Chapter 7 : The Reduction theorem.

In which we develop a generic solution to a class of problems.

So far you have seen, and used, our method of program construction employed to solve a small number of programming problems. We have mentioned that we choose our notation so that in a sense it does the work for us; all we have to do in many cases is to manipulate the notation according to a small set of laws. But our notation has another major benefit, it allows us to develop solutions which can be re-used.

It is a good idea, that every time you solve a new problem using our method, you should try to abstract the solution to a generic one. In that way you will build up a very useful "toolbox" of correct and re-useable solutions. Let us show how to do this.

Consider the postconditions for the problems you have seen so far.

Compute the sum of the values in f[0..N)

$$r = \langle + j : 0 \le j < N : f.j \rangle$$

Compute the product of the values in f[20..100)

$$p = \langle * j : 20 \le j < 100 : f.j \rangle$$

Determine the largest value in the array f[0.200)

$$l = \langle \uparrow j : 0 \le j < 200 : f.j \rangle$$

Now if we look at the shape of each of these postconditions we notice that the are quite similar. They all are instances of a more abstract shape

$$r = \langle \oplus j : \alpha \le j < \beta : f.j \rangle$$

Where $\oplus$ is of course an associative, symmetric binary operator which has an identity element, and α and β are the lower and upper bounds on the range.

Model the problem domain.

Now let us develop a little model of this problem domain.

* (0) $C.n \quad = \quad \langle \oplus j : \alpha \le j < n : f.j \rangle \qquad , \alpha \le n \le \beta$

Consider

$$\begin{aligned} & C.\alpha \\ = & \quad \{(0) \text{ in model}\} \\ & \langle \oplus j : \alpha \le j < \alpha : f.j \rangle \\ & \quad \{\text{ empty range }\} \\ & Id_\oplus \end{aligned}$$

Which gives us

\- (1) $C.\alpha \quad = \quad Id_\oplus$

Consider

$$\begin{aligned} & C.(n+1) \\ = & \quad \{(0) \text{ in model }\} \\ & \langle \oplus j : \alpha \le j < n+1 : f.j \rangle \\ = & \quad \{\text{ split off } j = n \text{ term }\} \\ & \langle \oplus j : \alpha \le j < n : f.j \rangle \oplus f.n \end{aligned}$$

Which gives us

\- (2) $C.(n+1) \quad = \quad C.n \oplus f.n \qquad , \alpha \le n < \beta$

Rewrite the postcondition in terms of the model.

Given this model we can now rewrite our postcondition as follows.

Post : $r = C.\beta$

Invariants.

p0 : $r = C.n$
P1 : $\alpha \le n \le \beta$

Noting that $P0 \wedge P1 \wedge n = \beta \Rightarrow$ Post, we choose our loop guard

Guard.

$n \ne \beta$

Establish invariants.

Appealing to (1) we can establish both invariants by the assignment

$n, r := \alpha, \mathbf{Id}_{\oplus}$

Variant.

$\beta - n$

Loop body.

$(n, r := n+1, E).P0$

$=$ {textual substitution}

$E = C.(n+1)$

$=$ {(2) above}

$E = C.n \oplus f.n$

$=$ {P0}

$E = r \oplus f.n$

Finished program.

n, r := α, Id⊕
; do n ≠ β →
n, r := n+1, r ⊕ f.n
od

$\{P0 \wedge P1 \wedge n = \beta\}$

$\Rightarrow$

$\{r = C.\ \beta\}$

This is known as the Reduction Theorem.

Now, whenever we are faced with the problem of writing a program to achieve a postcondition of the shape

$r = \langle \oplus j : \alpha \leq j < \beta : f.j \rangle$

we can simply appeal to this theorem, instantiate ⊕, α, and β appropriately, and be guaranteed that we have a correct solution.