Chapter 9: The Partitioned Reduction Theorem II

In which we construct the generic solution to the partitioned reduction problem.

We are given an array f[0..N) of int which contains values and we are asked to construct a program to establish the following

$$\{\ r = \langle \oplus j : \alpha \le j < \beta : g.(f.j) \rangle \ ,\ 0 \le \alpha \le \beta \le N\ \}$$

where

$$g.x \quad = \quad h.x \quad \Leftarrow \quad Q.x$$
$$g.x \quad = \quad Id\oplus \quad \Leftarrow \quad not.(Q.x)$$

Postcondition.

$$Post : \ r = \langle \oplus j : \alpha \le j < \beta : g.(f.j) \rangle$$

Strengthen postcondition.

$$Post' : r = \langle \oplus j : \alpha \le j < n : g.(f.j) \rangle \wedge n = \beta$$

Domain modelling.

Inspired by the shape of our postcondition, we now proceed to develop a little mathematical model of our domain. We begin with a single postulate.

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

The function g is defined as follows:

* (1) $g.x \quad = \quad h.x \quad \Leftarrow \quad Q.x$

* (2) $g.x \quad = \quad Id\oplus \quad \Leftarrow \quad not.(Q.x)$

We now explore some theorems.

Consider

$C.\alpha$

= {(0) in model }

$\langle \oplus j : \alpha \le j < \alpha : g.(f.j) \rangle$

= { empty range }

$Id\oplus$

Which gives us

- (3) $C.\alpha \ = \ Id_\oplus$

Consider

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

Which gives us

- (4) $C.(n+1) \ = \ C.n \oplus h.(f.n) \quad \Leftarrow \quad Q.(f.n) \quad , \alpha \leq n < \beta$

Now consider the other case.

$$
\begin{aligned}
& C.(n+1) \\
= \quad & \{(0) \text{ in model}\} \\
& \langle \oplus\, j : \ \alpha \ \leq j < \ n+1 : g.(f.j)\rangle \\
= \quad & \{\text{split off } j = n \text{ term }\} \\
& \langle \oplus\, j : \ \alpha \ \leq j < \ n : g.(f.j)\rangle \oplus g.(f.n) \\
= \quad & \{ \text{ case not.}(Q.(f.n)), (2) \text{ in model } \} \\
& \langle \oplus\, j : \ \alpha \ \leq j < \ n : g.(f.j)\rangle \oplus Id_\oplus \\
= \quad & \{ \ (0) \text{ in model } \} \\
& C.n \oplus Id_\oplus
\end{aligned}
$$

Which gives us

- (5) $C.(n+1) = C.n \oplus Id\oplus \Leftarrow not.(Q.(f.n)) \quad , \alpha \leq n < \beta$

This completes our model.

Rewrite postcondition in terms of model.

Post ' : $r = C.n \wedge n = \beta$

Invariants.

P0 : $r = C.n$

P1 : $\alpha \leq n \leq \beta$

Termination.

We observe that

$$P0 \wedge P1 \wedge n = \beta \quad \Rightarrow \quad Post$$

Establishing the invariants.

To establish P0 we need to bind r to the value of C.n, for some n. Theorem (1) in our model gives us the value of C.n when $n = \alpha$. So the following assignment establishes P0 and also P1.

$n, r := \alpha, Id\oplus$

Guard.

$n \neq \beta$

Variant.

β-n

Loop body.

$$
\begin{array}{ll}
 & (n, r := n+1, E).P0 \\
= & \qquad \{ \text{ textual substitution } \} \\
 & E = C.(n+1) \\
= & \qquad \{\text{Case analysis, } Q.(f.n) \text{ , } P1 \text{ and } n \neq \beta \text{ allow us to appeal to (4)}\} \\
 & E = C.n \oplus h.(f.n) \\
= & \qquad \{P0 \} \\
 & E = r \oplus h.(f.n)
\end{array}
$$

Giving us the program fragment

[] Q.(f.n) → n, r := n+1, r ⊕ h.(f.n)

We now look at the other case.

$$
\begin{array}{ll}
 & (n, r := n+1, E).P0 \\
= & \qquad \{ \text{ textual substitution } \} \\
 & E = C.(n+1) \\
= & \qquad \{\text{Case analysis, } not.(Q.(f.n)) \text{ , } P1 \text{ and } n \neq \beta \text{ allow us to appeal to (5)}\} \\
 & E = C.n \oplus Id\oplus \\
= & \qquad \{P0 \} \\
 & E = r \oplus Id\oplus
\end{array}
$$

Giving us the program fragment

[] not.(Q.(f.n)) → n, r := n+1, r ⊕ Id⊕

As $(Q.x \vee not.(Q.x)) \equiv$ true we have covered all possibilities so we can now write the finished loop program.

Finished program.

```
n, r := α, Id⊕
;do n ≠ β      →

               if Q.(f.n)       →    n, r := n+1, r ⊕ h.(f.n)
               [] not.(Q.(f.n)) →    n, r := n+1, r ⊕ Id⊕
               fi

od

{P0 ∧ P1 ∧ n = β}
```