A detailed derivation of a very simple program

We are going to derive a program "test all true", satisfying the following specification:

\[
\text{l[con N: int; con f: int \rightarrow bool; var x: bool; test all true \{R: x \equiv (\forall k: 0 \leq k \land k < N: f(k))\]]}
\]

* * *

Our first step is to rewrite the right-hand side of \( R \) as a function (whose properties we'll subsequently exploit) of some argument(s). Since functions of integers are simpler than functions of functions, we consider the function \( f \) as a constant, but do consider the right-hand side of \( R \) a function of the two bounds of the range, that is, we rewrite the post-condition

\[
R: \quad x \equiv H_{\text{O.N}}
\]

where \( H \) is given by

\[
(0) \quad H_{i,j} \equiv (\forall k: i \leq k \land k < j: f(k))
\]

* * *

Our next step is -unavoidably!- the investigation of the (hopefully exploitable) properties of the function \( H \), just introduced. Standard application of predicate calculus (and nothing else but lemmata like
\[ k < j \lor k = j \equiv k < j+1 \]

yields the following three properties of \( H \)

1. \( i > j \Rightarrow H \cdot i \cdot j \)
2. \( i < j \Rightarrow (H \cdot (i+1) \cdot j \land \exists i \equiv H \cdot i \cdot j) \)
3. \( i < j \Rightarrow (H \cdot i \cdot (j-1) \land \exists (j-1) \equiv H \cdot i \cdot j) \)

In (1) we recognize "bases" — (1) holds for all \( i, j \) — in (2) and (3) "steps".

* * *

Heading for a repetition, our third step is unavoidably! — the choice of an invariant that can be (i) initialized, (ii) maintained, and (iii) used to conclude \( R \). Postponing our concerns regarding (ii), we observe that — thanks to Leibniz's Principle — we can initialize

\[ H \cdot i \cdot j \equiv H \cdot 0 \cdot N \]

— viz. by \( i, j := 0, N \) — but that the left-hand side has to be generalized so as to contain \( x \) in such a fashion that we can conclude \( R \). We choose for invariant \( P_0 \land P_1 \) with

\[ P_0: \quad H \cdot i \cdot j \land x \equiv H \cdot 0 \cdot N \]

\[ P_1: \quad 0 \leq i \land j \leq N \]

Ad \( P_0 \) we remark that the alternative left-hand
sides $H_{ij} \lor x$ and $H_{ij} \equiv x$ would have
given invariants as easily initialized as $P_0$.
But in order to conclude $R$, i.e.

$$x \equiv H_0 N$$

from

$$(H_{ij} \lor x) \equiv H_0 N$$,

the known value of $H_{ij}$ should be the
unit element of $\lor$. Since (1) tells us
that the known value of $H_{ij}$ is true,
this rules out the disjunction. In view of
(2) and (3), the equivalence seems to
lead to an invariant hard to maintain,
and hence the conjunction in $P_0$.

* * *

Our fourth step is to determine under
what additional condition the invariant
implies $R$. To this end we observe

$$R = \{\text{definition of } R\}$$

$$x \equiv H_0 N$$

$$= \{P_0\}$$

$$x \equiv x \land H_{ij}$$

$$= \{\text{pred. calc.}\}$$

$$\neg x \lor H_{ij}$$

$$\iff \{\text{(1)}\}$$

$$\neg x \lor i \geq j$$,

after which observation we decide to use
the negation of the last expression, i.e.

\[ (4) \quad x \land i < j \]

as guard.

\[ * \quad * \quad * \]

Our fifth step is to derive the command(s) guarded in the repetition. In order to use (2) we consider a guarded command that comprises \( i' := i + 1 \) and leaves \( j \) constant. (Note that \( i' := i + 1 \) and \( j := j - 1 \) work towards falsification of \( i < j \), and that is nice for termination.) We consider for as yet unknown \( E \) the requirement that

\[ x \land i < j \rightarrow i, x := i + 1, E \]

leave \( P_0 \) invariant. We observe

\[
\begin{align*}
\text{wp. } (i, x := i + 1, E). P_0 &= \{ \text{def. } := ; \text{ def. } P_0 \} \\
&= \{ \text{def. } := ; \text{ def. } P_0 \} \\
&= \{ P_0, \text{ ex hypothesis} \} \\
&= \{ P_0, \text{ ex hypothesis} \} \\
&= \{ x \text{ from the guard} \} \\
&= \{ i < j \text{ from the guard, (2)} \} \\
&= \{ i < j \text{ from the guard, (2)} \} \\
E &\equiv \mathcal{P}. i
\end{align*}
\]
i.e. the guarded command

\[ x \land i < j \rightarrow i, x := i + 1, f.c \]

leaves \( P_0 \) invariant. Similarly we derive from the requirement that \( P_0 \) is maintained that the guarded command

\[ x \land i < j \rightarrow j, x := j - 1, f.(j - 1) \]

meets the requirement. Hence a solution is

\[
\begin{align*}
\text{test all true:} & \\
\text{[\text{var i; j: int; i, j := 0, N \{inv: P_0 \land P_1; bnd j - i\}}]} & \\
\text{do} & \ x \land i < j \rightarrow i, x := i + 1, f.c & \\
\text{od} & \ x \land i < j \rightarrow j, x := j - 1, f.(j - 1) & \\
\text{end } & \{P_0 \land (\forall x \land i \geq j), \text{hence}\} & \\
\text{end } & \{R\} & \\
\end{align*}
\]

Remark In the context of this note it is not interesting that in the above program the nondeterminacy can be reduced (by removal of one of the guarded commands). Nondeterminacy can always be reduced. (End of Remark.)

* * * *

Cognoscenti recognize an Eindhoven influence, in particular of Anne Kaldewijj, who did much to establish the superiority of the tail invariant, and of Wim H.J. Feijen, who taught
me (in WF155) how to derive the guard $x > i < j$
"from outside in", that is by confronting the
postcondition with the invariant. These influences
are gratefully acknowledged.

This note has been written for the sake of
the fifth step, in which the repeatable
statement is derived. (I don't remember
ever having done it that way.)

The way in which nondeterminacy entered
this program was a surprise; it was a
pleasure to see that it almost dictated
a tail invariant.

Austin, 18 October 1993

PS The weakest solution of the equation

$$X: [X \land Y \Rightarrow Z]$$

is $Y \Rightarrow Z$. It helps to know this. (End of PS.)

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