The Linear Search, and some more

Those who are more or less familiar with the way in which the mathematical science of programming has evolved over the last two decades, may be somewhat amazed to find a presupposedly matured computing scientist readdressing a simple algorithm like the Linear Search. There are, however, a number of compelling reasons for doing so.

One of them is that, even after these two decades, an impressive lack of familiarity with the Linear Search and all that hovers around it, is still an important source of error, confusion, and complifi-cation for the every-day-programmer, whether we like it or not. (See, for instance, the discussion evoked by Frank Rubin in various 1987-issues of the CACM; it all centered around the Bounded Linear Search.)

Another reason is that these two decades have not yet produced a fully satisfactory development of the algorithm. The presentations that can be found - and there are many of them - range from simply stating the algorithm without much attention for development or even a proof of correctness, via those in which the correctness argument takes the form "it is fairly easy to see", to formal developments that for no good reason

rely on the Principle of Mathematical Induction. Therefore, we offer a next presentation (, of which we can be sure that it won't be the last one).

But the most important reason for us to revisit the Linear Search is that it provides us with a rather nice and simple example of program or proof development in which the problem statement leaves us barely any choice on how to proceed. I.e., at almost any state of the development, the shapes of the formulae and the design obligations to be met will strongly dictate the next step. Since this is a frequently recurring theme, it can best be illustrated by means of a simple example.

* * *

The Linear Search can be specified as follows. Given a boolean function $b.(i : 0 \leq i)$, defined on the naturals, we wish to establish

$$\|[\ \underline{\text{var}}\ x : \text{int} ;\ \{Q\}\ \text{Linear Search}\ \{R\}\ ]\|,$$

in which Q and R are given by

$$Q: \quad (\underline{\exists}\, i : 0 \leq i : b.i) ,$$

$$R: \quad R0 \wedge R1 \wedge R2 ,$$

R0: $0 \leq x$.
R1: $(\underline{A}i: 0 \leq i \wedge i < x: \neg b.i)$.
R2: $b.x$

Now, the first thing we do is invite the reader to completely ignore precondition Q, since our design of the algorithm will be entirely driven by the obligation to guarantee postcondition R. Only at the very end of our derivation emerges Q as the precise condition needed to guarantee termination of the constructed program.

× × ×

Postcondition R, considered as an equation in unknown x, does not admit of an analytical solution, because function b is too arbitrary. We therefore have to establish R by means of a repetitive construct.

A repetitive construct cannot come without an invariant – P, say – and a guard – B, say –. As always, we then have to ensure several things, to wit

- $P \wedge \neg B \Rightarrow R$
- P is established initially, i.e. as a precondition of the repetition
- P is maintained by a step of the repetition, and

- the whole construct terminates.

We will deal with these concerns in turn.

In order to guarantee that $P \wedge \neg B \Rightarrow R$. we propose to divide the three conjuncts of R over P and $\neg B$.
The question of which conjuncts to include in P and which ones in $\neg B$ is answered by investigating the possibilities for initialization.

For establishing conjunct $R0$ we have a wealth of values for x at our disposal. As for $R1$, there is only one realistic possibility for x, viz. 0. For $R2$ there is none at all. The latter implies that $R2$ cannot be part of the invariant and must go into the guard. The guard thus will contain expression $\neg b.x$ and in order to guarantee that it is defined, we had better ensure the invariance of $0 \leq x$. Hence, $R0$ must be part of the invariant. Finally, $R1$ is too complicated to be admitted in the guard. As a result, our program has to be of the form

$x := 0$
$\{ \text{inv. } R0 \wedge R1 \}$
; **do** $\neg R2 \rightarrow \cdots$ **od**
$\{ R0 \wedge R1 \wedge R2, \text{ i.e. } R \}$.

Next, we investigate the step $\cdots$ of

the repetition, which has to be designed so that it maintain the invariant. Of course, skip would be okay, but with that choice we foresee problems in giving a termination argument (which is a task still ahead of us). Therefore, we had better see to it that the step effectively modifies some of the program variables. But there is only one such variable, viz. x. So we are forced to modify x. Moreover, if we are to ensure the invariance of $R0$ $-$ $0 \leq x$ $-$ then we have only little choice beyond a step of the form: $x := x +$ "something positive".

Now we turn our attention to the required invariance of $R1$. The precondition of the step is $R0 \wedge R1 \wedge \neg R2$, i.e.

$$0 \leq x \;\wedge\; (\underline{A}i: 0 \leq i \wedge i < x : \neg b.i) \;\wedge\; \neg b.x \;,$$

which implies

$$(\underline{A}i: 0 \leq i \wedge i < x+1 : \neg b.i) \;.$$

The postcondition is $R1$, and the step's functional specification

$$\begin{array}{l} \{(\underline{A}i: 0 \leq i \wedge i < x+1 : \neg b.i)\} \\ \quad x := x + \text{"something positive"} \\ \{(\underline{A}i: 0 \leq i \wedge i < x : \neg b.i)\} \;. \end{array}$$

Thus, by the axiom of assignment, $x := x+1$ is forced upon us.

As a result, we obtain for the Linear Search

$$x := 0 \quad \{\text{inv. } R0 \wedge R1\}$$
$$; \ \underline{do} \ \neg b.x \ \rightarrow \ x := x+1 \ \underline{od}$$
$$\{R\},$$

a design which has been completely dictated by the shape of R .

* * *

How to prove termination? The Theorem of Invariance tells us that we have to design a variant function, i.e. an integer-valued function that is increasing and bounded from above (or decreasing and bounded from below). In the above solution for the Linear Search, integer x is the only variable involved and moreover it is increasing. So we have proved termination provided we can guarantee the existence of an X such that $x \leq X$ follows from the conjunction of the invariant and the guard of the repetitive construct. Well, neither guard $\neg b.x$ nor part $0 \leq x$ - R0 - of the invariant can reveal an upperbound for x. Hence, the conclusion $x \leq X$ must follow from $R1$, i.e. from

$$(\underline{A}i: 0 \leq i \wedge i < x : \neg b.i) \ ,$$

if possible at all. Fortunately, $R1$ is helpful, since it can be rewritten as

$$(\underline{A}i: 0 \leq i \wedge b.i : x \leq i) \quad .$$

Now the existence of an X such that $x \leq X$ holds can only follow if we can guarantee the existence of an X for which

$$0 \leq X \wedge b.X \quad ;$$

and this is precisely the rôle fulfilled by precondition Q, which has been ignored so far.

* * *

This completes the derivation of the Linear Search. Finally, we wish to draw the attention to the intimate relationship between the Principle of Mathematical Induction and the Theorem of Invariance (for repetitive constructs). For this to become clear we should we willing to appreciate a terminating program as a proof that the program's postcondition, regarded as an equation in its free variables, has a solution. If we adopt this view — and we do — , the Linear Search as developed above is a proof of the theorem

$$Q \Rightarrow (\underline{E}x :: R) ,$$

for any b. Now, we massage the theorem a bit, leaving the ranges of the dummies understood (as naturals):

$$
\begin{array}{ll}
 & Q \Rightarrow (\underline{E}x :: R) \\
= & \quad \{ \text{definitions of } Q \text{ and } R \} \\
 & (\underline{E}i :: b.i) \\
 & \Rightarrow (\underline{E}x :: b.x \wedge (\underline{A}i : i<x : \neg b.i)) \\
= & \quad \{ \text{contra-positive and de Morgan} \} \\
 & (\underline{A}i :: \neg b.i) \\
 & \Leftarrow (\underline{A}x :: \neg b.x \vee \neg (\underline{A}i : i<x : \neg b.i)) \\
= & \quad \{ \text{renaming } \neg b \text{ as } c \} \{ \text{pred. calc.} \} \\
 & (\underline{A}i :: c.i) \\
 & \Leftarrow (\underline{A}x :: c.x \Leftarrow (\underline{A}i : i<x : c.i)) \ .
\end{array}
$$

The last line of this calculation is known as the Principle of Mathematical Induction for the naturals. Since in our derivation we have not used anything else than standard predicate calculus and the Theorem of Invariance, we conclude that the Principle follows from the Theorem. That the converse holds as well, has been discussed extensively at many places in the literature. Thus the two are just two faces of the same coin. Of course, all this is known, but sometimes it may help to be actively aware of it. It offers the opportunity to design a program when a proof by mathematical induction is due. This may have the advantage that we can come away with a more sober nomenclature, sufficient for discussing one step of the induction, so to speak. Moreover, the variant function is not so oppressively suggestive of the

miserable "step - by - one" induction that has been presented to many of us as the Principle of Mathematical Induction.

Eindhoven. April 1990

W.H.J. Feijen,
Department of Mathematics and
Computing Science.
Eindhoven University of Technology.
Den Dolech 2,
5600 MB Eindhoven,
The Netherlands