comments by Jones
probably when he was at Oxford.

PLO4 ?
CAAH

Hoare's formalisation of Jones' calculus for programming

C B Jones identifies a program with a pair of predicates $(P(x), R(x,x\checkmark))$, where

- x is a list of all non-local variables of the program;
- $x\checkmark$ is a list of ticked variables, standing for the final values of the variables in x;
- $P(x)$ is the precondition, describing the initial values of the variables x, which ensure that the program will terminate;
- $R(x,x\checkmark)$ will be true of the initial and final values

If x does not satisfy $P(x)$, the program will not terminate, and the truth or falsity of $R(x,x\checkmark)$ is irrelevant; for the sake of uniqueness, we stipulate that in such cases it is always true:

$$\forall x, x\checkmark (\tilde{P}(x) \vee R(x,x\checkmark)) \qquad \text{(A)}$$

We then stipulate that the domain of R is total:

$$\forall x\, \exists x\checkmark\, R(x,x\checkmark) \qquad \forall x.\, P(x) \Rightarrow \exists x\checkmark.\, R(x,x\checkmark) \qquad \text{(B)}$$

Finally, the set of possible final values $x\checkmark$ is finite for each x satisfying $P(x)$:

$$\forall x\ (P(x) \Rightarrow (\{x\checkmark | R(x,x\checkmark)\} \text{ is finite})). \qquad \text{(C)}$$

not nec!

Now we can define our language. For each definition, we need to prove that the right hand side satisfies conditions ~~(1)~~ A, ~~(2)~~ B and ~~(3)~~ C, (provided that all programs mentioned on the left hand side do so).

(1) abort $\triangleq$ (false, true)

(2) skip $\triangleq$ (true, $x\checkmark = x$)

(3) $x_i := e \triangleq (De,\ x_0\checkmark = x_0 \wedge \ldots \wedge x_i\checkmark = e \wedge \ldots \wedge x_n\checkmark = x_n)$

where De is a predicate which is true of x just when the values of x make e defined.

(4) $(P,R) \vee (Q,S) \triangleq (P \wedge Q,\ R \vee S)$ non-deterministic union

(5) $(P,R) <b> (Q,S) \triangleq (Db \wedge P <b> Q,\ R <b> S)$ conditional

where $P <e> Q \triangleq$

$$(P, R) \equiv (P, P \wedge R) \not\equiv$$
$$\equiv (P, \sim P \vee R)$$

(6) $(P(x),\ R(x,x^{\checkmark}));(Q(x),\ S(x,x^{\checkmark}))$

$$(P(x) \wedge (\forall \dot{x}\ R(x,\dot{x}) \Rightarrow Q(\dot{x})),\ \exists \dot{x}\ R(x,\dot{x}) \wedge S(\dot{x},x^{\checkmark}))$$

(7) $\mu X.\ F(X) = (\exists n\ P_n,\ \forall n.\ R_n)$ recursion

where $(P_o,\ R_o)$ = abort

$$(P_{n+1},\ R_{n+1}) = F\ (P_n,\ R_n)$$

(8) (P,R) <u>sat</u> $(Q,S) \triangleq \vdash (Q \Rightarrow P) \wedge (Q \wedge R \Rightarrow S)$

Some of these laws are a bit complicated. Let us try to simplify them by a coding trick.

Let st be a fresh variable, not among x, (and never ~~explicitly~~ mentioned in the program).

Let $st^{\checkmark}$ be its dashed variant.

Let y be the list st, x_o, ..., x_n.

Let $y^{\checkmark}$ be the list $st^{\checkmark}$, $x_o^{\checkmark}$, ..., $x_n^{\checkmark}$.

Let $P(x)$ and $R(x,x^{\checkmark})$ satisfy conditions (A), (B), (C).

We now identify a program

$(P(x),\ R(x,x^{\checkmark}))$

with the <u>single</u> predicate

$G(y,y^{\checkmark}) \triangleq ((st \wedge P(x)) \Rightarrow st^{\checkmark} \wedge R(x,x^{\checkmark}))$, $(st \wedge P \Rightarrow st^{\checkmark}) \wedge R$

Similarly, we define

$H(y,y^{\checkmark}) \triangleq ((st \wedge Q(x)) \Rightarrow st^{\checkmark} \wedge S(x,x^{\checkmark}))$

Given a G of the above form, we can extract the original P and R as follows:

$$\left.\begin{array}{l} P = \sim(G[\text{false}/st^{\checkmark},\ \text{true}/st]) \\ R = G[\text{true}/st^{\checkmark},\ \text{true}/st] \end{array}\right\} \begin{array}{l}\text{where } [k/x] \text{ means} \\ \text{substitute } k \text{ for } x\end{array}$$

P ⇒

Now we can transform the earlier definitions as follows:

(1) abort $\triangleq$ true   any subst gives (t, t)

(2) skip $\triangleq$ $(st \Rightarrow (st^{\checkmark} \wedge x^{\checkmark} = x))$

(3) $x_i := e \triangleq (De \Rightarrow \text{skip}\ [e/x_i])$

(4) $G \vee H \triangleq G \vee H$

(5) $G <b> H \triangleq (Db \Rightarrow G <b> H)$

(6) $G\ (y, y^{\checkmark});\ H\ (y, y^{\checkmark}) \triangleq \exists \dot{y}\ G(y, \dot{y}) \wedge H(\dot{y}, y^{\checkmark})$

(7) $\mu X . F(X) \triangleq \forall n\ F^{n}\ (\text{true})$

(8) $G\ \underline{\text{sat}}\ H \triangleq\ \vdash\ G \Rightarrow H.$

If my calculations (of some time ago) are correct, the two formulations of the eight laws are isomorphic. For practical use, the predicate pairs may be more convenient. But for proof of algebraic properties, the single-predicate formulation seems simpler. Also, the single predicate generalises more easily to communicating processes. ✓

If the details can be corrected, this is very nice!