ФУНКЦИОНАЛЬНОЕ ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Санкт-петербургский государственный политехнический университет
Институт Компьютерных Наук и Технологий
Кафедра: Информационные и Управляющий Системы
Автор: Лукашин Антон Андреевич

2016-2017

Содержание

- Функторы
- Естественные преобразования
  - *Изоморфизмы*
  - *Начальный и терминальные объекты*
  - *Продукты и сопродукты*
- Монады

Содержание

Map

- Вернемся к определению функции map:

  «Для заданной функции f и списка xs, map f xs порождает список, получаемые вызовом функции f для всех элементов исходного списка»

  map f [x1, x2, …] = [f x1, f x2, …]

- Более формально можно дать следующее определение:
  - *map :: (a -> b) -> [a] -> [b]*
  - *map _ [] = []*
  - *map f (x:xs) = f x : map f xs*

Функторы

- Функтор иначе называют морфизмом категорий – функтор преобразует объекты и морфизмы одной категории в морфизмы и объекты новой категории
- Пусть заданы категории $\mathcal{C}$ и $\mathcal{D}$. Функтор F $: \mathcal{C} \to \mathcal{D}$ определяет для каждого объекто a из $\mathcal{C}$ соответствующий объект $\mathrm{F_O}(a)$ из $\mathcal{D}$, а также для каждого морфизма $f : a \to b$ из $\mathcal{C}$ – соответствующий морфизм $\mathrm{F_M}(f) : \mathrm{F_O}(a) \to \mathrm{F_O}(b)$ из $\mathcal{D}$, такие что:
  - *для все всех объектов a из $\mathcal{C}$*

$$\mathrm{F_M}(\mathrm{id}_a) = \mathrm{id}_{\mathrm{F_O}(a)} \qquad (1)$$

  - *для всех морфизмов $f : a \to b$ и $g : b \to c$ из $\mathcal{C}$*

$$\mathrm{F_M}(g \circ f) = \mathrm{F_M}(g) \circ \mathrm{F_M}(f) \qquad (2)$$

- Функтор сопоставляющий категорию самой себе называется **эндофунктор**

Пример

- Произведение множеств – эндофунктор над множествами
- Эндофунктор произведения множеств Р : $\mathbf{Set} \rightarrow \mathbf{Set}$ присваивает каждому множеству A новое множество из всех подмножеств A, иначе:

$$\mathrm{P_O}(A) = \{X \mid X \subseteq A\} \qquad (3)$$

- А для каждого морфизма $f : A \rightarrow B$ – морфизм $\mathrm{P_M}(f) : \mathrm{P_O}(A) \rightarrow \mathrm{P_O}(B)$ такой что, для всех $X \in \mathrm{P_O}(A)$

$$\mathrm{P_M}(f)(X) = \{f(x) \mid x \in X\} \qquad (4)$$

- Покажем:
  - *Первое правило из определения функтора* *(1)*
  - *Второе правило из определение функтора* *(2)*

Доказательство примера - 1

- Доказательство 1го свойства

Pм(id$A$)(X)

= {id$A$(x) | $x \in X$} (по (4) для f = id$A$)

= {x | $x \in X$}

= X (по определению категории)

= idPo(A)(X) (по определению категории для A = Po(A) и $x = X$)

- Доказательство 2го свойства

PM($g \circ f$)(X)

= {($g \circ f$)(x) | $x \in X$} (по (4) для $f = g \circ f$)

= {g(f(x)) | $x \in X$} (по определению категории

= PM(g)({f(x) | $x \in X$}) (по (4) для $f = g$ и X = {f(x) | $x \in X$})

= PM(g)(PM(f)(X)) (по (4))

= (PM(g) $\circ$ PM(f))(X) (по определению для f = Pм(f), g = Pм(g) и $x = X$)

Функторы в Scala

- Scalaz – набор «чистых» функциональных структур (функторы, монады и.т.д)
- https://github.com/scalaz/scalaz
- libraryDependencies += "org.scalaz" %% "scalaz-core" % "7.2.7«
- trait X[T] {

```
  def map[R](f: T => R): X[R]                     - ковариантный функтор
  def contramap[R](f: R => T): X[R]               - контр-вариантный функтор
  def xmap[R](f: (T => R, R => T)): X[R]  - инвариантный(экспоненциальный) ф-р
  def apply[R](f: X[T => R]): X[R]                - аппликативный ф-р
  def flatMap[R](f: T => X[R]): X[R]              - монада
  def coflatMap[R](f: X[T] => R): X[R]            - комонада
  }
```

Пример - Option

- object Test extends App {

```
    val k: Option[Int] = Map("A" -> 0, "B" -> 1).get("C")
    val s: Option[String] = s map toHexString
}
```

- List – рассмотреть самостоятельно

Естественные преобразования

- Освежим в памяти основные положения
  - *Категории состоят из объектов и морфизм*
  - *Морфизмы описывают преобразования объектов в рамках катеории*
  - *Функторы описывают преобразования (морфизмы) категорий*
- Естественные преобразования описывают морфизмы функторов
- Пусть F и G : $\mathcal{C} \longrightarrow \mathcal{D}$ функторы для двух категорий $\mathcal{C}$ и $\mathcal{D}$. Естественное преобразование

$$\tau : \mathrm{F} \longrightarrow \mathrm{G} : \mathcal{C} \longrightarrow \mathcal{D}$$

сопоставляет каждому объекту a из $\mathcal{C}$ морфизм τa : $\mathrm{Fo}(a) \longrightarrow \mathrm{Go}(a)$ из $\mathcal{D}$, называемый компонентом естественного преобразования, так что для всех морфизмов $f : a \longrightarrow b$ in $\mathcal{C}$ выполняется:

$$\tau b \circ \mathrm{FM}(f) = \mathrm{GM}(f) \circ \tau a,$$

Свойство естественных преобразований

$$
\begin{tikzcd}
\mathsf{F}_{\mathrm{O}}(a) \arrow[r, "\tau_a"] \arrow[d, "\mathsf{F}_{\mathrm{M}}(f)"'] & \mathsf{G}_{\mathrm{O}}(a) \arrow[d, "\mathsf{G}_{\mathrm{M}}(f)"] \\
\mathsf{F}_{\mathrm{O}}(b) \arrow[r, "\tau_b"'] & \mathsf{G}_{\mathrm{O}}(b)
\end{tikzcd}
$$

Пример

- Для категории $\mathcal{C}$, тождественное естественное преобразование

$$\mathrm{id} : \mathsf{I} \to \mathsf{I} : \mathcal{C} \to \mathcal{C}$$

присваивает каждому объекту a тождественный морфизм

$$\mathrm{id}a : a \to a.$$

- Это естественное преобразование из и в- эндофукнтор

$$\begin{tikzcd}
a \arrow[r, "\mathrm{id}_a"] \arrow[d, "f"'] & a \arrow[d, "f"] \\
b \arrow[r, "\mathrm{id}_b"'] & b
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{I}_{\mathrm{O}}(a) \arrow[r, "\mathrm{id}_a"] \arrow[d, "\mathsf{I}_{\mathrm{M}}(f)"'] & \mathsf{I}_{\mathrm{O}}(a) \arrow[d, "\mathsf{I}_{\mathrm{M}}(f)"] \\
\mathsf{I}_{\mathrm{O}}(b) \arrow[r, "\mathrm{id}_b"'] & \mathsf{I}_{\mathrm{O}}(b)
\end{tikzcd}$$

Естественные преобразования в Scala

- headOption
  - *из функтора List в Option*
- lastOption
  - *из функтора List в Option*
- concat
  - *из функтора List в функтор List*
- Для всех этих преобразований выполняется коммутативность (порядок применения функции map не важен)

Монады

- Пусть $\mathcal{C}$ некоторая категория. Монада Т = (Т, η, μ) для $\mathcal{C}$ определяется как эндофунктор Т : $\mathcal{C} \rightarrow \mathcal{C}$ совместно с двумя естественными преобразованиями:
  - $\eta : I \rightarrow T : \mathcal{C} \rightarrow \mathcal{C}$ - *называемое unit(единение)*
  - μ : Т ∘ Т → Т : $\mathcal{C} \rightarrow \mathcal{C}$ - называемое произведением
- Для всех объектов a
  - $\mu a \circ \mu T$о(a) = $\mu a \circ T$м(μa)
  - $\mu a \circ \eta$То(a) = idТо(a)
  - μa ∘ Тм(ηa) = idТо(a)
- Так как η и μ являются естественными преобразованиями
  - ηТо$(b) \circ f$ = Тм$(f) \circ \eta a$
  - μТо(b) ∘ Тм(Тм(f)) = Тм$(f) \circ \mu a$

Свойства монад

- Ассоциативность

$$
\begin{tikzcd}
\mathsf{T}_{\mathrm{O}}(\mathsf{T}_{\mathrm{O}}(\mathsf{T}_{\mathrm{O}}(a))) \arrow[r, "\mathsf{T}_{\mathrm{M}}(\mu_a)"] \arrow[d, "\mu_{\mathsf{T}_{\mathrm{O}}(a)}"'] & \mathsf{T}_{\mathrm{O}}(\mathsf{T}_{\mathrm{O}}(a)) \arrow[d, "\mu_a"] \\
\mathsf{T}_{\mathrm{O}}(\mathsf{T}_{\mathrm{O}}(a)) \arrow[r, "\mu_a"'] & \mathsf{T}_{\mathrm{O}}(a)
\end{tikzcd}
$$

- Объединение

$$
\begin{tikzcd}
\mathsf{I}_{\mathrm{O}}(\mathsf{T}_{\mathrm{O}}(a)) \arrow[r, "\eta_{\mathsf{T}_{\mathrm{O}}(a)}"] \arrow[d, equal] & \mathsf{T}_{\mathrm{O}}(\mathsf{T}_{\mathrm{O}}(a)) \arrow[d, "\mu_a"] & \mathsf{T}_{\mathrm{O}}(\mathsf{I}_{\mathrm{O}}(a)) \arrow[l, "\mathsf{T}_{\mathrm{M}}(\eta_a)"'] \arrow[d, equal] \\
\mathsf{T}_{\mathrm{O}}(a) \arrow[r, "\mathrm{id}_{\mathsf{T}_{\mathrm{O}}(a)}"'] & \mathsf{T}_{\mathrm{O}}(a) & \mathsf{T}_{\mathrm{O}}(a) \arrow[l, "\mathrm{id}_{\mathsf{T}_{\mathrm{O}}(a)}"]
\end{tikzcd}
$$

- «Натуральность» преобразований

$$
\begin{tikzcd}
a \arrow[r, "\eta_a"] \arrow[d, "f"'] & \mathsf{T}_{\mathrm{O}}(a) \arrow[d, "\mathsf{T}_{\mathrm{M}}(f)"] \\
\mathsf{T}_{\mathrm{O}}(b) \arrow[r, "\eta_{\mathsf{T}_{\mathrm{O}}(b)}"'] & \mathsf{T}_{\mathrm{O}}(\mathsf{T}_{\mathrm{O}}(b))
\end{tikzcd}
$$

Тройки Клейсли

- Пусть $\mathcal{C}$ – некоторая категория. Тройка Клейсли

$$T = (T_0, \eta, *) \text{ для } \mathcal{C},$$

  - *где η – трансформация, сопоставляющая каждому a объект $T_0(a)$*
  - *Для каждого морфизма $f : a \longrightarrow T_0(b)$ – морфизм $f* : T_0(a) \longrightarrow T_0(b)$ такой что для всех морфизмов $f : a \longrightarrow T_0(b)$ and $g : b \longrightarrow T_0(c)$:*

$$g* \circ f* = (g* \circ f)*$$

- Тройки Клейсли являются альтернативным определением монад
  - *Доказательство остается за кадром (по причине его объемности)*
  - *Если кто-то заинтересуется – я могу выслать*

Монады в Scala

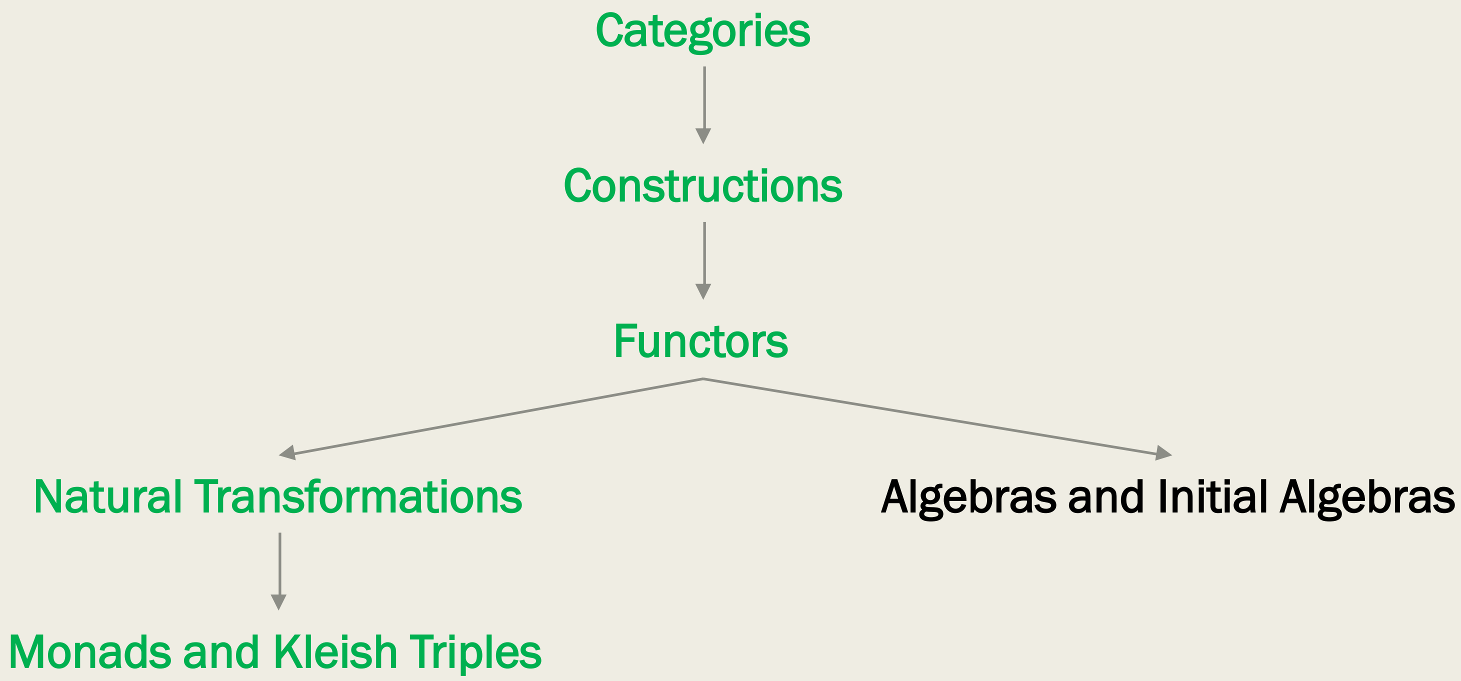
- trait Monad[T] {

  def flatMap[U](f: T => Monad[U]): Monad[U]

  }

  def unit[T](x: T): Monad[T]
- Unit – отвечает за создание монады
- Для уже изученных типов:
  - *Option – Some(x)*
  - *List – List(x)*
  - *Try – Success(x)*
- Законы
  - Left unit law (unit(x) flatMap f == f(x))
  - Right unit law (monad flatMap unit == monad)
  - Associativity law (monad flatMap f) flatMap g == monad flatMap(x => f(x) flatMap g)

Спасибо за внимание



Спасибо за внимание

- Дальнейшие план – знакомство со Scala на реальном проекте
  - Spray framework (Akka HTTP)
  - MongoDB
  - Web UI
  - Scatter plot
  - Clusterization