Теоретико-категорная семантика модальной теории типов, основанной на интуиционистской эпистемической логике

1 Приложение А. Глоссарий по теории категорий.

Определение 1. *Категория $\mathcal{C}$ состоит из:*

- *Класса объектов $Ob_{\mathcal{C}}$;*
- *Для любых объекта $A, B \in Ob_{\mathcal{C}}$ определено множество стрелок (или морфизмов) из A в B $Hom_{\mathcal{C}}(A, B)$;*
- *Если $f \in Hom_{\mathcal{C}}(A, B)$ и $g \in Hom_{\mathcal{C}}(B, C)$, то $g \circ f \in Hom_{\mathcal{C}}(A, C)$;*
- *Для любого объекта $A \in Ob_{\mathcal{C}}$, определен тождественный морфизм $id_A \in Hom_{\mathcal{C}}(A, A)$;*
- *Для любой стрелки $f \in Hom_{\mathcal{C}}(A, B)$, для любой стрелки $g \in Hom_{\mathcal{C}}(B, C)$ и для любой стрелки $h \in Hom_{\mathcal{C}}(C, D)$, $h \circ (g \circ f) = (h \circ g) \circ f$.*
- *Для любой стрелки $f \in Hom_{\mathcal{C}}(A, B)$, $f \circ id_A = f$ и $id_B \circ f = f$.*

Определение 2. *Функтор*

Пусть $\mathcal{C}, \mathcal{D}$ – категории. Функтором называется отображение $F : \mathcal{C} \to \mathcal{D}$, такое, что:

- *$F : A \mapsto FA$, где $A \in Ob_{\mathcal{C}}$;*
- *$F(g \circ f) = F(g) \circ F(f)$;*
- *$F(id_A) = id_{FA}$.*

Определение 3. *Естественное преобразование Пусть $\mathcal{F}, \mathcal{G} : \mathcal{C} \to \mathcal{D}$ – функторы. Естественным преобразованием $\alpha : \mathcal{F} \Rightarrow \mathcal{G}$ называется такое индексированное семейство стрелок $(\alpha_X)_{X \in Ob_{\mathcal{C}}}$, что для любых $A, B \in Ob_{\mathcal{C}}$, для любой стрелки $f \in Hom_{\mathcal{C}}(A, B)$, диаграмма коммутирует:*

$$\begin{tikzcd}
\mathcal{F}A \arrow[r, "\mathcal{F}(f)"] \arrow[d, "\alpha_A"'] & \mathcal{F}B \arrow[d, "\alpha_B"] \\
\mathcal{G}A \arrow[r, "\mathcal{G}(f)"'] & \mathcal{G}B
\end{tikzcd}$$

Определение 4. *Моноидальная категория*

Моноидальная категория – это категория $\mathcal{C}$ с дополнительной структурой:

- *Бифунктор* $\otimes : \mathcal{C} \times \mathcal{C} \to C$, *который мы будем называть тензором;*
- *Единица* $\mathbb{1}$*;*
- *Изоморфизм, который мы будем называть ассоциатором:* $\alpha_{A,B,C} : (A \otimes B) \otimes C \cong A \otimes (B \otimes C)$*;*
- *Изоморфизм* $L_A : \mathbb{1} \otimes A \cong A$*;*
- *Изоморфизм* $R_A : A \otimes \mathbb{1} \cong A$*;*
- *Первое условие когерентности (пятиугольник Маклейна) (данная диаграмма коммутирует):*

$$\begin{tikzcd}
 & (A \otimes (B \otimes C)) \otimes D \arrow[dr, "\alpha_{A,B\otimes C,D}"] & \\
((A \otimes B) \otimes C) \otimes D \arrow[ur, "\alpha_{A,B,C}\otimes id_D"] \arrow[d, "\alpha_{A\otimes B,C,D}"'] & & A \otimes ((B \otimes C) \otimes D) \arrow[d, "id_A\otimes\alpha_{B,C,D}"] \\
(A \otimes B) \otimes (C \otimes D) \arrow[rr, "\alpha_{A,B,C\otimes D}"'] & & A \otimes (B \otimes (C \otimes D))
\end{tikzcd}$$

- *Второе условие когерентности (тождество треугольника):*

$$\begin{tikzcd}
(A \otimes \mathbb{1}) \otimes B \arrow[rr, "\alpha_{A,\mathbb{1},B}"] \arrow[dr, "R_A\otimes id_B"'] & & A \otimes (\mathbb{1} \otimes B) \arrow[dl, "id_A\otimes L_B"] \\
 & A \otimes B &
\end{tikzcd}$$

- *Моноидальная категория $\mathcal{C}$ называется симметрической, если для любых* $A, B \in Ob_{\mathcal{C}}$, *имеет место изоморфизм* $\sigma_{A,B} : A \otimes B \cong B \otimes A$.

Определение 5. *Декартово замкнутая категория*

Декартово замкнутая категория – это категория с терминальным объектом, конечными произведениями и экспоненцированием.

Легко видеть, что декартово замкнутая категория – это частный случай (симметрической) моноидальной категории, в котором тензор – это произведения, а единица – это терминальный объект.

Определение 6. *Нестрогий моноидальный функтор*

Пусть $\langle \mathcal{C}, \otimes_1, \mathbb{1}_{\mathcal{C}} \rangle$ *и* $\langle \mathcal{D}, \otimes_2, \mathbb{1}_{\mathcal{D}} \rangle$ *моноидальные категории.*

Нестрогий моноидальный функтор $\mathcal{F} : \langle \mathcal{C}, \otimes_1, \mathbb{1} \rangle \to \langle \mathcal{D}, \otimes_2, \mathbb{1}' \rangle$ *это функтор* $\mathcal{F} : \mathcal{C} \to \mathcal{D}$ *с дополнительными естественными преобразованиями:*

- $u : \mathbb{1}_{\mathcal{D}} \to \mathcal{F}\mathbb{1}_{\mathcal{C}}$*;*
- $*_{A,B} : \mathcal{F}A \otimes_{\mathcal{D}} \mathcal{F}B \to \mathcal{F}(A \otimes_{\mathcal{C}} B)$.

и условиями когерентности:

- *Ассоциативность:*

$$\begin{tikzcd}
(\mathcal{F}A \otimes_{\mathcal{D}} \mathcal{F}B) \otimes_{\mathcal{D}} \mathcal{F}C \arrow[r, "\alpha^{\mathcal{D}}_{\mathcal{F}A,\mathcal{F}B,\mathcal{F}C}"] \arrow[d, "*_{A,B} \otimes_{\mathcal{D}} id_{\mathcal{F}B}"'] & \mathcal{F}A \otimes_{\mathcal{D}} (\mathcal{F}B \otimes_{\mathcal{D}} \mathcal{F}C) \arrow[d, "id_{\mathcal{F}A} \otimes_{\mathcal{D}} *_{B,C}"] \\
\mathcal{F}(A \otimes_{\mathcal{C}} B) \otimes_{\mathcal{D}} \mathcal{C} \arrow[d, "*_{A \otimes_{\mathcal{C}} B, C}"'] & \mathcal{F}A \otimes_{\mathcal{D}} \mathcal{F}(B \otimes_{\mathcal{C}} C) \arrow[d, "*_{A, B \otimes_{\mathcal{C}} C}"] \\
\mathcal{F}((A \otimes_{\mathcal{C}} B) \otimes_{\mathcal{C}} C) \arrow[r, "\mathcal{F}(\alpha^{\mathcal{C}}_{A,B,C})"'] & \mathcal{F}(A \otimes_{\mathcal{C}} (B \otimes_{\mathcal{C}} C))
\end{tikzcd}$$

- *Свойство левой единицы:*

$$\begin{tikzcd}
\mathbb{1}_{\mathcal{D}} \otimes_{\mathcal{D}} \mathcal{F}A \arrow[r, "u \otimes_{\mathcal{D}} id_{\mathcal{F}A}"] \arrow[d, "L^{\mathcal{D}}_{\mathcal{F}A}"'] & \mathcal{F}\mathbb{1}_{\mathcal{C}} \otimes_{\mathcal{D}} \mathcal{F}A \arrow[d, "*_{\mathbb{1}_{\mathcal{C}}, A}"] \\
\mathcal{F}A & \mathcal{F}(\mathbb{1}_{\mathcal{C}} \otimes_{\mathcal{C}} A) \arrow[l, "\mathcal{F}(L^{\mathcal{C}}_{A})"']
\end{tikzcd}$$

- *Свойство правой единицы:*

$$\begin{tikzcd}
\mathcal{F}A \otimes_{\mathcal{D}} \mathbb{1}_{\mathcal{D}} \arrow[r, "id_{\mathcal{F}A} \otimes_{\mathcal{D}} u"] \arrow[d, "R^{\mathcal{D}}_{\mathcal{F}A}"'] & \mathcal{F}A \otimes_{\mathcal{D}} \mathcal{F}\mathbb{1}_{\mathcal{C}} \arrow[d, "*_{A, \mathbb{1}_{\mathcal{C}}}"] \\
\mathcal{F}A & \mathcal{F}(A \otimes_{\mathcal{C}} \mathbb{1}_{\mathcal{C}}) \arrow[l, "\mathcal{F}(R^{\mathcal{C}}_{A})"]
\end{tikzcd}$$

Определение 7. *Тензорно-сильный функтор – это эндофунктор над моноидальной категорией с дополнительным естественным преобразованием и условиями когерентности для него (ниже соответствующие коммутирующие диаграммы):*

$$\tau_{A,B} : A \otimes \mathcal{K}B \to \mathcal{K}(A \otimes B)$$

$$\begin{tikzcd}
(A \otimes B) \otimes \mathcal{K}C \arrow[rr, "\tau_{A \otimes B, C}"] \arrow[d, "\alpha_{A,B,\mathcal{K}C}"'] & & \mathcal{K}((A \otimes B) \otimes C) \arrow[d, "\mathcal{K}(\alpha_{A,B,C})"] \\
A \otimes (B \otimes \mathcal{K}C) \arrow[r, "id_A \otimes \tau_{B,C}"'] & A \otimes \mathcal{K}(B \otimes C) \arrow[r, "\tau_{A,(B \otimes C)}"'] & \mathcal{K}(A \otimes (B \otimes C))
\end{tikzcd}$$

$$\begin{tikzcd}
\mathbb{1} \otimes \mathcal{K}A \arrow[r, "\mu_{\mathbb{1},A}"] \arrow[dr, "R_{\mathcal{K}A}"'] & \mathcal{K}(\mathbb{1} \otimes A) \arrow[d, "\mathcal{K}(R_A)"] \\
& \mathcal{K}A
\end{tikzcd}$$

Определение 8. *Аппликативный функтор*

Аппликативный функтор – это тройка $\langle \mathcal{C}, \mathcal{K}, \eta \rangle$, где $\mathcal{C}$ – это моноидальная категория, $\mathcal{K}$ - это тензорно-сильный нестрогий моноидальный эндофунктор и $\eta : Id_{\mathcal{C}} \Rightarrow \mathcal{K}$ – это естественное преобразование, такое, что:

- $u = \eta_{\mathbb{1}}$;

- $*_{A,B} \circ (\eta_A \otimes \eta_B) = \eta_{A \otimes B}$, *то есть диаграмма коммутирует:*

$$\begin{tikzcd} A \otimes B \arrow[r, "\eta_A \otimes \eta_B"] \arrow[dr, "\eta_{A\otimes B}"'] & \mathcal{K}A \otimes \mathcal{K}B \arrow[d, "*_{A,B}"] \\ & \mathcal{K}(A \otimes B) \end{tikzcd}$$

- $\tau_{A,B} = *_{A,B} \circ \eta_A \otimes id_{\mathcal{K}B}$.

По умолчанию мы будем рассматривать ниже аппликативный функтор над декартово замкнутой категорией.

2 Глоссарий по основным конструкциям функционального языка программирования Haskell: функторы, монады, аппликативные функторы

Определение 9. *Класс типов*

Классом типов в языке Haskell – это реализация некоторого общего интерфейса для совокупности типов.

Представителем (или наследников) класса типов называется реализация данного класса для конкретного типа.

Определение 10. *Функтор*

Функтор – это однопараметрический класс типов, позволяющий пронести действие функции через значения, полученные в результате применения к их типу одноместного типового оператора.

Определение в стандартной библиотеке выглядит следующим образом:

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

Рассмотрим примеры:

- Список (неограниченная в длине последовательность) является функтором: Определение в стандартной библиотеке выглядит следующим образом:

  ```
  instance Functor [] where
    fmap :: (a -> b) -> [a] -> [b]
    fmap f [] = []
    fmap f (x:xs) = (f x) : (fmap f xs)
  ```

  Данный пример достаточно прост: реализация функтора для списка – это функция высшего порядка, которая, принимая на входе одноместную функцию из типа a в тип b и список элементов типа a, возвращает список элементов типа b, который получен применением функции к каждому элементу списка, полученного на вход.

- Пара (тип декартова произведения типов) также функтор:

```haskell
instance Functor (b,) where
  fmap :: (a -> c) -> (b,a) -> (b,c)
  fmap f (x,y) = (x, f y)
```

Конструктор пары является двухпараметрическим типовым оператором, но мы сделали из него однопараметрический оператор фиксацией первого параметра.

Данная реализация также довольно проста: на вход принимается функция из типа a в тип c и кортеж, в котором первая координата имеет тип b, а вторая – тип a. На выходе мы получаем кортеж типа (b, c), применяя полученную на вход функцию ко второй координате пары.

- Тип $Maybe$ – это однопараметрический типовой оператор, для обработки неопределенных значений:

  ```haskell
  data Maybe a = Nothing | Just a
  ```

  Реализация функтора для типа $Maybe$:

  ```haskell
  instance Functor Maybe where
    fmap :: (a -> b) -> Maybe a -> Maybe b
    fmap f Nothing = Nothing
    fmap f (Just x) = Just (f x)
  ```

  Если второй аргумент является неопределенным значением (на вход передан $Nothing$), то и возвращается $Nothing$. Если же значение определено, то есть оно имеет вид $Just\, x$, тогда мы применяем функцию функцию к x, а результат вычисления оборачиваем в конструктор $Just$.