

The list unfold

The list unfold, $unfoldl$, is specified by

$$\begin{aligned} unfoldl & : (a \rightsquigarrow \mathbf{Bool}) \rightsquigarrow (a \rightsquigarrow b) \rightsquigarrow (a \rightsquigarrow a) \rightsquigarrow a \rightsquigarrow [b] \\ unfoldl.p.f.g.x & = \mathbf{if } p.x \mathbf{ then } [] \mathbf{ else } f.x : unfoldl.p.f.g.(g.x) \end{aligned}$$

In this note we derive a procedure to compute $unfoldl$.

* *
 *

As $unfoldl$ is a recurrence, we will compute it with a loop. For $ys : [b]$ and $X : a$ the loop's postcondition is

$$ys = unfoldl.p.f.g.X \quad .$$

We choose as invariant

$$unfoldl.p.f.g.X = ys ++ unfoldl.p.f.g.x$$

which is established with $ys, x := [], X$.

With regards to maintenance of the invariant we observe

$$\begin{aligned} & [[\text{Context: } p.x \Rightarrow unfoldn.p.f.g.x = [] \\ & \quad ys ++ unfoldn.p.f.x \\ & = \{ p.x \} \\ & \quad ys \\ &]] \end{aligned}$$

and

$$\begin{aligned} & [[\text{Context: } \neg p.x \Rightarrow unfoldn.p.f.g.x = f.x : unfoldl.p.f.g.(g.x) \\ & \quad ys ++ unfoldn.p.f.x \\ & = \{ \neg p.x \} \\ & \quad ys ++ f.x : unfoldl.p.f.g.(g.x) \\ & = \{ \text{property of } : \} \\ & \quad ys ++ [f.x] ++ unfoldl.p.f.g.(g.x) \\ & = \nabla ys := ys ++ [f.x] \quad ; \quad x := g.x \quad \nabla \\ & \quad ys ++ unfoldn.p.f.x \end{aligned}$$

]]

* *
*

We have derived the procedure

```
unfoldl.p.f.g.X = [[var x : a; ys : [b];  
                    x := X; ys := [ ];  
                    do  $\neg p.n \rightarrow$   
                        ys := ys ++ [f.x]  
                        ; x := g.x  
                    od;  
                    ys  
                    ]].
```

So concludes our derivation.

2008.03.25

E. Emmanuel Macaulay
eric@mathmeth.com