Hw9 非编程题

2015080118 软件61 崔殷庇

1. Maybe

```
fmap :: (a -> b) -> Maybe a -> Maybe b
fmap f Nothing  = Nothing          (fmap.1)
fmap f (Just a) = Just (f a)       (fmap.2)
```

1. *fmap id == id*

   case 1:

   ```
   fmap id Nothing === Nothing       (fmap.1)
                   === id Nothing    (id)
   ```

   case 2:

   ```
   fmap id (Just a) === Just (id a)  (fmap.2)
                    === Just a       (id)
                    === id (Just a)  (id)
   ```

   Therefore *fmap id == id* for all cases.

2. fmap (f . g) == fmap f . fmap g

   case 1:

   ```
   fmap (f . g) Nothing       === Nothing                    (fmap.1)
   (fmap f . fmap g) Nothing  === fmap f (fmap g Nothing)    (.)
                              === fmap f Nothibg             (fmap.1)
                              === Nothing                    (fmap.1)
   ```

   case 2:

   ```
   fmap (f . g) (Just a)       === Just((f . g) a)           (fmap.2)
   (fmap f . fmap g) (Just a)  === fmap f (fmap g (Just a))  (.)
                               === fmap f (Just (g a))       (fmap.2)
                               === Just (f (g a))            (fmap.2)
                               === Just ((f . g) a)          (.)
   ```

   Therefore *fmap (f . g) == fmap f . fmap g* for all cases.

2. []

```
fmap :: (a -> b) -> [a] -> [b]
fmap f []     = []                (fmap.1)
fmap f (a:as) = f a : fmap f as   (fmap.2)
```

1. *fmap id == id*

   case 1:

   ```
   fmap id [] === []       (fmap.1)
              === id []    (id)
   ```

   case 2:

   ```
   fmap id (x:xs) === id x : fmap id xs    (fmap.2)
                  === id x : id xs         (IH)
                  === x:xs                 (id)
                  === id (x:xs)            (id)
   ```

   Therefore *fmap id == id* for all cases.

2. *fmap (f . g) == fmap f . fmap g*

   case 1:

   ```
   fmap (f . g) []      === []                   (fmap.1)
   (fmap f . fmap g) [] === fmap f (fmap g [])   (.)
                        === fmap f []            (fmap.1)
                        === []                   (fmap.1)
   ```

   case 2:

   ```
   fmap (f . g) (x:xs)      === (f . g) x : fmap (f . g) xs        (fmap.2)
   (fmap f . fmap g) (x:xs) === fmap f (fmap g (x:xs))             (.)
                            === fmap f (g x : fmap g xs)          (fmap.2)
                            === (f . g) x : fmap f (fmap g xs)    (fmap.2)
                            === (f . g) x : (fmap f . fmap g) xs  (.)
                            === (f . g) x : fmap (f . g) xs       (IH)
   ```

   Therefore *fmap (f . g) == fmap f . fmap g* for all cases.

3. Either

```
fmap :: (a -> b) -> Either c a -> Either c b
fmap _ (Left c)  = Left c          (fmap.1)
fmap f (Right a) = Right (f a)     (fmap.2)
```

1. *fmap id == id*

   case 1:

   ```
   fmap id (Left a) === Left a           (fmap.1)
                    === id (Left a)      (id)
   ```

   case 2:

   ```
   fmap id (Right a) === Right (id a)    (fmap.2)
                     === Right a         (id)
                     === id (Right a)    (id)
   ```

   Therefore *fmap id == id* for all cases.

2. *fmap (f . g) == fmap f . fmap g*

   case 1:

   ```
   fmap (f . g) (Left a)      === Left a                      (fmap.1)
   (fmap f . fmap g) (Left a) === fmap f (fmap g (Left a))    (.)
                              === fmap f (Left a)             (fmap.1)
                              === Left a                      (fmap.1)
   ```

   case 2:

   ```
   fmap (f . g) (Right a)      === Right ((f . g) a)          (fmap.2)
   (fmap f . fmap g) (Right a) === fmap f (fmap g (Right a))  (.)
                               === fmap f (Right (g a))       (fmap.2)
                               === Right (f (g a))            (fmap.2)
                               === Right ((f . g) a)          (.)
   ```

   Therefore *fmap (f . g) == fmap f . fmap g* for all cases.