Chapter 20: Strengthening Postconditions and Choosing Invariants.

In the examples done so far we have often used strengthening to rewrite the postcondition so as to introduce an "∧" operator and so have the shape necessary to start constructing a loop program. Most of our examples of strengthening have involved us replacing the constant upper bound of the range in the postcondition with a suitably bound variable. However, there are many ways to strengthen and the manner in which we strengthen determines the shape of the loop invariant we will get and that in turn determines the program we get. In some cases the complexity of the resulting program will be different depending on how we choose to strengthen.

In this chapter, we look at some other ways to strengthen a postcondition. Each of them results in us choosing a different invariant. before we do some examples we should ask the question "What is the role of a loop invariant?" Well, if we can establish it and maintain it as we decrease vf until ¬B is true, then we will have arrived at a state which implies the postcondition.

So the qualities we should look for in an invariant are that

It is "relatively" easy to establish it.

It can be maintained as we decrease vf

It together with the negation of the guard will imply our postcondition.

We will now look at a few simple examples of program construction, but I would like you to study them from the point of view of the loop invariant asking questions like

What strengthening did we use?

How easy was it to establish the invariant?

How easy was it to maintain it as vf was decreased?

How easy was it to demonstrate that the result implied the postcondition?

Example 1.

Given f[0..N) of int, construct a program to achieve the postcondition

Post : $r = \langle \uparrow i : 0 \leq i < N : f.i \rangle$

We strengthen to get

Post' : $r = \langle \uparrow i : 0 \leq i < n : f.i \rangle \ \wedge n = N$

Model.

* (0) C.n = $\langle \uparrow i : 0 \le i < n : f.i \rangle$, $0 \le n \le N$

- (1) $C.0 = id\uparrow$

- (2) $C.(n+1) = C.n \uparrow f.n$, $0 \le n < N$

We can now rewrite our strengthened postcondition as

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

Invariants.

$P0 : r = C.n$
$P1 : 0 \le n \le N$

Establish invariants.

$n, r := 0, id\uparrow$

Guard.

$n \neq N$

Achieving Post.

$P0 \wedge P1 \wedge n = N \Rightarrow Post$

vf.

$N-n$

Loop body.

$(n, r := n+1, E).P0$
= {text substitution}
$E = C.(n+1)$
= {(2)}
$E = C.n \uparrow f.n$
= {P0}
$E = r \uparrow f.n$

Finished program.

n, r := 0, id↑
;Do n ≠ N —>

n, r := n+1, r ↑ f.n

Od
{r = C.N}

Example 2.

We use the same problem as in Example 1. Given f[0..N) of int, construct a program to achieve the postcondition

Post : r = ⟨ ↑ i : 0 ≤ i < N : f.i ⟩

This time we go for a different strengthening step. We strengthen to get

Post' : r = ⟨ ↑ i : n ≤ i < N : f.i ⟩ ∧ n = 0

Model.

* (0) C.n = ⟨ ↑ i : n ≤ i < N : f.i ⟩ , 0 ≤ n ≤ N

- (1) C.N = id↑

- (2) C.(n-1) = C.n ↑ f.(n-1) , 0 < n ≤ N

We can now rewrite our strengthened postcondition as

Post' : r = C.n ∧ n = 0

Invariants.

P0 : r = C.n
P1 : 0 ≤ n ≤ N

Establish invariants.

n, r := N, id↑

Guard.

n ≠ 0

Variant.

n

Loop body.

(n, r := n-1, E).P0
= {text substitution}
E = C.(n-1)
= {(2)}
E = C.n ↑ f.(n-1)
= {P0}
E = r ↑ f.(n-1)

Finished program.

n, r := N, id↑ ;
Do n ≠ 0 —>

n, r := n-1, r ↑ f.(n-1)

Od
{r = C.n ∧ n = 0}

Example 3.

We stay with the same example. Given f[0..N) of int, construct a program to achieve the postcondition

Post : r = ⟨ ↑ i : 0 ≤ i < N : f.i ⟩

This time we do a slightly different model.

Model.

* (0) C.n = ⟨ ↑ i : n ≤ i < N : f.i ⟩ , 0 ≤ n ≤ N

- (1) C.N = id↑

- (2) C.n = C.(n+1) ↑ f.n , 0 ≤ n < N

Invariants.

We choose to use a tail invariant P0

P0 : $r \uparrow C.n = C.0$
P1 : $0 \leq n \leq N$

Establish invariants.

$n, r := 0, id\uparrow$

Termination.

We observe

Post' : $r \uparrow C.n = C.0 \wedge n = N$
=> {algebra}
$r \uparrow C.N = C.0$
= {(1)}
$r = C.0$

Guard.

$n \neq N$

Variant.

$N-n$

Loop body.

$r \uparrow C.n = C.0$
= {(2)}
$r \uparrow C.(n+1) \uparrow f.n = C.0$
= {WP.}
$(n, r := n+1, r \uparrow f.n).P0$

Finished program.

```
n, r := 0, id↑
;Do n ≠ N —>

        n, r := n+1,      r ↑ f.n

 Od
{r  = C.0}
```

In this example, we get the same program as we did in Example 1 but the invariant we used was different.

This use of a tail invariant is quite common and you will see this shape used a lot more next year.