Chapter 11: The Bounded Linear Search Theorem.

In which we introduce a very useful searching algorithm.

Suppose we are given an array f[0..N) of int, where $\{1 \le N\}$, and a target value X, and we are asked to find the location of the leftmost X in f, i.e. the smallest index n where f.n = X. This time however, we have no guarantee that X is in f.

We begin as usual with a problem specification.

There are two possibilities, either X is there or X is absent. In the case where it is present the postcondition we want to achieve is the same as in the Linear Search

$$\text{Post1} : \langle \forall\, j : 0 \le j < n : f.j \ne X \rangle \wedge f.n = X$$

In the case where it is absent we could phrase the postcondition as

$$\text{Post2} : \langle \forall\, j : 0 \le j < N : f.j \ne X \rangle$$

But instead we choose to phrase it as follows

$$\text{Post2} : \langle \forall\, j : 0 \le j < n : f.j \ne X \rangle \wedge (n = N\text{-}1 \wedge f.n \ne X)$$

In this way both Post1 and Post2 contain exactly the same quantified expression. We can now combine them to give our overall postcondition

$$\text{Post} : \langle \forall\, j : 0 \le j < n : f.j \ne X \rangle \wedge (f.n = X \vee (n = N\text{-}1 \wedge f.n \ne X))$$

Recalling one of our theorems from Boolean Calculus $[X \vee (\neg X \wedge Y) \equiv X \vee Y]$ we can now simplify this to get

$$\text{Post} : \langle \forall\, j : 0 \le j < n : f.j \ne X \rangle \wedge (f.n = X \vee n = N\text{-}1)$$

Model problem domain.

* (0) C.n $\quad \equiv \quad \langle \forall\, j : 0 \le j < n : f.j \ne X \rangle \qquad , 0 \le n \le N$

Consider.

$$
\begin{array}{ll}
 & C.0 \\
= & \quad \{(0) \text{ in model }\} \\
 & \langle \forall\, j : 0 \le j < 0 : f.j \ne X \rangle \\
= & \quad \{ \text{ empty range } \} \\
 & \text{true}
\end{array}
$$

- (1) $C.0 \quad \equiv \quad true$

Consider

$$
\begin{array}{ll}
 & C.(n+1) \\
= & \quad \{(0) \text{ in model }\} \\
 & \langle \forall\, j : 0 \le j < n+1 : f.j \ne X \rangle \\
= & \quad \{ \text{ split off } j = n \text{ term}\} \\
 & \langle \forall\, j : 0 \le j < n : f.j \ne X \rangle \wedge f.n \ne X \\
= & \quad \{(0) \text{ in model}\} \\
 & C.n \wedge f.n \ne X
\end{array}
$$

- (2) $C.(n+1) \quad \equiv \quad C.n \wedge f.n \ne X \qquad , 0 \le n < N$

We can now rewrite our postcondition as

$$\text{Post} : C.n \wedge (f.n = X \vee n = N\text{-}1)$$

Choose Invariants.

We choose as our invariants

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

Termination.

We note that

$$P0 \wedge P1 \wedge (f.n = X \vee n = N\text{-}1) \Rightarrow \text{Post}$$

Establish Invariants.

Our model (1) shows us that we can establish P0 by the assignment

$$n := 0$$

This also establishes P1.

Guard.

We choose our loop guard to be

$B : f.n \neq X \wedge n \neq N - 1$

Calculate Loop body.

Decreasing the variant by the assignment n := n+1 is a standard step and maintains P1. Let us se what effect it has on P0

$$
\begin{array}{ll}
 & (n := n+1).\ P0 \\
= & \quad \{\text{textual substitution}\} \\
 & C.(n+1) \\
= & \quad \{(2)\ \text{above}\} \\
 & C.n \wedge f.n \neq X \\
= & \quad \{\text{Remember } P0 \wedge f.n \neq X \wedge n \neq N - 1 \text{ at start of loop body}\} \\
 & \text{true}
\end{array}
$$

Final program.

```
  n := 0
; do f.n ≠ X ∧ n ≠ N - 1→

            n := n+1

  od
{C.n ∧ (f.n = X ∨ n = N-1)}
```

When the loop terminates we can now decide which outcome has occurred and communicate this to the user by adding a simple if..fi as follows.

```
if f.n= X                → write('X found at position', n )
[] n = N-1 ∧ f.n ≠ X     → write('X is not in f')
fi
```