4 MAR 1983

Dear Cliff,

I have been trying to prove a simple correctness assertion in terms of your notion of interference and I haven't yet succeeded. I wonder if you have the time to look at it. Indeed, do you think that it can be done? The important restriction is not to introduce fictitious variables or token positions into the predicates. The program is a simplified version of a Producer/Consumer example found in the paper "Proving the correctness of Multiprocess Programs" by Leslie Lamport, in IEEE, Trans. on Software engineering Vol 3 (1977) p 125.

The correctness assertion is

$\{pre\ x=t=0\}$ prod // cons $\{guar.\ 0 \le x \le b\}$

where $b>0$ is constant and

prod is

(*) ↓ $\underline{do}\ t<b \to x:=x+1;\ t:=t+1\ \underline{od}$

and cons is

$\underline{do}\ 0<t \to x:=x-1;\ t:=t-1\ \underline{od}$.

My idea has been to use the following special version of your proof rule.

$$\frac{\{pre\ P,\ rely\ G_2\}\ S_1\ \{guar\ G_1\} \qquad \{pre\ P,\ rely\ G_1\}\ S_2\ \{guar\ G_2\}}{\{pre\ P\}\ S_1 \parallel S_2\ \{guar\ G\}}$$

provided that $G_1 \vee G_2 \Rightarrow G$.

Using this rule it suffices to find predicates G_P and G_C such that the following can be proved

(1) $\{pre\ x=t=0,\ rely\ G_C\}\ prod\ \{guar\ G_P\}$

(2) $\{pre\ x=t=0,\ rely\ G_P\}\ cons\ \{guar\ G_C\}$

(3) $G_P \vee G_C \Rightarrow 0 \le x \le b$

(*) $\underline{do}\ B \to S\ \underline{od}$ should be taken to be blocked, not terminated, in a state where B is false.

On some reflection I have decided that the following definitions ought to work.

$$G_P \equiv (0 \le x = \overleftarrow{x} = t = \overleftarrow{t} \le b)$$
$$\vee (0 \le x-1 = \overleftarrow{x} \le \overleftarrow{t} = t < b)$$
$$\vee (0 \le x = \overleftarrow{x} \le \overleftarrow{t} = t-1 < b)$$

$$G_C \equiv (0 \le x = \overleftarrow{x} = t = \overleftarrow{t} \le b)$$
$$\vee (0 < t = \overleftarrow{t} \le \overleftarrow{x} = x+1 \le b)$$
$$\vee (0 < t+1 = \overleftarrow{t} \le \overleftarrow{x} = x \le b)$$

Certainly (3) is clear. Also, a little thought has convinced me that (1) and (2) are true. But I would like to prove them true using proof rules involving rely and guarantee predicates. Can this be done without using the position tokens that Lamport uses? How, in your approach, should (1) and (2) be proved?

The following proof rule seems a plausible tool

$$\frac{\{\text{pre } P' \,\&\, B, \text{rely } R\}\ S\ \{\text{guar } G, \text{post } P'\}}{\{\text{pre } P, \text{rely } R\}\ \underline{\text{do}}\ B \to S\ \underline{\text{od}}\ \{\text{guar } G\}}$$

provided that $P \Rightarrow P'$ and $R^+ \,\&\, \overleftarrow{P'} \,\&\, B \Rightarrow P'$, where R^+ is the transitive closure of R. why

But I do not see how to apply the rule to prove (1) and (2).

Thanks for your note. I would welcome having a discussion about data types with you. My impression is that the notion of data type is a major problem for computing theory. I don't think that it is simply a question of giving a definition within standard mathematics. Rather a whole theory is needed which uses a well-articulated and precise framework of terminology. Conventional mathematical terminology is not sufficiently refined to carry the needed distinctions.

As for the term "predicate", I have looked up Kleene's book and you are

correct about his usage. I then checked other logic books and it appears that there is no general agreement. My own approach is to treat ~~Boolean~~ predicates as expressions in connection with program verification. This is the approach of some logic books. See, for example Wilfred Hodges  pelican book on Logic.
I then briefly checked the computing literature. It appears that Hoare, Dijkstra etc... tend to use the terms "assertion" and "predicate" interchangeably. What they mean by these terms does not appear fixed to me. Sometimes they mean an expression and sometimes they mean what the expression ~~means~~ expresses and the latter is sometimes taken to be a function from states to truth values and sometimes a relationship between the values of the variables occurring in the expression. David Gries in his recent book takes predicates to be expressions. In my view, in the expression $\{P\}S\{Q\}$ the items P, S, Q

ought to be treated on the same level, i.e. all syntactic or all semantic. Surely program statements S are syntactic. Though so must P and Q. So after my literary investigations I am still inclined to take predicates to be expressions - even though there seems to be no general agreement. An alternative might be to use 'predicate' as you do and keep 'assertion' for the expression — but 'assertion' suggests that an act (of assertion) is involved, and we may wish to consider say the expression 'x < y' without any intention of it ever being asserted. No other term occurs to me. 'Predicate expression' is too unwieldy.

Yours Peter