Декларативное программирование

Семинар №12, группа 22215

Завьялов А.А.

21 ноября 2022 г.

Кафедра систем информатики ФИТ НГУ

Глубокий смысл монадического связывания

```
class Applicative m => Monad m where
    -- `bind`, оператор монадического связывания
    (>>=)         :: m a -> (a -> m b) -> m b
    (>>)          :: m a -> m b -> m b
    return        :: a -> m a
```

Монадические законы

- `return a >>= k` ≡ `k a`
- `m >>= return` ≡ `m`
- **Ассоциативность**

  `m >>= (\x -> k x >>= h)` ≡ `(m >>= k) >>= h`

Глубокий смысл монадического связывания

- Что такое связывание (обычное)?
- Связывание значения с именем

```haskell
expr :: Double
expr =
  let a = 10
      b = 20
      c = 30
      d = 40
  in a + b * c / d
```

```haskell
expr' :: Double
expr' =
  (\a ->
   \b ->
   \c ->
   \d ->
     a + b * c / d)
  10 20 30 40
```

Глубокий смысл монадического связывания

- А теперь через монаду Identity
- Она очень простая – не делает *ничего*

```
expr' =
  (\a ->
   \b ->
   \c ->
   \d ->
     a + b * c / d)
  10 20 30 40
```

```
expr'' =
  runIdentity $
    return 10 >>= \a ->
    return 20 >>= \b ->
    return 30 >>= \c ->
    return 40 >>= \d ->
      return a + b * c / d
```

Глубокий смысл монадического связывания

- А теперь через do-нотацию

```
expr'' =
  runIdentity $
    return 10 >>= \a ->
    return 20 >>= \b ->
    return 30 >>= \c ->
    return 40 >>= \d ->
      return a + b * c / d
```

```
expr''' =
  runIdentity $ do
    a <- return 10
    b <- return 20
    c <- return 30
    d <- return 40
    return a + b * c / d
```

Вычислительные эффекты

Что такое вычислительный эффект

Свойства чистых функций

- **Полная определённость (тотальность)** – даёт ответ для любого значения аргумента
- **Детерминированность** – для заданных входных данных функция всегда возвращает одинаковый результат
- **Отсутствие побочных эффектов** – не ссылаются на данные, значения которых могут измениться в процессе работы программы и не производит таких изменений

Вычислительный эффект — любое действие, нарушающее свойства чистых функций

Какие бывают вычислительные эффекты

- **Частичность** – функция не определена для каких-либо значений аргумента (не возвращает значения для него, т.е. *не завершается*)
- **Недетерминированность**
  - **Нестабильность** – функция не всегда возвращает одинаковые значения для одного значения аргумента
  - **Множественность** – функция возвращает сразу несколько ответов
- **Побочный эффект** – функция не только возвращает результат, но и делает *что-то ещё*

Эффекты в Haskell

Эффект частичности

```
f :: Double -> Double
f d =
  let a = 10
      b = 20
      c = 30
  in a + b * c / d

>f 0
Infinity
```

Эффект частичности

Maybe!

```
f :: Double -> Double
f d =
  let a = 10
      b = 20
      c = 30
  in a + b * c / d

>f 0
Infinity
```

```
f :: Double -> Maybe Double
f d =
  let a = 10
      b = 20
      c = 30
  in if (abs d < 0.0001)
     then Nothing
     else
       Just (a + b * c / d)
>f 0
Nothing
>f 1
Just 610.0
```

Эффект частичности

```haskell
f :: Double -> Maybe Double
f d =
  let a = 10
      b = 20
      c = 30
  in if (abs d < 0.0001)
     then Nothing
     else
       Just (a + b * c / d)
```

```haskell
f :: Double -> Maybe Double
f d = do
  guard (abs d >= 0.0001)
  let a = 10
      b = 20
      c = 30
  return $ a + b * c / d
```

Maybe как монада

Реализуется достаточно просто:

```
instance Monad Maybe where
  return = Just

  Nothing >>= _ = Nothing
  (Just x) >>= f = f x

  (Just _) >> a = a
  Nothing >> _ = Nothing
```

Эффект множественности

- Квадратное уравнение может иметь два корня, один корень, не иметь корней
- `solver :: Double -> Double -> Double -> ?`

Эффект множественности

- Квадратное уравнение может иметь два корня, один корень, не иметь корней
- `solver :: Double -> Double -> Double -> [Double]`

Список как монада

Определяется следующим образом:

```
instance Monad [] where
  return x = [x]
  xs >>= k = concat (map k xs)
```

Связь монады списка и list comprehension

Следующие выражения эквивалентны:

```haskell
triples =
  [(x,y,z) |
   x <- [1..10],
   y <- [1..10],
   z <- [1..10],
   x^2 + y^2 == z^2]
```

```haskell
triples = do
  x <- [1..10]
  y <- [1..10]
  z <- [1..10]
  guard $ x^2 + y^2 == z^2
  return (x, y, z)
```

Ввод-вывод

- Частичный (файл может отсутствовать, прочитали до EOF)
- Недетерминированный (файл может измениться)
- Имеет побочные эффекты (печать в файл)

Ввод-вывод

- Частичный (файл может отсутствовать, прочитали до EOF)
- Недетерминированный (файл может измениться)
- Имеет побочные эффекты (печать в файл)

Нечистый!

Ввод-вывод

- Частичный (файл может отсутствовать, прочитали до EOF)
- Недетерминированный (файл может измениться)
- Имеет побочные эффекты (печать в файл)

Нечистый!

Но такой нужный...

Что почитать

- https://ruhaskell.org/posts/theory/2018/01/10/effects.html
- https://ruhaskell.org/posts/theory/2018/01/18/effects-haskell.html

Q&A