Лекция 5

Ilya Yaroshevskiy

24 сентября

Содержание

1 Functor

Определение.

```haskell
class Functor f where
  fmap :: (a -> b) -> f a -> f b
```

Примечание. $(\texttt{<\$>}) = fmap$ — инфиксный оператор

Пример.

```
Prelude> :kind (->)
(->) :: * -> * -> *
Prelude> :kind (->) Int
(->) Int :: * -> *
```

```haskell
instance Functor ((->) r)  where
  fmap :: (a -> b) -> (r -> a) -> r -> b
  fmap = (.)
```

1.1 Законы

1. `fmap id` $\equiv$ `id`
2. `fmap (f . g)` $\equiv$ `fmap f . fmap g`

2 Applicative

Определение.

```haskell
class Functor f => Applicative f where
  pure :: a -> f a
  (<*>) :: f (a -> b) -> f a -> f b
```

Пример.

```haskell
instance Applicative ((->) r) where
  pure :: a -> r -> a
  pure x = \_ -> x

  (<*>) :: (r -> a -> b) -> (r -> a) -> r -> b
  f <*> g = \x -> f x (g x)
```

2.1 Законы

1. `pure id <*> v ≡ b`
2. `pure ( . ) <*> u <*> v <*> w ≡ u <*> (v <*> w)`
3. `pure f <*> pure x ≡ pure (f x)`
4. `u <*> pure y ≡ pure($ y) <*> u`

3 Alternative

Определение.

```haskell
class Applicative f => Alternative f where
  empty :: f a
  (<|>) :: f a -> f a -> f a
```

Пример.

```haskell
instance Alternative Maybe where
  empty :: Maybe a
  empty = Nothing

  (<|>) :: Maybe a -> Maybe a -> Maybe a
  Nothing <|> r = r
  l       <|> _ = l
```

```haskell
instance Alternative [] where
  empty :: [a]
  empty = []

  (<|>) :: [a] -> [a] -> {a}
  <|> = (++)
```

4 Traversable

Определение.

```haskell
class (Functor t, Foldable t)  => Traversable t where
  traverse :: Applicative f => (a -> f b) -> t a -> f (t b)
  sequenceA :: Applicative f => t (f a) -> f (t a)
```

Пример.

```haskell
instance Traversable Maybe where
  traverse :: Applicative f => (a -> f b) -> Maybe a -> f (Maybe b)
  traverse _ Nothing = pure Nothing
  traverse f (Just x) = Just <$> f x
```

```haskell
instance Traversable [] where
  traverse :: Applicative f => (a -> f b) -> [a] -> f [b]
  traverse f = foldr consF (pure [])
    where
      consF x ys = (:) <$> f x <*> ys
```