

# The Natural Fold

(filed under “general programming techniques”)

## Introduction

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

$$h.0 = e \wedge h.(n + 1) = f.(h.n)$$

Many functions that we need to compute can be expressed as natural folds. In this note we will derive a procedure,  $foldn$ , for computing a natural fold. We begin by giving a formal specification.

\*                    \*  
                         \*  
                         \*

## Specification

Let  $f : a \rightsquigarrow a$ ,  $e : a$ , for any  $a$ , and let  $n : \mathbf{Nat}$ .  $foldn$  may be specified by

- $foldn.f.e.0 = e \wedge foldn.f.e.(n + 1) = f.(foldn.f.e.n)$
- $[wp.(foldn.f.e.n).R \equiv R]$  for all  $R$ .

\*                    \*  
                         \*  
                         \*

## Implementation

As  $foldn$  is recursive, we will compute it with a loop. For  $r : a$ ,  $N : \mathbf{Nat}$ , our loop’s postcondition is

$$r = foldn.f.e.N$$

We choose as invariant

$$r = foldn.f.e.n$$

From the first conjunct of the specification we see that we may easily establish this invariant with  $r := e ; n := 0$  .

The second conjunct of the specification suggests that we modify  $n$  with the statement  $n := n + 1$  . We observe

$$\begin{aligned}
& \llbracket \text{Context: } r = \textit{foldn.f.e.n} \\
& \quad (n := n + 1).(\textit{foldn.f.e.n}) \\
& = \quad \{ \textit{substitution} \} \\
& \quad \textit{foldn.f.e.}(n + 1) \\
& = \quad \{ \textit{property of foldn} \} \\
& \quad f.(\textit{foldn.f.e.n}) \\
& = \quad \{ \textit{Context} \} \\
& \quad f.r \\
& \rrbracket
\end{aligned}$$

Thus we see that the statement  $r := f.r ; n := n + 1$  maintains the invariant and progresses to termination.

And so we have derived the following procedure:

$$\begin{aligned}
\textit{foldn.f.e.N} = \llbracket & \mathbf{var } n : \mathbf{Nat}; r : a; \\
& r := e; n := 0; \\
& \mathbf{do } n \neq N \rightarrow \\
& \quad r := f.r \\
& \quad ; n := n + 1 \\
& \mathbf{od}; \\
& r \\
& \rrbracket
\end{aligned}$$

By the First Termination Lemma the loop terminates.

So concludes our derivation of the Natural Fold algorithm.

2008.02.18

E. Emmanuel Macaulay  
eric@mathmeth.com