On the ascendingness-test and on tail invariants

One of our standard programming exercises for first-year students is the "ascendingness-test": given an integer array $f[0..N]$, $0 \leq N$, the question is to derive a programming establishing postcondition $R$: $x = "f[0..N] is ascending"$.

The solution that will emerge critically depends on what formalization is taken for $f$'s ascendingness. One—meanwhile classical—possibility is this:

$$(\forall i,j : 0 \leq i < j < N : f.i \leq f.j)$$

Based on this formulation the program is readily derived, in an absolutely standard way, by adopting as an invariant $R(N:=n)$. The resulting derivation is, however, a little bit laborious; also one may feel a little bit annoyed by the circumstance that—almost inevitably—the need arises to strengthen the invariant with the somewhat outlandish

$$y = (\max i : 0 \leq i < n : f.i)$$

But for the rest, a derivation along the above lines runs very smooth and does not evoke case analysis.

How different is the situation if one chooses to formalize $f$'s ascendingness by the very traditional
Now let us investigate our standard invariant for such problems, viz \( P_0 \land P_1 \) given by

\[
\begin{align*}
\forall i: 1 \leq i \leq N : f(i-1) &\leq f(i) \\
P_0: &\quad 0 \leq n \leq N \\
P_1: &\quad x \equiv (\forall i: 1 \leq i < n : f(i-1) \leq f.i ) \\
\end{align*}
\]

As always, we seek to increment \( n \) by 1 and carry out the corresponding calculation:

\[
\begin{align*}
(\forall i: 1 \leq i < n + 1 : f(i-1) &\leq f.i ) \\
&= \quad \{ \text{splitting off the term with } i = n \} \\
\end{align*}
\]

However, this splitting off is only possible whenever \( n \) is in the range of the quantification, in particular whenever \( 1 \leq n \).

And here we are in trouble, because the needed condition \( 1 \leq n \) cannot be expelled from any reasonably chosen guard nor from \( P_0 \).

The way out seems to replace \( P_0 \) with the stronger

\[
P_0': \quad 1 \leq n \leq N 
\]

but now we are in trouble again because the initial value of \( N \) could be 0. Thus a case-distinction between \( 0 = N \) and \( 1 \leq N \) is threatening.

The way out seems to replace \( P_0' \) with

\[
P_0'': \quad 1 \leq n 
\]

and to assign \( n < N \) as a conjunct to the guard. But then the demonstration that \( P_1 \land n \geq N \Rightarrow R \) becomes inconvenient because we will have to come up with benevolent thought values for \( f(N \ldots \infty) \).
In short: trouble all around.

We might suspect that the chosen formalization of \( f \)'s ascendingness is the culprit, but this is not the case at all as is witnessed by the following derivation.

We define \( K.n \) for \( 1 \leq n \) by

\[
K.n \equiv (\forall i : n \leq i < N : f.(i-1) \leq f.i)
\]

As invariant we choose \( Q_0 \land Q_1 \) given by

\[
\begin{align*}
Q_0 & : 1 \leq n \\
Q_1 & : K.1 = x \land K.n
\end{align*}
\]

The postcondition is

\[
R : K.1 = x
\]

The repetition can terminate whenever

\[
RHS.R = RHS.Q_1:
\]

\[
\begin{align*}
x &= x \land K.n \\
&= \{ \text{pred. calc.} \} \\
&\quad \land x \lor K.n \\
&\equiv \{ \text{def. of } K \} \\
&\quad \land x \lor n \geq N,
\end{align*}
\]

and therefore the negation of the last line is an acceptable guard. Thus the macroscopic structure of our program becomes

\[
\begin{align*}
x, n := \text{true, 1} \\
&\{ \text{Inv } Q_0 \land Q_1 \} \{ \text{Bnd } N-n \} \\
; \ \text{do } x \land n < N \rightarrow \cdots \text{ od} \\
&\{ R \}
\end{align*}
\]
The repeatable statement will contain \( n := n + 1 \), for termination’s sake. The required adjustment of \( x \) follows from

\[
\text{RHS. } Q_1
= \{ \}
x \land K.n
= \{ x = \text{true} \quad \text{from the guard} \quad \}
K.n
= \{ n < N \quad \text{from the guard} \quad
\}
l \leq n \quad \text{from } Q_0 \quad \}
f.(n-1) \leq f.n \land K.(n+1)
= \{ \text{substitution} \}
(x \land K.n) \left( x, n := f.(n-1) \leq f.n , n + 1 \right),
\]

and the resulting program is

\[
x, n := \text{true}, 1
; \quad \text{do } x \land n < N \rightarrow x, n := f.(n-1) \leq f.n , n + 1 \quad \text{od}
\]

* * *

The above program can also be developed on the basis of the invariance of \( P_0 \land P_1 \), the difference being that the derivation of the guard and the proof that \( R \) is established upon termination is much more cumbersome than with choice \( Q_0 \land Q_1 \). This note has been written to recall that tail invariants are just nicer, but as yet I don’t have a satisfactory technical explanation of the phenomenon.

Storksender, 5 November 1992,

W.H.J. Feijen