*, and *i5*, and extend *P1* with

... **and** $x2 = 2 * aq(i2)$ **and** $x3 = 3 * aq(i3)$ **and** $x5 = 5 * aq(i5)$

Our inner block, initializing the global array variable *aq* with the desired final value could be:

```
begin virvar aq; privar i2, i3, i5, x2, x3, x5;
    aq vir int array:= (1, 1); {P0 established}
    i2 vir int, i3 vir int, i5 vir int:= 1, 1, 1;
    x2 vir int, x3 vir int, x5 vir int:= 2, 3, 5; {P1 established}
    do aq.dom ≠ 1000 →
        if x3 ≥ x2 ≤ x5 → aq:hiext(x2)
        ▯ x2 ≥ x3 ≤ x5 → aq:hiext(x3)
        ▯ x2 ≥ x5 ≤ x3 → aq:hiext(x5)
        fi {aq.dom has been increased by 1 under invariance of P0};
        do x2 ≤ aq.high → i2:= i2 + 1; x2:= 2 * aq(i2) od;
        do x3 ≤ aq.high → i3:= i3 + 1; x3:= 3 * aq(i3) od;
        do x5 ≤ aq.high → i5:= i5 + 1; x5:= 5 * aq(i5) od
            {P1 has been re-established}
    od
end
```

In the above version it is clearly expressed that after re-establishing *P1* we have $x2 > aq.high$ **and** $x3 > aq.high$ **and** $x5 > aq.high$. Apart from that we could have used "$\ldots = aq.high$" instead of "$\ldots \leq aq.high$" as well.

Note 1. In the last three inner repetitive constructs each guarded statement list is selected for execution at most once. Therefore, we could have coded them

```
if x2 = aq.high → i2:= i2 + 1; x2:= 2 * aq(i2)
▯ x2 > aq.high → skip
fi; etc.
```

When I start to think about this choice, I come out with a marked preference for the repetitive constructs, for what is so particular about the fact that a repetition terminates after zero or one execution as to justify expression by syntactic means? Very little, I am afraid. Any hesitation to recognize "zero or one times" as a special instance of "at most *k* times" is probably due to our linguistic inheritance, as all Western languages distinguish between singular and plural forms. (If we had been classical Greeks (i.e. used to thinking in terms of a dual form as well) we might have felt obliged to introduce in addition special syntactical gear for expressing termination after at most two executions!) To end in "Updating a sequential file" with

do *xx.norm* → *newfile:hiext*(*xx*); *xx:setabnorm* **od**

instead of with

if *xx.norm* → *newfile:hiext*(*xx*) ▯ **non** *xx.norm* → *skip* **fi**

would, in a sense, have been more "honest", for the output obligation as expressed by *xx.norm* has been met. (*End of note 1.*)

Note 2. The last three inner repetitive constructs could have been combined into a single one:

```
do x2 ≤ aq.high → i2:= i2 + 1; x2:= 2 * aq(i2)
 ▯ x3 ≤ aq.high → i3:= i3 + 1; x3:= 3 * aq(i3)
 ▯ x5 ≤ aq.high → i5:= i5 + 1; x5:= 5 * aq(i5)
od
```

I prefer, however, not to do so, and not to combine the guarded commands into a single set when the execution of one guarded statement list cannot influence the truth of other guards from the set. The fact that the three repetitive constructs, separated by semicolons, now appear in an arbitrary order does not worry me: it is the usual form of over-specification that we always encounter in sequential programs prescribing things in succession that could take place concurrently. (*End of note 2.*)

The exercise solved in this chapter is a specific instance of a more general problem, viz. to generate the first N values of the sequence given axiomatically by

Axiom *1*. The value *1* is in the sequence.

Axiom *2*. If x is in the sequence, so are $f(x)$, $g(x)$, and $h(x)$, where f, g, and h are monotonically increasing functions with the property $f(x) > x$, $g(x) > x$, and $h(x) > x$.

Axiom *3*. The sequence contains no other values than those that belong to it on account of Axioms *1* and *2*.

Note that if nothing about the functions f, g, and h were given, the problem could not be solved!

EXERCISES

1. Solve the problem if Axiom *2* is replaced by:

Axiom *2*. If x is in the sequence, so are $f(x)$ and $g(x)$, where f and g have the property $f(x) > x$ and $g(x) > x$.

2. Solve the problem if Axiom *2* is replaced by:
Axiom *2.* If x and y are in the sequence, so is $f(x, y)$, where f has the properties

1. $f(x, y) > x$
2. $(y1 > y2) \Rightarrow (f(x, y1) > f(x, y2))$
(*End of exercises.*)

The inventive reader who has done the above exercises successfully can think of further variations himself.