Monad Laws

September 17, 2024

Outline

Outline

Law 1:

`return x >>= f = f x`

This means that a monad containing x when passed to `f`, should return the same value as `f` applied to (unboxed) `x`.

Outline

Law 2:

`mx >>= return = mx`

This is the dual of the first law. This says that a "boxed" x when passed to return should evaluate to the "boxed" x itself.
One way to see this as follows: `return` unboxes `x` from `x`, and then evaluates to the boxed value `x`. This is in fact, `mx` itself.

Outline

Law 3: associativity of bind

```
(mx >>= f) >>= g     =     mx >>= (\x -> f x >>= g)
```

This is the associative law for bind. Note that any associative law involves only one operator.

We now verify these laws are satisfied by the `Maybe` monad

Outline

Maybe monad: return and »=

```
instance Monad Maybe where
        return x = Just x

        mx >>= f = f x
```

Law 1

Monadic Law 1 for Maybe

```
  return x >>= f
= (Just x) >>= f        [definition of return]
= f x                   [definition of >>=]
```

which satisfies the Monadic Law 1

Law 2

Monadic Law 2 for Maybe

```
(Just x) >>= return
    = return x                    [definition of >>=]
```

which satisfies the Monadic Law 2

Law 3

Monadic Law 3 for Maybe

```
((Just x)>>= f) >>= g
  = (f x) >>= g                [definition of >>=]
  = (Just y) >>= g             [assuming (Just y)=f x]
  = g y                        [definition of >>=]

(Just x) >>= (\a -> f a >>= g)
      = (f x) >>= g            [definition of >>=]
      = (Just y) >>= g         [assuming (Just y)=f x]
      = g y
```

This satisfies the Monadic Law 3.

Law 3

Monadic Law 3 for Maybe

```
((Just x)>>= f) >>= g
  = (f x) >>= g                [definition of >>=]
  = (Just y) >>= g             [assuming (Just y)=f x]
  = g y                        [definition of >>=]

(Just x) >>= (\a -> f a >>= g)
      = (f x) >>= g            [definition of >>=]
      = (Just y) >>= g         [assuming (Just y)=f x]
      = g y
```

This satisfies the Monadic Law 3.

Monadic Law 3 for Maybe

```
((Just x)>>= f) >>= g
  = (f x) >>= g                [definition of >>=]
  = (Just y) >>= g             [assuming (Just y)=f x]
  = g y                        [definition of >>=]

(Just x) >>= (\a -> f a >>= g)
      = (f x) >>= g            [definition of >>=]
      = (Just y) >>= g         [assuming (Just y)=f x]
      = g y
```

This satisfies the Monadic Law 3.
(the derivation can be shortened, of course.)