No operational insight required !

The following problem occurs in the book "Verification of sequential and concurrent programs" by Kreysztof R Apt and Ernst-Rüdiger Olderog. Given the two-component multiprogram

Pre: true

A: $x := x + 2$ B: $x := 0$,

prove that

R: $x = 0 \vee x = 2$

is a correct postcondition.
(The two components are really as simple as possible: they each consist of just one assignment.)

Apt & Olderog use the above example to illustrate the need to introduce auxiliary variables. Furthermore they remark that "the introduction of the auxiliary variable *done* required some insight into the execution of the given program", and that "the correctness proof is more complicated than expected".

Here is a proof for which no operational insight is required, but just predicate calculus and the proof rules of Owicki and Gries. We shall construct the proof in

very small steps and include most of the heuristics.

For a terminating multiprogram the postcondition should be implied by the conjunction of the postconditions of the individual components. So now we have to think about conditions RA and RB such that

(i) $x := x+2 \{RA\}$ and $x := 0 \{RB\}$

(ii) $RA \wedge RB \Rightarrow x=0 \vee x=2$

The simplest possible choice is $x=0 \vee x=2$ for both RA and RB; then (ii) is met. But that choice is too naive:

- it is too naive for RA because component A cannot establish $x=0 \vee x=2$ in isolation
- it is too naive for RB because component A can falsify it.

Therefore, we propose weaker conditions for RA and RB, viz.

RA: $x=0 \vee x=2 \vee q$

RB: $x=0 \vee x=2 \vee p$.

Then (ii) is met if we require that the postcondition also implies $\neg p \vee \neg q$:

(iii) $\neg p \vee \neg q$ is valid postcondition.

Next we focus on $x := x+2 \{RA\}$:

- RA gets locally correct by prefixing $x := x+2$ with assertion $RA(x := x+2)$, i.e.

  $x = -2 \;\vee\; x = 0 \;\vee\; q$

- RA is truthified by $x := 0$. In view of our obligation to satisfy (icl), we have a splendid opportunity here to embellish $x := 0$ with $q := false$.

In summary we have:

A: $\{x = -2 \;\vee\; x = 0 \;\vee\; q\,,\; ?\}$

$x := x+2$

$\{x = 0 \;\vee\; x = 2 \;\vee\; q\,,\; \heartsuit\}$

B: $x, q := 0, false$

$\{x = 0 \;\vee\; x = 2 \;\vee\; p\,,\; ?\}$

A's first assertion is globally correct. It is locally correct by adopting q as a precondition of the multiprogram.

The assertion in B is locally correct. It gets globally correct by embellishing $x := x+2$ in A with $p := true$. (Variable p is completely fresh, so that all previous proofs remain valid).

In summary, we arrived at the following fully and correctly annotated program text yielding the desired result.

Pre: q

A: $\{x = -2 \vee x = 0 \vee q\}$
$x, p := x+2, \text{true}$
$\{x = 0 \vee x = 2 \vee q\}$

B: $x, q := 0, \text{false}$
$\{x = 0 \vee x = 2 \vee p\}\{\neg q\}$

Post: $x = 0 \vee x = 2$

* * *

There is no reason for blaming Apt or Olderog. In a way, and given their goals, they have written a lovely book. The moral of the story is that operational reasoning, even if it is mimicked by some sort of formal logic, is almost always defeating. The technical mistake made by A & O is that they introduced just one auxiliary variable — *done!* — thereby reducing their manipulative freedom.

Meanwhile, I have become utterly convinced that in coming to grips with multiprograms, the proof rules should guide the way, and nothing else. And in case these proof rules are ala Owicki and Gries, an extremely well-versedness of the predicate calculus is an indispensable prerequisite.

WHJ Feijen
31 March 1995