# CS457 Functional Programming
# Mark Jones Winter 2012
# Homework 8

Russell Miller

March 7, 2012

## QUESTION 1

**Prove the following law holds for all `f`, `e`, and `g`.**

```
foldr f e . map g = foldr (f . g) e
```

I'm going to rewrite this law using a variable xs to represent the list argument for each side. We'll say that this law is P(xs).

```
P(xs) = (foldr f e . map g) xs = (foldr (f . g) e) xs
```

Similar to the proof we did in class, we will need to prove 3 cases: `P([])`, `P(⊥)`, and `P(xs)` $\Rightarrow$ `P(x:xs)`, where $\perp$ is execution that does not terminate properly.

First we need the definition of `foldr`:

```
foldr f z []     = z                     (foldr.0)
foldr f z (x:xs) = f x (foldr f z xs)    (foldr.1)
```

(found in the Prelude using Hugs's `:f` command.)

The definition of map we defined in class.

```
map f []            = []                 (map.0)
map f (x:xs)        = f x : map f xs     (map.1)
```

Great! `foldr` and `map` are defined for `[]` and `x:xs`. Now we need to come up with laws about `map` and `foldr` for the case of $\perp$.

In class we talked about `map f ⊥`.

```
map f ⊥      = ⊥ (map.⊥)
```

By looking at the definition of `foldr`, it is clear that it will work the same. It does something to each element of a list, and recursively works through the list the same way map does just that. Thus:

```
foldr f z ⊥  = ⊥ (foldr.⊥)
```

Now we're ready to prove the property for the 3 cases talked about earlier.

P([]):
We want to show that
`(foldr f e . map g) [] = foldr (f . g) e []`

```
LHS = (foldr f e . map g) []
    = foldr f e (map g [])                {definition of .}
    = foldr f e []                        {by map.0}
    = e                                   {by foldr.0}
RHS = foldr (f . g) e []
    = e                                   {by foldr.0}
```

P(⊥):
We want to show that
`(foldr f e . map g) ⊥ = foldr (f . g) e ⊥`

```
LHS = (foldr f e . map g) ⊥
    = foldr f e (map g ⊥)                 {definition of .}
    = foldr f e ⊥                         {by map.⊥}
    = ⊥                                   {by foldr.⊥}
RHS = foldr (f . g) e ⊥
    = ⊥                                   {by foldr.⊥}
```

P(xs) ⇒ P(x:xs):
We want to show that
`(foldr f e . map g) (x:xs) = foldr (f . g) e (x:xs)`

```
LHS = (foldr f e . map g) (x:xs)
    = foldr f e (map g (x:xs))            {definition of .}
    = foldr f e (g x : map g xs)          {by map.1}
    = f (g x) (foldr f e (map g xs))      {by foldr.1}
RHS = foldr (f . g) e (x:xs)
    = (f . g) x (foldr (f . g) e xs)      {by foldr.1}
    = (f . g) x ((foldr f e . map g) xs)  {induction, P(xs)}
    = f (g x) (foldr f e (map g xs))      {definition of .}
```

■