

## The natural unfold

(filed under “general programming techniques”)

A natural unfold,  $h$ , is a recursive function of the shape

$$h = \text{unfold}.p.f \equiv h.n = \text{if } p.n \text{ then } 0 \text{ else } 1 + h.(f.n)$$

In this note we will derive a procedure,  $\text{unfoldn}$ , which computes a natural unfold.

\*            \*  
              \*

### Specification

Let  $p : a \rightsquigarrow \text{Bool}$ ,  $f : a \rightsquigarrow a$  and  $n : a$ . We specify  $\text{unfoldn}$  by

- $\text{unfoldn}.p.f.n = \text{if } p.n \text{ then } 0 \text{ else } 1 + \text{unfoldn}.p.f.(f.n)$
- $[\text{wp}(\text{unfoldn}.p.f.n).R \equiv R]$  for all  $R$ .

\*            \*  
              \*

### Implementation

As  $\text{unfoldn}$  is recursive, we will use a loop to compute it. For  $r : \text{Nat}$ ,  $N : a$  the postcondition of our loop is

$$r = \text{unfoldn}.b.f.N \quad .$$

We choose as loop invariant

$$\text{unfoldn}.b.f.N = r + \text{unfoldn}.b.f.n$$

which we establish with  $r := 0 ; n := N$ .

With regards to maintenance of the invariant we observe

$$\begin{aligned}
 & \llbracket \text{Context: } b.n \Rightarrow \text{unfoldn.b.f.n} = 0 \\
 & \quad r + \text{unfoldn.b.f.n} \\
 & = \{ b.n \} \\
 & \quad r \\
 & \rrbracket
 \end{aligned}$$

and

$$\begin{aligned}
 & \llbracket \text{Context: } \neg b.n \Rightarrow \text{unfoldn.b.f.n} = 1 + \text{unfoldn.b.f.}(f.n) \\
 & \quad r + \text{unfoldn.b.f.n} \\
 & = \{ \neg b.n \} \\
 & \quad r + 1 + \text{unfoldn.b.f.}(f.n) \\
 & = \nabla r := r + 1 ; n := f.n \nabla \\
 & \quad r + \text{unfoldn.b.f.n} \\
 & \rrbracket .
 \end{aligned}$$

We have derived the following procedure

$$\begin{aligned}
 \text{unfoldn.p.f.N} = & \llbracket \mathbf{var} \ n : a; \ r : \mathbf{Nat}; \\
 & \quad r := 0; \ n := N; \\
 & \quad \mathbf{do} \ \neg b.n \ \rightarrow \\
 & \quad \quad r := r + 1 \\
 & \quad \quad ; n := f.n \\
 & \quad \mathbf{od}; \\
 & \quad r \\
 & \rrbracket .
 \end{aligned}$$

So concludes our derivation.

\*            \*

\*

Example: The Linear Search

For  $b : \text{Nat} \rightsquigarrow \text{Bool}$ , the Linear Search is a standard algorithm for finding the least  $x$  such that  $b.x$ , given that such an  $x$  exists. For  $i : \text{Nat}$  a program for the Linear Search is

```
[[con  $b : \text{Nat} \rightsquigarrow \text{Bool}$ ; var  $x : \text{Nat}$ ;  
  { $\langle \exists i :: b.i \rangle$ }  
   $x := \text{unfoldn}.b.(1+).0$   
  { $b.x \wedge \langle \forall i : i < x : \neg b.i \rangle$ }  
]] .
```

2008.03.17

E. Emmanuel Macaulay  
eric@mathmeth.com