Лямбда-исчисление и теория типов

вводная лекция

Ламберов Лев Дмитриевич
lev.lamberov@urfu.ru

Уральский федеральный университет

29 ноября 2016 г.

Немного истории

- Пеано, 1889
- Фреге, 1891, 1893
- Рассел, 1903-1905
- Рассел, Уайтхед, 1910-1913

Комбинаторная логика

Шейнфинкель, 1920 (1924):

- избавление от связанных переменных
- базовые комбинаторы
- достаточно комбинаторов S и K
- каррирование (шейнфинкелирование/фрегеирование?)

Карри, 1926, 1927, 1930:

- изучение понятия подстановки
- применение любого комбинатора к любому комбинатору

Как выглядит комбинаторная логика?

Комбинаторы:

- $\mathsf{I}x = x$
- $(\mathsf{K}\,x)\,y = x$
- $\mathsf{S}\,x\,y\,z = (x\,z\,(y\,z))$

Как выглядит комбинаторная логика?

Комбинаторы:

- $\mathsf{I}x = x$
- $(\mathsf{K}\,x)\,y = x$
- $\mathsf{S}\,x\,y\,z = (x\,z\,(y\,z))$

Пример:
$((\mathsf{S}\,\mathsf{K}\,\mathsf{K})\,x)$

Как выглядит комбинаторная логика?

Комбинаторы:

- $\mathbf{I}x = x$
- $(\mathbf{K}\,x)\,y = x$
- $\mathbf{S}\,x\,y\,z = (x\,z\,(y\,z))$

Пример:
$((\mathbf{S}\,\mathbf{K}\,\mathbf{K})\,x)$
$= (\mathbf{S}\,\mathbf{K}\,\mathbf{K}\,x)$

Как выглядит комбинаторная логика?

Комбинаторы:

- $\mathbf{I}x = x$
- $(\mathbf{K}\,x)\,y = x$
- $\mathbf{S}\,x\,y\,z = (x\,z\,(y\,z))$

Пример:

$((\mathbf{S}\,\mathbf{K}\,\mathbf{K})\,x)$
$= (\mathbf{S}\,\mathbf{K}\,\mathbf{K}\,x)$
$= (\mathbf{K}\,x\,(\mathbf{K}\,x))$

Как выглядит комбинаторная логика?

Комбинаторы:

- $\mathbf{I}x = x$
- $(\mathbf{K}\,x)\,y = x$
- $\mathbf{S}\,x\,y\,z = (x\,z\,(y\,z))$

Пример:

$$((\mathbf{S}\,\mathbf{K}\,\mathbf{K})\,x)$$
$$= (\mathbf{S}\,\mathbf{K}\,\mathbf{K}\,x)$$
$$= (\mathbf{K}\,x\,(\mathbf{K}\,x))$$
$$= x$$

Как выглядит комбинаторная логика?

Комбинаторы:

- $\mathbf{I}x = x$
- $(\mathbf{K}\,x)\,y = x$
- $\mathbf{S}\,x\,y\,z = (x\,z\,(y\,z))$

Пример:

$((\mathbf{S}\,\mathbf{K}\,\mathbf{K})\,x)$
$= (\mathbf{S}\,\mathbf{K}\,\mathbf{K}\,x)$
$= (\mathbf{K}\,x\,(\mathbf{K}\,x))$
$= x$

* на самом деле, комбинаторы — это лямбда-термы, не имеющие свободных переменных

Лямбда-исчисление

Чёрч, 1928 (1932)

- построение более «естественной» системы, чем теория типов Рассела и теория множеств Цермело
- изначальный вариант представляет собой бестиповую логику с неограниченной квантификацией и без закона исключённого третьего
- . . . борьба с противоречиями
- эквивалентность рекурсивным функциям Эрбрана-Гёделя и Тьюринг-вычислимым функциям
- тезис Чёрча

Лямбда-исчисление и теория типов

Чёрч, 1940

- отход от идеи формулировки общей бестиповой логики для оснований математики
- формулировка теории простых типов на основе лямбда-исчисления

Карри vs. Чёрч

Рассел-Чёрч:

- значение приписывается только правильно типизированным термам
- типы ограничивают язык, запрещая проблемные выражения

Карри:

- значение присваивается термам независимо от их типов
- разрешено использование любого комбинатора, но последние получают «метки»

Соответствие Карри-Говарда

Карри, 1930-е
Говард, 1969 (1980)

- изоморфизм между типами и формулами (с импликацией) в интуиционистской логике
- изоморфизм между выводом типов и выводом высказываний

Лямбда-исчисление: синтаксис

Термы $t :=$

- x — переменная
- $\lambda x.t$ — абстракция
- $t_1\ t_2$ — применение

Обычные конвенции

- применение лево-ассоциативно, т. е. $s\ t\ u = (s\ t)\ u$
- абстракция простирается направо как можно дальше, т. е. $\lambda x.\lambda y.x\ y\ x = \lambda x.(\lambda y.(x\ y\ x))$

Связывание

- Переменная связана, если она находится в теле абстракции.
- Переменная свободна, если она не связывается вышележащей абстракцией.
- Терм без свободных переменных называется замкнутым (или комбинатором)

Лямбда-исчисление: операционная семантика

Каждый шаг вычисления состоит в том, что в терме-применении, в котором левый член является абстракцией, связанная переменная в теле этой абстракции заменяется на правый член.

$$(\lambda x.t_1)\, t_2 \longrightarrow [x \mapsto t_2]\, t_1$$

Пример: $(\lambda x.x)\, y \longrightarrow y$

Лямбда-исчисление: операционная семантика

Каждый шаг вычисления состоит в том, что в терме-применении, в котором левый член является абстракцией, связанная переменная в теле этой абстракции заменяется на правый член.

$(\lambda x.t_1)\, t_2 \longrightarrow [x \mapsto t_2]\, t_1$

Пример: $(\lambda x.x)\, y \longrightarrow y$

Стратегии вычисления:

- полная β-редукция
- нормальный порядок вычислений
- вызов по имени (Algol 60, Scala)
- вызов по значению (C, Common Lisp, Java, Scheme)
- ...

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$tst\ tru\ 1\ 2$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$tst\ tru\ 1\ 2$$
$$= \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
&tst\ tru\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
&\longrightarrow \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.tru\ 1\ z)\ 2}
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.tru\ 1\ z)\ 2} \\
\longrightarrow & tru\ 1\ 2
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.tru\ 1\ z)\ 2} \\
\longrightarrow & tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.x)\ 1}\ 2
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.tru\ 1\ z)\ 2} \\
\longrightarrow & tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.x)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda y.1)\ 2}
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.tru\ 1\ z)\ 2} \\
\longrightarrow & tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.x)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda y.1)\ 2} \\
\longrightarrow & 1
\end{aligned}
$$

$tst\ fls\ 1\ 2$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
&\longrightarrow \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
&\longrightarrow \underline{(\lambda z.tru\ 1\ z)\ 2} \\
&\longrightarrow tru\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.x)\ 1}\ 2 \\
&\longrightarrow \underline{(\lambda y.1)\ 2} \\
&\longrightarrow 1
\end{aligned}
$$

$$
\begin{aligned}
& tst\ fls\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ fls}\ 1\ 2
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.tru\ 1\ z)\ 2} \\
\longrightarrow & tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.x)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda y.1)\ 2} \\
\longrightarrow & 1
\end{aligned}
$$

$$
\begin{aligned}
& tst\ fls\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ fls}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.fls\ y\ z)\ 1}\ 2
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.tru\ 1\ z)\ 2} \\
\longrightarrow & tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.x)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda y.1)\ 2} \\
\longrightarrow & 1
\end{aligned}
$$

$$
\begin{aligned}
& tst\ fls\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ fls}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.fls\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.fls\ 1\ z)\ 2}
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.tru\ 1\ z)\ 2} \\
\longrightarrow & tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.x)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda y.1)\ 2} \\
\longrightarrow & 1
\end{aligned}
$$

$$
\begin{aligned}
& tst\ fls\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ fls}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.fls\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.fls\ 1\ z)\ 2} \\
\longrightarrow & fls\ 1\ 2
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
&tst\ tru\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
&\longrightarrow \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
&\longrightarrow \underline{(\lambda z.tru\ 1\ z)\ 2} \\
&\longrightarrow tru\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.x)\ 1}\ 2 \\
&\longrightarrow \underline{(\lambda y.1)\ 2} \\
&\longrightarrow 1
\end{aligned}
$$

$$
\begin{aligned}
&tst\ fls\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ fls}\ 1\ 2 \\
&\longrightarrow \underline{(\lambda y.\lambda z.fls\ y\ z)\ 1}\ 2 \\
&\longrightarrow \underline{(\lambda z.fls\ 1\ z)\ 2} \\
&\longrightarrow fls\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.y)\ 1}\ 2
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
& tst\ tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.tru\ 1\ z)\ 2} \\
\longrightarrow & tru\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.x)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda y.1)\ 2} \\
\longrightarrow & 1
\end{aligned}
$$

$$
\begin{aligned}
& tst\ fls\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ fls}\ 1\ 2 \\
\longrightarrow & \underline{(\lambda y.\lambda z.fls\ y\ z)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda z.fls\ 1\ z)\ 2} \\
\longrightarrow & fls\ 1\ 2 \\
= & \underline{(\lambda x.\lambda y.y)\ 1}\ 2 \\
\longrightarrow & \underline{(\lambda y.y)\ 2}
\end{aligned}
$$

Программирование на лямбда-исчислении: логика

Введём булевы константы:

- $tru = \lambda x.\lambda y.x$
- $fls = \lambda x.\lambda y.y$

Определим проверку «истинностного значения»:

- $tst = \lambda x.\lambda y.\lambda z.x\ y\ z$

Пример:

$$
\begin{aligned}
&tst\ tru\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ tru}\ 1\ 2 \\
&\longrightarrow \underline{(\lambda y.\lambda z.tru\ y\ z)\ 1}\ 2 \\
&\longrightarrow \underline{(\lambda z.tru\ 1\ z)\ 2} \\
&\longrightarrow tru\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.x)\ 1}\ 2 \\
&\longrightarrow \underline{(\lambda y.1)\ 2} \\
&\longrightarrow 1
\end{aligned}
$$

$$
\begin{aligned}
&tst\ fls\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.\lambda z.x\ y\ z)\ fls}\ 1\ 2 \\
&\longrightarrow \underline{(\lambda y.\lambda z.fls\ y\ z)\ 1}\ 2 \\
&\longrightarrow \underline{(\lambda z.fls\ 1\ z)\ 2} \\
&\longrightarrow fls\ 1\ 2 \\
&= \underline{(\lambda x.\lambda y.y)\ 1}\ 2 \\
&\longrightarrow \underline{(\lambda y.y)\ 2} \\
&\longrightarrow 2
\end{aligned}
$$

Программирование на лямбда-исчислении: арифметика

Числа Чёрча: каждое число n представляет собой комбинатор c_n, который принимает два аргумента, s (функция «следующий») и z («ноль»), и n раз применяет s к z:

- $c_0 = \lambda s.\lambda z.z$
- $c_1 = \lambda s.\lambda z.s\,z$
- $c_2 = \lambda s.\lambda z.s\,(s\,z)$
- ...

Программирование на лямбда-исчислении: арифметика

Числа Чёрча: каждое число n представляет собой комбинатор c_n, который принимает два аргумента, s (функция «следующий») и z («ноль»), и n раз применяет s к z:

- $c_0 = \lambda s.\lambda z.z$
- $c_1 = \lambda s.\lambda z.s\,z$
- $c_2 = \lambda s.\lambda z.s\,(s\,z)$
- ...

Арифметические операции:

$plus = \lambda m.\lambda n.\lambda s.\lambda z.m\,s\,(n\,s\,z)$
$times = \lambda m.\lambda n.\lambda s.m\,(n\,s)$

Программирование на лямбда-исчислении: арифметика

Попробуйте вычислить 2+1 и 1+2:

$$plus\ c_1\ c_2 = (\lambda m.\lambda n.\lambda s.\lambda z.m\ s\ (n\ s\ z))\ (\lambda s.\lambda z.s\ z)\ (\lambda s.\lambda z.s\ (s\ z))$$
$$plus\ c_2\ c_1 = (\lambda m.\lambda n.\lambda s.\lambda z.m\ s\ (n\ s\ z))\ (\lambda s.\lambda z.s\ (s\ z))\ (\lambda s.\lambda z.s\ z)$$

Программирование на лямбда-исчислении: арифметика

Попробуйте вычислить 2+1 и 1+2:

$$plus\ c_1\ c_2 = (\lambda m.\lambda n.\lambda s.\lambda z.m\ s\ (n\ s\ z))\ (\lambda s.\lambda z.s\ z)\ (\lambda s.\lambda z.s\ (s\ z))$$
$$plus\ c_2\ c_1 = (\lambda m.\lambda n.\lambda s.\lambda z.m\ s\ (n\ s\ z))\ (\lambda s.\lambda z.s\ (s\ z))\ (\lambda s.\lambda z.s\ z)$$

Даже при выборе одинаковой стратегии вычисления *количество шагов будет разным*. Дело в том, что в лямбда-исчислении тело абстракции может рассматриваться как описание описание операции, которую нужно произвести, имея некоторый аргумент. Два терма могут «вести» себя одинаково (иметь одинаковое значение при одинаковых аргументах), но будут требовать разных вычислительных ресурсов. Связано это с тем, что лямбда-исчисление интенсионально (но можно добавить к нему что-то типа принципа экстенсиональности).

Простые типы

Базовым типом является тип функций: $T \to T$.

Пример 1 (комбинатор **I**): $\lambda x.x \ : \ A \to A$

Простые типы

Базовым типом является тип функций: $T \to T$.

Пример 1 (комбинатор **I**): $\lambda x.x \ : \ A \to A$

Пример 2 (комбинатор **K**): $\lambda x.\lambda y.x \ : \ A \to (B \to A)$

Простые типы

Базовым типом является тип функций: $T \to T$.

Пример 1 (комбинатор **I**): $\lambda x.x \ : \ A \to A$

Пример 2 (комбинатор **K**): $\lambda x.\lambda y.x \ : \ A \to (B \to A)$

Пример 3 (комбинатор **S**):
$\lambda x.\lambda y.\lambda z.x\,z\,(y\,z) \ : \ (A \to B) \to ((A \to (B \to C)) \to (A \to C))$

Простые типы

Базовым типом является тип функций: $T \to T$.

Пример 1 (комбинатор **I**): $\lambda x.x \ : \ A \to A$

Пример 2 (комбинатор **K**): $\lambda x.\lambda y.x \ : \ A \to (B \to A)$

Пример 3 (комбинатор **S**):
$\lambda x.\lambda y.\lambda z.x\,z\,(y\,z) \ : \ (A \to B) \to ((A \to (B \to C)) \to (A \to C))$

В число простых типов для удобства можно добавить, например, типы *Bool* и *Nat*.

Правила типизации

- $\dfrac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x{:}T_1.t_2 : T_1 \to T_2}$
- $\dfrac{x : T \in \Gamma}{\Gamma \vdash x : T}$
- $\dfrac{\Gamma \vdash t_1 : T_1 \to T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : T_2}$

- правило типизации для абстракций
- правило типизации для переменных
- правило типизации для применений

Из экземпляров правил типизации можно строить деревья вывода типов.

Некоторые свойства

- Лемма о порождении: из утверждения о типе позволяет получать доказательство этого утверждения
- Теорема о единственности типов: всякий терм имеет не более одного типа
- Теорема о продвижении: хорошо типизированный терм не может быть тупиковым
- Теорема о сохранении: вычисление хорошо типизированного терма даёт другой хорошо типизированный терм

Полиморфизм через подтипы

$S <: T$ (тип S является подтипом типа T)

То есть, всякий терм, имеющий тип S, может безопасно использоваться в контексте, где ожидается терм типа T.

Параметрический полиморфизм

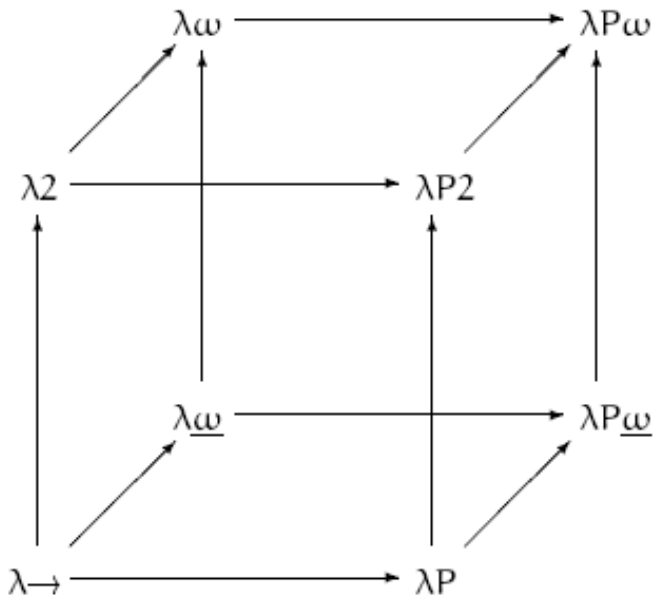
Позволяет одинаково работать со значениями разных типов. Термы зависят от типов.

В этом случае используются типовые переменные (параметры).

$\lambda x : \mathit{Bool}.x$ и $\lambda x : \mathit{Nat}.x$ не полиморфны и работают только со значениями булевого типа (в первом случае) или со значениями типа натуральных чисел (во втором случае).

$\lambda x : A.x$, где A — типовая переменная, работает со значением любого типа. В этом случае указанные выше функции являются результатами подстановки Bool и Nat вместо типовой переменной A.

Лямбда-куб



Возможные зависимости:

1. термы от термов
2. термы от типов, $\lambda 2$
3. типы от типов, $\lambda\omega$
4. типы от термов, λP

Универсумы

- Универсум — это тип, элементами которого являются типы.
- Универсумы организованы в иерархию, $U_0 : U_1 : U_2 : \ldots$
- Универсумы кумулятивны.

Зависимые типы

Функции вида $B : A \to U$ называются семействами типов или зависимыми типами.

Зависимые типы «зависят» от значения (зачастую некоторого другого типа).

Зависимые типы

Функции вида $B : A \to U$ называются семействами типов или зависимыми типами.

Зависимые типы «зависят» от значения (зачастую некоторого другого типа).

Пример 1: тип списков длины 3.

Зависимые типы

Функции вида $B : A \to U$ называются семействами типов или зависимыми типами.

Зависимые типы «зависят» от значения (зачастую некоторого другого типа).

Пример 1: тип списков длины 3.

Пример 2: предикат P, утверждающий что-то о термах типа A представляет собой функцию типа $A \to U$. Эта функция принимает терм $x : A$ и возвращает $P(x) : U$.

Тип зависимых функций

$$\Pi_{(x:A)} \to P(x)$$

- Термы этого типа — функции, которые принимают термы $x : A$ и возвращают терм типа $P(x)$.
- Тип зависимых функций представляет собой обобщение типа функций (в последнем P является константным семейством).
- Соответствует универсальной квантификации: функция, которая порождает сертификат для всякого $x : A$.

Тип зависимых произведений

$$\Sigma_{(x:A)} \to P(x)$$

- Термы этого типа имеют вид (x, p), где $x : A$, а $p : P(x)$.
- Тип зависимых функций представляет собой обобщение типа произведений, $A \times B$, (в последнем p является фиксированным типом).
- Соответствует экзистенциальной квантификации: существует по меньшей мере один x, для которого имеется сертификат p, что $P(x)$.

А дальше гомотопии...

Спасибо за внимание!

P.S. Гомотопия отказалась сделать фотографию для этого слайда.