don't type

Oege de Moor
Clare Martin
Paul Gardiner
Carroll Morgan.

4/2

I've just seen that Oege's construction arises naturally from consideration of preconditions and post-conditions in Hoare logic. That these are the same as monotonic predicate transformers in Clare's formulation is an important result, maybe rather surprising.

I hope it is pleasant to you, even if no surprise.

Tony

In "Hoare logic", correctness of a program $Q$ is expressed in terms of a triple

$$P\{Q\}R$$

where $P$ is a predicate describing initial values of the program variables and $R$ describes the final values.

The pair $(P, R)$ serves as a specification of $Q$, i.e., an abstract description of its behaviour.

One specification $(P', R')$ is tighter $(\preccurlyeq)$ than another $(P, R)$ if it is harder to meet, for example because it has a weaker precondition and a stronger postcondition

$$(P', R') \preccurlyeq (P, R) \quad \text{if} \quad P \Rightarrow P' \wedge R' \Rightarrow R.$$

This is justified by the law of consequence

$$(P \Rightarrow P' \wedge P'\{Q\}R' \wedge R' \Rightarrow R) \vdash P\{Q\}R$$

so everything that satisfies a tighter spec also satisfies a looser one.

In all practical cases, P and R need to relate the current values of (program variables $v, w, \ldots$ to arbitrary values of certain logical variables, often denoted $v_0, w_0, \ldots$, which are not accessible to the program, and are assumed to denote the same value when they occur in P and R. For example P often includes a "snapshot" assertion $v = v_0 \wedge w = w_0$, in which case, occurrences of $v_0, w_0$ in R denote the initial values. But sometimes it is more convenient to snapshot the final values. But most convenient of all is to allow the logic variables to stand just for an arbitrary abstract values in some set E. So P and R are nothing but <u>relations</u> between E and the space of program variables, and (P, R) is nothing but our familiar span:

$$(P : E \to V, \ R : E \to V)$$

In general, the result space will be different from the source, but we won't bother with that here.

Clearly, the choice of a particular abstract space $E$ is not essential, and a different choice (say $E'$) could be used to formulate the same specification, i.e one satisfied by all the same programs. Let $S : E' \to E$ be a relation between abstract spaces. Then $(S;P , S;R)$ is easier to meet than $(P, R)$, or formally

$$\forall Q \quad P\{Q\}R \Rightarrow (S;P)\{Q\}(S;R)$$

proof $P\{Q\}R$ (assumption)

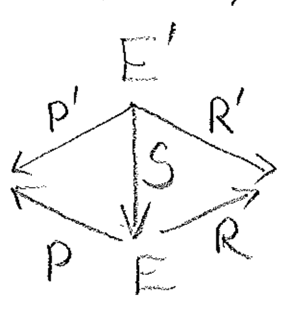
$\Rightarrow \quad S \wedge P\{Q\}S \wedge R$ (S mentions no program variables)

$\Rightarrow \exists e\, S \wedge P\{Q\}\exists e.\, S \wedge R$ (property of Hoare logic)

$\Rightarrow \quad RHS$ (definition of ;)

In fact we can get a looser specification still by combining this with the previous reasoning. Let $P', R' : E' \to V$. Then define:

$$(P', R') \preceq (P, R) \triangleq (S;P) \Rightarrow P' \wedge R' \Rightarrow (S;R)$$




so that justifies Oege's construction, in preference to the standard one.

Now we can justify composition of spans.

Consider $(P : E \to V, R : E \to V)$ and $(P' : E' \to V, R' : E' \to V)$.

Take first the simple case when $E = E'$ and $R = P'$.

Then clearly by Hoare logic

$$\forall Q, Q'.\ P\{Q\}R \wedge R\{Q'\}R' \vdash P\{Q;Q'\}R'$$

(denoted ⓢ)

We want to define the composition of ~~specifications~~ specifications as the loosest specification met by $Q;Q'$ whenever $Q$ meets the first of them and $Q'$ meets the other, ~~i.e.~~, so that the law quoted above could be written

$$(P, R) ⓢ (R, R') \preceq (P, R')$$

To get the loosest specification, just replace the $\preceq$ by equality $\triangleq$, and take this as the <u>definition</u> of ⓢ. But of course we need to generalise the definition to cover the case where the postcondition of the first operand differs from the precondition of the second, and even the logical variables may differ.

To get a general definition, we only need to reduce it to the special case. Let $E_0$ be a possibly fresh 'abstract' space, and let

$$S : E_0 \to E \quad , \quad S' : E_0 \to E'.$$

We already have proved

$$(P, R) \preceq ((S;P), (S;R))$$

$$(P', R') \preceq ((S';P'), (S';R'))$$

Since we want ⨟ to be monotonic, we require

$$(P,R) ⨟ (P',R') \preceq ((S;P),(S;R)) ⨟ ((S';P'),(S';R')).$$

Now all we have to do is to choose $S$ and $S'$ such that

$$S;R = S';P'$$

This reduces the general to the simple case, so we just define

$$(P,R) ⨟ (P',R') \triangleq (S;P \,,\, S';R')$$

$$\begin{tikzcd}
 & & E_0 \arrow[dl, "S"'] \arrow[dr, "S'"] & & \\
 & \cdot \arrow[dl, "P"'] \arrow[dr, "R"] & & \cdot \arrow[dl, "P'"'] \arrow[dr, "R'"] & \\
 \cdot & & \cdot & & \cdot
\end{tikzcd}$$

that is why we need weak pullbacks, and have to define sequential composition in this way.

And that all makes me very happy.