Теория типов

https://github.com/artemohanjanyan/tt-conspect

Содержание

λ-исчисление

Введение

Смысла в этом нет.

Д.Г.

Определение (λ-выражение)**.** λ-выражение — выражение, удовлетворяющее грамматике:

$$\begin{aligned}\Lambda ::= &\lambda x.\Lambda && \text{(абстракция)}\\ &|\,\Lambda\Lambda && \text{(аппликация)}\\ &|\,x\\ &|\,(\Lambda)\end{aligned}$$

(а) Аппликация левоассоциативна.

(б) Абстракция распространяется как можно дальше вправо.

Пример. $((\lambda z.(z(yz)))(zx)z) = (\lambda z.z(yz))(zx)z$

Есть понятия связанного и свободного вхождения переменной (аналогично исчислению предикатов). $\lambda x.A$ связывает все свободные вхождения x в A.

Определение. Функция $\mathrm{FV}(A)$ — множество свободных переменных, входящих в A:

$$\mathrm{FV}(A) = \begin{cases}\{x\} & \text{если } A \equiv x\\ \mathrm{FV}(P) \cup \mathrm{FV}(Q) & \text{если } A \equiv PQ\\ \mathrm{FV}(P) \setminus \{x\} & \text{если } A \equiv \lambda x.P\end{cases}$$

Договоримся, что:

(а) Переменные — x, a, b, c.

(б) Термы (части λ-выражения) — X, A, B, C.

(в) Фиксированные переменные обозначаются буквами из начала алфавита, метапеременные — из конца.

На самом деле, смысл в этом есть, λ-выражение можно понимать как функцию. Абстракция — это функция с аргументом, аппликация — это передача аргумента.

Определение (α-эквивалентность)**.** A и B называются α-эквивалентными ($A =_\alpha B$), если выполнено одно из следующих условий:

1. $A \equiv x$ и $B \equiv x$.
2. $A \equiv \lambda x.P$, $B \equiv \lambda y.Q$ и $P[x := t] =_\alpha Q[y := t]$, где t — новая переменная.
3. $A \equiv PQ$, $B \equiv RS$ и $P =_\alpha R$, $Q =_\alpha S$.

То есть два выражения α-эквивалентны, если они равны с точностью до переименования абстракций. В C++ между функциями `[](int x){return f(x);}` и `[](int y){return f(y);}` нет разницы.

Пример. $\lambda x.\lambda y.xy =_\alpha \lambda y.\lambda x.yx$.

Доказательство. Согласно второму правилу следующие утверждения верны:

$$\begin{aligned}\lambda y.ty =_\alpha \lambda x.tx &\implies \lambda x.\lambda y.xy =_\alpha \lambda y.\lambda x.yx\\ tz =_\alpha tz &\implies \lambda y.ty =_\alpha \lambda x.tx\end{aligned}$$

$tz =_\alpha tz$ верно по третьему условию. □

Определение (β-редекс)**.** Терм вида $(\lambda a.A)\,B$ называется β-редексом.

Пример. В выражении $(\lambda f.\underbrace{(\lambda x.\overbrace{f(xx)}^{A_1})}_{A_2}\overbrace{(\lambda x.f(xx))}^{B_1})\underbrace{g}_{B_2}$ два β-редекса.

Определение. Множество λ-термов $\mathbf{\Lambda}$ назовём множеством классов эквивалентности Λ по $(=_\alpha)$[1].

В дальнейшем мы будем по умолчанию подразумевать возможность α-эквивалентности, и под выражением из Λ будем иметь в виду любое выражение из соответствующего класса из $\mathbf{\Lambda}$.

Определение (β-редукция)**.** $A \to_\beta B$ (A и B состоят в отношении β-редукции), если выполняется одно из условий:

1. $A \equiv PQ$, $B \equiv RS$ и либо $P \to_\beta R$ и $Q =_\alpha S$, либо $P =_\alpha R$ и $Q \to_\beta S$.
2. $A \equiv \lambda x.P$, $B \equiv \lambda x.Q$, $P \to_\beta Q$.
3. $A \equiv (\lambda x.P)Q$, $B \equiv P[x \coloneqq Q]$, Q свободно для подстановки в P вместо x.

β-редукция — это вычисление функции, подстановка её аргумента.

Пример. $(\lambda x.xy)a \to_\beta ay$.

Пример. $(\lambda x.\lambda y.x)y \to_\beta \lambda y'.y$, так как $(\lambda x.\lambda y.x)y =_\alpha (\lambda x.\lambda y'.x)y$.

Будем обозначать транзитивно-рефлексивное замыкание $(\to_\beta)$ как $(\twoheadrightarrow_\beta)$ и называть его β-редуцируемостью. Также для удобства введём сокращение записи:

$$\lambda xy.A = \lambda x.\lambda y.A$$

Числа Чёрча

Хотите знать, что такое истина?

Д.Г.

$$\mathsf{T} = \lambda x\lambda y.x \qquad \mathsf{F} = \lambda x\lambda y.y$$
$$\mathsf{Not} = \lambda a.a\,\mathsf{F}\,\mathsf{T}$$

Похоже на тип `boolean`, не правда ли?

Пример. Отрицание истины — это ложь:

$$\mathsf{Not}\,\mathsf{T} = (\lambda a.a\,\mathsf{F}\,\mathsf{T})\mathsf{T} \to_\beta \mathsf{T}\,\mathsf{F}\,\mathsf{T} = (\lambda x.\lambda y.x)\mathsf{F}\,\mathsf{T} \to_\beta (\lambda y.\mathsf{F})\mathsf{T} \to_\beta \mathsf{F}$$

Истина — это функция, которая принимает два аргумента и возвращает первый аргумент. Ложь принимает два аргумента и возвращает второй. Если в выражении $a\,\mathsf{F}\,\mathsf{T}$ терм a является истиной, то результатом будет первый аргумент, F. Если ложью — второй аргумент, T.

Можно продолжить:

$$\mathtt{And} = \lambda a.\lambda b.ab\,\mathsf{F} \qquad \mathtt{Or} = \lambda a.\lambda b.a\,\mathsf{T}\,b$$

Разберём $\mathtt{And}$. Он принимает два аргумента. Если первый аргумент это истина, то тогда результат $\mathtt{And}$ равен второму аргументу. Если первый аргумент это ложь, то тогда результат $\mathtt{And}$ это ложь, и от второго аргумента не зависит. Это и записано в выражении. $\mathtt{Or}$ разбирается аналогично.

Попробуем определить числа:

Определение (чёрчевский нумерал)**.**

$$\overline{n} = \lambda f.\lambda x.f^n x, \quad \text{где} \quad f^n x = \begin{cases} f\left(f^{n-1}x\right) & \text{если } n > 0 \\ x & \text{если } n = 0 \end{cases}$$

[1] Если $\circ$ — двухместный оператор, то $(\circ)$ — это обозначение функции $(\circ)\,ab = a \circ b$. $(a\,\circ)$ и $(\circ\,b)$ это «сечения» оператора, передача ему одного из аргументов. Например, $(\div 3)$ — это функция деления на три, а $(2^\wedge)$ — это функция возведения двойки в степень.

Пример.

$$\overline{3} = \lambda f.\lambda x.f\,(f\,(f x))$$

Несложно определить прибавление единицы к такому нумералу:

$$(+1) = \lambda n.\lambda f.\lambda x.f(nfx)$$

Пример.

$$(+1)\overline{1} = (\lambda n.\lambda f.\lambda x.f(nfx))(\lambda f.\lambda x.fx) \to_\beta \lambda f.\lambda x.f((\lambda f.\lambda x.fx)fx) \twoheadrightarrow_\beta \lambda f.\lambda x.f(fx) = \overline{2}$$

Определение (η-эквивалентность)**.**

$$\lambda x.fx =_\eta f$$

Точно так же результаты вычисления `int f(int x)` и `[](int x){return f(x);}` равны.
Арифметические операции:

$$\mathsf{IsZero} = \lambda n.n\,(\lambda x.\,\mathsf{F})\,\mathsf{T} \qquad \mathsf{Add} = \lambda a.\lambda b.\lambda f.\lambda x.a\,f\,(b\,f\,x) \qquad \mathsf{Pow} = \lambda a.\lambda b.b\,(\mathsf{Mul}\,a)\,\overline{1}$$

$$\mathsf{IsEven} = \lambda n.n\,\mathsf{Not}\,\mathsf{T} \qquad \mathsf{Mul} = \lambda a.\lambda b.a\,(\mathsf{Add}\,b)\,\overline{0} \qquad \mathsf{Pow'} = \lambda a.\lambda b.ba$$

Для того, чтобы определить (-1), сначала определим «пару»:

$$\langle a, b\rangle = \lambda f.f\,a\,b \qquad \mathsf{First} = \lambda p.p\,\mathsf{T} \qquad \mathsf{Second} = \lambda p.p\,\mathsf{F}$$

Затем n раз применим функцию $f\,(\langle a, b\rangle) = \langle b, b+1\rangle$ и возьмём первый элемент пары:

$$(-1) = \lambda n.\,\mathsf{First}(n\,(\lambda p.\,\langle(\mathsf{Second}\,p)\,,(+1)\,(\mathsf{Second}\,p)\rangle)\,\langle\overline{0},\overline{0}\rangle)$$

Определение (нормальная форма)**.** Терм A — нормальная форма, если в нём нет β-редексов. Нормальной формой A называется такая нормальная форма B, что $A \twoheadrightarrow_\beta B$.

Утверждение 1.1. *Существует λ-выражение, не имеющее нормальной формы.*

Определение (комбинатор)**.** Комбинатор — λ-выражение без свободных переменных.

Определение.

$$\Omega = \omega\,\omega \qquad \omega = \lambda x.xx$$

Ω не имеет нормальной формы.

Определение (комбинатор неподвижной точки)**.**

$$\mathcal{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

Определение (β-эквивалентность)**.** $A =_\beta B$, если $\exists C : C \twoheadrightarrow_\beta A, C \twoheadrightarrow_\beta B$

Утверждение 1.2.

$$\mathcal{Y}\,f =_\beta f(\mathcal{Y}\,f)$$

Из-за этого свойства $\mathcal{Y}$-комбинатор и получил своё название.

Доказательство. Наивное β-редуцирование:

$$\begin{aligned}\mathcal{Y}\,f &=_\beta (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))f\\ &=_\beta (\lambda x.f(xx))(\lambda x.f(xx))\\ &=_\beta f((\lambda x.f(xx))(\lambda x.f(xx)))\\ &=_\beta f(\mathcal{Y}\,f)\end{aligned}$$

□

Таким образом, с помощью $\mathcal{Y}$-комбинатора можно определять рекурсивные функции.

Пример. Определим факториал Fact с помощью вспомогательной функции Fact'.

$$\mathsf{Fact'} = \lambda f n.\,\mathsf{IsZero}\,n\,\overline{1}\,(\mathsf{Mul}\,n\,(f\,((-1)\,n)))$$

$$\mathsf{Fact} = \mathsf{Y}\,\mathsf{Fact'}$$

$$\begin{aligned}\mathsf{Fact}\,\overline{3} &=_\beta \mathcal{Y}\,\mathsf{Fact'}\,\overline{3} =_\beta \mathsf{Fact'}\,(\mathcal{Y}\,\mathsf{Fact'})\,\overline{3}\\ &=_\beta \mathsf{IsZero}\,\overline{3}\,\overline{1}\,(\mathsf{Mul}\,\overline{3}\,(\mathcal{Y}\,\mathsf{Fact'}\,((-1)\,\overline{3}))) =_\beta \mathsf{Mul}\,\overline{3}\,(\mathcal{Y}\,\mathsf{Fact'}\,((-1)\,\overline{3})) =_\beta \mathsf{Mul}\,\overline{3}\,(\mathcal{Y}\,\mathsf{Fact'}\,\overline{2})\\ &=_\beta \ldots =_\beta \mathsf{Mul}\,\overline{3}\,(\mathsf{Mul}\,\overline{2}\,(\mathcal{Y}\,\mathsf{Fact'}\,\overline{1})) =_\beta \ldots =_\beta \mathsf{Mul}\,\overline{3}\,(\mathsf{Mul}\,\overline{2}\,(\mathsf{Mul}\,\overline{1}\,(\mathcal{Y}\,\mathsf{Fact'}\,\overline{0}))) =_\beta \ldots\\ &=_\beta \mathsf{Mul}\,\overline{3}\,(\mathsf{Mul}\,\overline{2}\,(\mathsf{Mul}\,\overline{1}\,\overline{1})) =_\beta \overline{6}\end{aligned}$$

Теорема Чёрча-Россера

Определение (ромбовидное свойство)**.** G обладает ромбовидным свойством, если какие бы ни были a, b, c, что aGb, aGc, $b \neq c$, найдётся такое d, что bGd и cGd.

Пример. $(<)$ на натуральных числах обладает ромбовидным свойством. $(>)$ на натуральных числах не обладает ромбовидным свойством.

β-редукция не обладает ромбовидным свойством (рисунок 1).

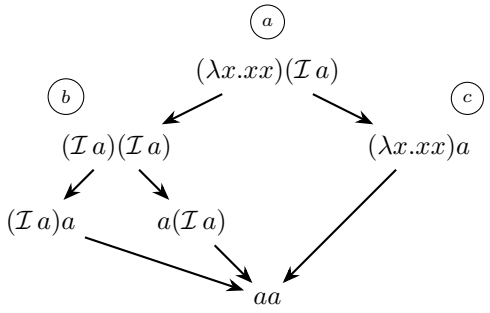

Рис. 1: Нет такого d, что $b \to_\beta d$ и $c \to_\beta d$.

Теорема 1.3 (Чёрча-Россера)**.** *β-редуцируемость обладает ромбовидным свойством.*

Лемма 1.4. *Если R обладает ромбовидным свойством, то R^* (транзитивно-рефлексивное замыкание R) обладает ромбовидным свойством.*

Доказательство. Пусть $M_1 R^* M_n$ и $M_1 R N_1$. Тогда существуют такие $M_2 \dots M_{n-1}$, что $M_1 R M_2 \dots M_{n-1} R M_n$. Так как R обладает ромбовидным свойством, $M_1 R M_2$ и $M_1 R N_1$, то существует такое N_2, что $N_1 R N_2$ и $M_2 R N_2$. Аналогично, существуют такие $N_3 \dots N_n$, что $N_{i-1} R N_i$ и $M_i R N_i$. Мы получили такое N_n, что $N_1 R^* N_n$ и $M_n R^* N_n$.

Пусть теперь $M_{1,1} R^* M_{1,n}$ и $M_{1,1} R^* M_{m,1}$, то есть имеются $M_{1,2} \dots M_{1,n-1}$ и $M_{2,1} \dots M_{m-1,1}$, что $M_{1,i-1} R M_{1,i}$ и $M_{i-1,1} R M_{i,1}$. Тогда существует такое $M_{2,n}$, что $M_{2,1} R^* M_{2,n}$ и $M_{1,n} R^* M_{2,n}$. Аналогично, существуют такие $M_{3,n} \dots M_{m,n}$, что $M_{i,1} R^* M_{i,n}$ и $M_{1,n} R^* M_{i,n}$. Тогда $M_{1,n} R^* M_{m,n}$ и $M_{m,1} R^* M_{m,n}$. □

Определение (параллельная β-редукция)**.** Обозначается как $A \rightrightarrows_\beta B$.

1. Если $A =_\alpha B$, то $A \rightrightarrows_\beta B$
2. Если $A \rightrightarrows_\beta B$, то $\lambda x.A \rightrightarrows_\beta \lambda x.B$
3. Если $P \rightrightarrows_\beta Q$ и $R \rightrightarrows_\beta S$, то $PR \rightrightarrows_\beta QS$
4. Если $P \rightrightarrows_\beta R$ и $Q \rightrightarrows_\beta S$, то $(\lambda x.P)Q \rightrightarrows_\beta R[x := S]$.

Утверждение 1.5. *$(\rightrightarrows_\beta)$ обладает ромбовидным свойством.*

Доказательство. **TODO** □

Утверждение 1.6. *Если $A \to_\beta B$, то $A \rightrightarrows_\beta B$.*

Утверждение 1.7. *Если $A \rightrightarrows_\beta B$, то $A \twoheadrightarrow_\beta B$.*

Доказательство. **TODO** □

При этом обратное, очевидно, не всегда верно (рисунок 2).

Утверждение 1.8. *Из 1.6 и 1.7 следует, что $(\to_\beta)^* = (\rightrightarrows_\beta)^*$.*

Доказательство. Теорема Чёрча-Россера следует из 1.5 и 1.8. □

Следствие 1.9. *Нормальная форма для λ-выражения единственна, если существует.*

Теорема 1.10 (тезис Чёрча)**.** *Если функция вычислима с помощью механического аппарата, то она вычислима с помощью λ-выражения.*

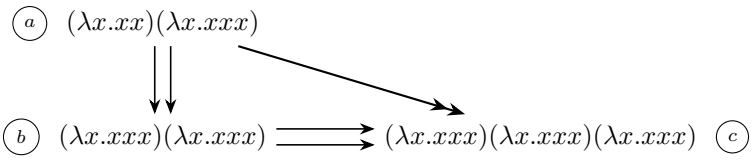

Рис. 2: $a \twoheadrightarrow_\beta c$, но $a \not\twoheadrightarrow_\beta c$.

Порядок редукции

> «Завтра! Завтра! Не сегодня!» — так ленивцы говорят.
>
> Das deutsches Sprichwort

Допустим, мы действительно хотим с помощью λ-исчисления что-то посчитать. Нам надо определиться со стратегией редукции. Если она будет выбрана неудачно, то мы рискуем не дождаться ожидаемого результата.

Определение.

$$\mathcal{K} = \lambda x \lambda y.x \qquad \mathcal{I} = \lambda x.x \qquad \mathcal{S} = \lambda xyz.xz(yz)$$

$\mathcal{I}$ выражается через $\mathcal{S}$ и $\mathcal{K}$: $\mathcal{I} = \mathcal{S}\,\mathcal{K}\,\mathcal{K}$.

Утверждение 1.11. *Пусть A — замкнутое λ-выражение. Тогда найдётся выражение T, состоящее только из $\mathcal{S}$ и $\mathcal{K}$, что $A =_\beta T$.*

Нормальная форма некоторых λ-выражений может не достигаться при неудачном выборе порядка редукции (рисунок 3).

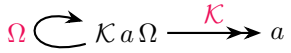

Рис. 3: Редукция Ω обращает выражение в себя.

Определение (нормальный порядок редукции)**.** Редукция самого левого β-редекса.

Если добавить меморизацию к нормальному порядку редукции, то мы получим «ленивые вычисления». То есть, если значение выражения запрашивается много раз, то вычисляться оно будет только один раз. Например, пусть $\mathtt{Mul2} = \lambda a.\,\mathtt{Add}\,aa$. Тогда при вычислении $\mathtt{Mul2}(\mathtt{Add}\,\overline{2}\,\overline{2})$ в $\mathtt{Mul2}$ будет передана ссылка на выражение $\mathtt{Add}\,\overline{2}\,\overline{2}$, во время первого запроса значения выражение будет вычислено и результат будет сохранён по ссылке, и во время второго запроса будет возвращён уже посчитанный результат.

Определение (аппликативный порядок редукции)**.** Редукция самого левого β-редекса из самых вложенных.

Такой порядок редукции ещё называют «энергичными вычислениями».

Утверждение 1.12. *Если нормальная форма существует, она может быть достигнута нормальным порядком редукции.*

Парадокс Карри

Попробуем наивно построить логику на основе λ-исчисления. Введём комбнатор-импликацию, обозначим ($\supset$). Довольно естественно требовать он нашего исчисления M.P. и следующие схемы аксиом:

1. $A \supset A$

2. $(A \supset (A \supset B)) \supset (A \supset B)$

3. $A =_\beta B$, тогда $A \supset B$

Не смотря на кажущуюся простоту, данная логика уже противоречива. Покажем, как в полученной логике можно доказать любое утверждение. Введём обозначение: $\mathcal{Y}_{\supset a} \equiv \mathcal{Y}(\lambda t.t \supset a) =_\beta (\lambda t.t \supset a)(\mathcal{Y}(\lambda t.t \supset a)) =_\beta \mathcal{Y}(\lambda t.t \supset a) \supset a$.

1)	$\mathcal{Y}_{\supset a} \supset \mathcal{Y}_{\supset a}$	(схема аксиом)
2)	$\mathcal{Y}_{\supset a} \supset (\mathcal{Y}_{\supset a} \supset a)$	(можно доказать)
3)	$(\mathcal{Y}_{\supset a} \supset \mathcal{Y}_{\supset a} \supset a) \supset (\mathcal{Y}_{\supset a} \supset a)$	(схема аксиом)
4)	$\mathcal{Y}_{\supset a} \supset a$	(M.P.)
5)	$(\mathcal{Y}_{\supset a} \supset a) \supset \mathcal{Y}_{\supset a}$	(третье правило)
6)	$\mathcal{Y}_{\supset a}$	(M.P.)
7)	a	(M.P.)

Сделать это нам дал возможность тот факт, что в нашей логике с помощью $\mathcal{Y}$-комбинатора мы можем ссылаться в утверждении на само себя. Аналогично можно прийти к парадоксальному выводу из высказывания «если это утверждение верно, то русалки существуют» на нашем мета-языке.

Просто типизированное λ-исчисление

Импликационный фрагмент ИИВ

Определение (импликационный фрагмент ИИВ)**.** Рассмотрим интуиционистское исчисление высказываний.

1. Введём схему аксиом:
$$\frac{}{\Gamma, \varphi \vdash \varphi}$$
2. Правило введения импликации:
$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}$$
3. И правило удаления импликации:
$$\frac{\Gamma \vdash \varphi \to \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$

Мы построили импликационный фрагмент ИИВ (и.ф.и.и.в).

Пример. Докажем $\varphi \to \psi \to \varphi$:
$$\dfrac{\dfrac{\dfrac{}{\varphi, \psi \vdash \varphi}\ (1)}{\varphi \vdash \psi \to \varphi}\ (2)}{\vdash \varphi \to (\psi \to \varphi)}\ (2)$$

Теорема 2.1. *И.ф.и.и.в полон в моделях Крипке, то есть* $\Gamma \vdash \varphi$ *т.и.т.т., когда для любой модели крипке* C *из* $\Vdash_C \Gamma$ *следует* $\Vdash_C \varphi$*.*

Доказательство. Рассмотрим модель Крипке вида $W = \{\Delta \mid \Gamma \subseteq \Delta\}$, где Δ замкнуто относительно $\vdash$, то есть из $\Delta \vdash \varphi$ следует $\varphi \in \Delta$. $\Gamma \leq \Delta$ если $\Gamma \subseteq \Delta$. Индукцией по структуре φ покажем, что $\Delta \Vdash \varphi$ т.и.т.т., когда $\Delta \vdash \varphi$.

1. Пусть $\varphi \equiv x$ — переменная. Тогда $\Gamma \vdash \varphi$ эквивалентно $x \in \Gamma$, что эквивалентно $\Gamma \Vdash x$ (по определению).
2. Пусть $\varphi \equiv \alpha \to \beta$.
   (а) Пусть $\Delta \vdash \alpha \to \beta$. Рассмотрим такое Δ', что $\Delta \leq \Delta'$ и $\Delta' \Vdash \alpha$. Так как $\Delta \vdash \alpha \to \beta$, то $\Delta' \vdash \alpha \to \beta$. Так как $\Delta' \Vdash \alpha$, то по индукционному предположению $\Delta' \vdash \alpha$.
   $$\frac{\Delta' \vdash \alpha \to \beta \qquad \Delta' \vdash \alpha}{\Delta' \vdash \beta}$$
   Значит, $\Delta' \Vdash \beta$ по индуционному предположению. Тогда и $\Delta' \Vdash \alpha \to \beta$.
   (б) Пусть $\Delta \Vdash \alpha \to \beta$. Пусть Δ' — замыкание относительно выводимости $\Delta \cup \{\alpha\}$. Тогда $\Delta' \Vdash \beta$. Тогда $\Delta' \vdash \beta$ по предположению индукции.
   $$\frac{\Delta', \alpha \vdash \beta}{\Delta' \vdash \alpha \to \beta}$$
□

Следствие 2.2. *И.ф.и.и.в замкнут относительно выводимости.*

Если некоторое утверждение выводится в ИИВ ($\vdash_{\text{и}} \varphi$) и содержит только импликации, то оно выводится и в и.ф.и.и.в. ($\vdash_{\text{и}\to} \varphi$).

Исчисление по Карри

Определение (тип)**.** $T = \{\alpha, \beta, \gamma \ldots\}$ — множество типов. σ, τ — метапеременные для типов. Если σ, τ — типы, то $\sigma \to \tau$ — тип.

$$\Pi ::= T \mid \Pi \to \Pi \mid \ (\Pi)$$

$(\to)$ правоассоциативна.

Определение (контекст)**.** Контекст — Γ.

$$\Gamma = \{\Lambda_1 : \sigma_1 \ldots \Lambda_n : \sigma_n\}$$
$$|\Gamma| = \{\sigma_1 \ldots \sigma_n\}$$
$$\operatorname{dom}\Gamma = \{\Lambda_1 \ldots \Lambda_n\}$$

Если $i \neq j$, то $\Lambda_i \neq \Lambda_j$.

Определение (типизируемость по Карри)**.** Рассмотрим исчисление со следующими правилами:

1. $\dfrac{}{\Gamma, x : \sigma \vdash x : \sigma}$ $(x \notin \operatorname{dom}(\Gamma))$
2. $\dfrac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$
3. $\dfrac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}$ $(x \notin \operatorname{dom}(\Gamma))$

Если λ-выражение типизируется этими трёмя правилами, то говорят, что оно типизируется по Карри.

Лемма 2.3 (subject reduction)**.** *Если* $\Gamma \vdash M : \sigma$ *и* $M \to_\beta N$*, то* $\Gamma \vdash N : \sigma$*.*

Следствие 2.4. *Если* $\Gamma \vdash M : \sigma$ *и* $M \twoheadrightarrow_\beta N$*, то* $\Gamma \vdash N : \sigma$*.*

Теорема 2.5 (Чёрча-Россера)**.** *Если* $\Gamma \vdash M : \sigma$*,* $M \twoheadrightarrow_\beta N$ *и* $M \twoheadrightarrow_\beta P$*, то найдётся такое* Q*, что* $N \twoheadrightarrow_\beta Q$*,* $P \twoheadrightarrow_\beta Q$ *и* $\Gamma \vdash Q : \sigma$*.*

Пример. Несколько доказательств:

1. $\lambda x.x : \alpha \to \alpha$:
$$\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}\ (1)}{\vdash \lambda x.x : \alpha \to \alpha}\ (3)$$

2. $\lambda f.\lambda x.fx : (\alpha \to \beta) \to \alpha \to \beta$:
$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f : \alpha \to \beta}\ (1) \qquad \dfrac{}{\Gamma \vdash x : \alpha}\ (1)}{f : \alpha \to \beta; x : \alpha \vdash fx : \beta}\ (2)}{f : \alpha \to \beta \vdash \lambda x.fx : \alpha \to \beta}\ (3)}{\vdash \lambda f.\lambda x.fx : (\alpha \to \beta) \to (\alpha \to \beta)}\ (3)$$

3. Ω *и* $\mathcal{Y}$ *не типизируемы.* Допустим обратное. Тогда в выводе должен будет присутствовать вывод подвыражения xx:
$$\dfrac{\dfrac{}{\Gamma, x : \sigma \to \tau, x : \sigma \vdash x : \sigma \to \tau} \qquad \dfrac{}{\Gamma, x : \sigma \to \tau, x : \sigma \vdash x : \sigma}}{\Gamma, x : \sigma \to \tau, x : \sigma \vdash xx : \tau}$$

   Однако первое правило вывода запрещает повторение переменных в контексте. Значит, такой вывод не может быть корректным. □

Если мы знаем тип выражения, то построить вывод этого типа нам не составит труда. По виду λ-терма можно однозначно сказать, каким правилом был выведен его тип. Правилом 1 выводится тип обособленных переменных, правилом 2 выводится тип аппликаций, правилом 3 — абстракций.

Лемма 2.6 (отсутствие subject expansion)**.** *Неверно, что если* $M \to_\beta N$, $\Gamma \vdash N : \sigma$, *то* $\Gamma \vdash M : \sigma$.

Например, для $\mathcal{K}\, a\, \Omega$.

В общем случае тип не уникален. Например, одновременно $\vdash \lambda x.x : \alpha \to \alpha$ и $\vdash \lambda x.x : (\beta \to \beta) \to (\beta \to \beta)$.

Определение (сильная нормализация)**.** Назовём исчисление сильно нормализуемым, если любая последовательность β-редукций неизбежно приводит к нормальной форме.

Или, иными словами, если не существует бесконечной последовательности β-редукций.

Определение (слабая нормализация)**.** Назовём исчисление слабо нормализуемым, если для любого терма существует последовательность β-редукций, приводящая его к нормальной форме.

Теорема 2.7 (о сильной нормализации)**.** *Просто типизируемое* λ*-исчисление сильно нормализуемо. Любое просто типизируемое* λ*-выражение сильно нормализуемо.*

Исчисление по Чёрчу

Определение (типизация по Чёрчу)**.** Расширим грамматику:

$$\Lambda_{\text{ч}} ::= x \mid \lambda x^\sigma.\Lambda_{\text{ч}} \mid (\Lambda_{\text{ч}}) \mid \Lambda_{\text{ч}}\Lambda_{\text{ч}}$$

Правила вывода:

1. $\dfrac{}{\Gamma, x : \sigma \vdash_{\text{ч}} x : \sigma}$ $(x \notin \mathrm{dom}(\Gamma))$
2. $\dfrac{\Gamma \vdash_{\text{ч}} M : \sigma \to \tau \qquad \Gamma \vdash_{\text{ч}} N : \sigma}{\Gamma \vdash_{\text{ч}} MN : \tau}$
3. $\dfrac{\Gamma, x : \sigma \vdash_{\text{ч}} M : \tau}{\Gamma \vdash_{\text{ч}} \lambda x^\sigma.M : \sigma \to \tau}$ $(x \notin \mathrm{dom}(\Gamma))$

Определение (стирание)**.** Функцией стирания называется следующая функция из $\Lambda_{\text{ч}}$ в Λ:

$$|\Lambda_{\text{ч}}| = \begin{cases} x & \Lambda_{\text{ч}} \equiv x \\ |\Lambda_1|\,|\Lambda_2| & \Lambda_{\text{ч}} \equiv \Lambda_1\Lambda_2 \\ \lambda x.\,|\Lambda| & \Lambda_{\text{ч}} \equiv \lambda x^\sigma.\Lambda \end{cases}$$

Лемма 2.8 (subject reduction по Чёрчу)**.** *Пусть* $\Gamma \vdash_{\text{ч}} M : \sigma$ *и* $|M| \to_\beta N$. *Тогда найдётся такое* H, *что* $|H| = N$, $\Gamma \vdash_{\text{ч}} H : \sigma$.

Теорема 2.9 (Чёрча-Россера)**.** *Пусть* $\Gamma \vdash_{\text{ч}} M : \sigma$, $|M| \twoheadrightarrow_\beta N$, $|M| \twoheadrightarrow_\beta T$. *Тогда найдётся такое* P, *что* $\Gamma \vdash_{\text{ч}} P : \sigma$, $N \twoheadrightarrow_\beta |P|$ *и* $T \twoheadrightarrow_\beta |P|$.

Единственное отличие исчисления по Чёрчу от исчисления по Карри — это типизация всех абстракций. Из этого вытекает следующая лемма:

Лемма 2.10 (уникальность типов)**.** *Если* $\Gamma \vdash_{\text{ч}} M : \sigma$ *и* $\Gamma \vdash_{\text{ч}} M : \tau$, *то* $\sigma = \tau$.

Теорема 2.11 (о стирании)**.**

1. *Если* $M \to_\beta N$ *и* $\Gamma \vdash_{\text{ч}} M : \sigma$, *то* $|M| \to_\beta |N|$.
2. *Если* $\Gamma \vdash_{\text{ч}} M : \sigma$, *то* $\Gamma \vdash_{\text{к}} |M| : \sigma$.

Теорема 2.12 (о поднятии)**.** *Пусть* $P \in \Lambda_{\text{ч}}$, $M, N \in \Lambda_{\text{к}}$.

1. *Если* $M \to_\beta N$, $|P| = M$, *то найдётся такое* Q, *что* $|Q| = N$, $P \to_\beta Q$.
2. *Если* $\Gamma \vdash_{\text{к}} M : \sigma$, *то найдётся такое* $P \in \Lambda_{\text{ч}}$, *что* $\Gamma \vdash_{\text{ч}} P : \sigma$, $|P| = M$.

Изоморфизм Карри-Ховарда

Можно отчётливо проследить связь между аксиомами типизированного λ-исчисления и аксиомами импликационного фрагмента ИИВ. Сейчас мы введём два новых типа данных, правила для которых будут соответствовать связкам $\&$ и $\vee$ из ИИВ.

Первый — тип «пары». Пара хранит в себе два значения. Пусть $A : \sigma$ и $B : \tau$, тогда:

$$\langle A, B\rangle : \sigma \,\&\, \tau$$
$$\pi_1 : \sigma \,\&\, \tau \to \sigma \qquad \pi_2 : \sigma \,\&\, \tau \to \tau$$

π_1 и π_2 это функции для доступа к элементам пары, левая и правая проекции. Например, $\pi_2 \left\langle \overline{7}, \overline{5} \right\rangle =_\beta \overline{5}$.

Второй — алгебраический тип данных. Пусть $T : \sigma \vee \tau$, $L : \sigma \to \pi$, $R : \tau \to \pi$. Тогда:

$$\texttt{case}\ T\ L\ R : \pi \qquad \mathrm{inj}_1 : \sigma \to \sigma \vee \tau \qquad \mathrm{inj}_2 : \tau \to \sigma \vee \tau$$

Если $A : \sigma$ и $B : \tau$, то $\mathrm{inj}_1\, A$ и $\mathrm{inj}_2\, B$ имеют тип $\sigma \vee \tau$. inj_1 и inj_2 — это левая и правая инъекции. Тип $\sigma \vee \tau$ означает, что это либо σ, либо τ. Его можно понимать как `union`, который дополнительно хранит в себе информацию о типе, который в нём записан.

Рассмотрим алгебраический тип данных на Haskell: `data Fruit = Orange Int | Banana Int`. Левая инъекция для него — `Orange :: Int -> Fruit`, правая — `Banana :: Int -> Fruit`. На нашем языке можно сказать, что `Fruit` = `Int` $\vee$ `Int`. Тогда если T : `Fruit`, то `case` T $(+1)$ (-1) : `Int` это число, хранящееся в фрукте, увеличенное на единицу, если это апельсин, и уменьшенное на единицу, если это банан.

Соответствие описано в таблицах 1 и 2. Формально изоморфизм описывается следующей теоремой:

Теорема 2.13 (об изоморфизме Карри-Ховарда)**.**

1. *Пусть* $\Delta \vdash_{\text{ч}} M : \sigma$*, тогда* $|\Delta| \vdash \sigma$*.*
2. *Пусть* $\Gamma \vdash \sigma$ *в и.ф.и.и.в., тогда найдётся такой терм* M*, что* $\Delta \vdash_{\text{ч}} M : \sigma$*, где* $\Delta = \{(x : \varphi) \mid \varphi \in \Gamma\}$*.*

Доказательство.

1. Индукция по выводу $\Delta \vdash_{\text{ч}} M : \sigma$. Заменяем каждое правило в выводе соответствующим правилом из ИИВ и получаем доказательство $|\Delta| \vdash \sigma$.
2. Индукция по структуре вывода $\Gamma \vdash \sigma$. Пусть $\Gamma = \{\sigma_1, \sigma_2 \ldots\}$, $\Delta = \{x_1 : \sigma_1, x_2 : \sigma_2 \ldots\}$.
   (а) Вывод имеет вид:
   $$\frac{}{\Gamma, \varphi \vdash \varphi}$$
   i. Если $\varphi \in \Gamma$, то $\dfrac{}{\Delta \vdash x : \varphi}$.
   ii. Если $\varphi \notin \Gamma$, то $\dfrac{}{\Delta, x : \varphi \vdash x : \varphi}$.
   (б) Вывод заканчивается правилом:
   $$\frac{\Gamma \vdash \varphi \to \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$
   По индукционному предположению $\Delta \vdash M : \varphi \to \psi$ и $\Delta \vdash N : \varphi$. Тогда:
   $$\frac{\Delta \vdash A : \varphi \to \psi \qquad \Delta \vdash B : \varphi}{\Delta \vdash MN : \psi}$$
   (в) Вывод заканчивается правилом:
   $$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}$$

ИИВ	Типизированное λ-исчисление
$\dfrac{\Gamma \vdash \varphi \to \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$	$\dfrac{\Gamma \vdash A : \varphi \to \psi \qquad \Gamma \vdash B : \varphi}{\Gamma \vdash AB : \psi}$
$\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}$	$\dfrac{\Gamma, x : \varphi \vdash A : \psi}{\Gamma \vdash \lambda x^{\varphi}.A : \varphi \to \psi}$
$\dfrac{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}{\Gamma \vdash \varphi \,\&\, \psi}$	$\dfrac{\Gamma \vdash A : \varphi \qquad \Gamma \vdash B : \psi}{\Gamma \vdash \langle A, B \rangle : \varphi \,\&\, \psi}$
$\dfrac{\Gamma \vdash \varphi \,\&\, \psi}{\Gamma \vdash \varphi}$	$\dfrac{\Gamma \vdash R : \varphi \,\&\, \psi}{\Gamma \vdash \pi_1 R : \varphi}$
$\dfrac{\Gamma \vdash \varphi \,\&\, \psi}{\Gamma \vdash \psi}$	$\dfrac{\Gamma \vdash R : \varphi \,\&\, \psi}{\Gamma \vdash \pi_2 R : \psi}$
$\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi}$	$\dfrac{\Gamma \vdash A : \varphi}{\Gamma \vdash \mathrm{inj}_1\, A : \varphi \vee \psi}$
$\dfrac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi}$	$\dfrac{\Gamma \vdash A : \psi}{\Gamma \vdash \mathrm{inj}_2\, A : \varphi \vee \psi}$
$\dfrac{\Gamma \vdash \varphi \to \pi \qquad \Gamma \vdash \psi \to \pi}{\Gamma \vdash \varphi \vee \psi \to \pi}$	$\dfrac{\Gamma \vdash T : \varphi \vee \psi \qquad \Gamma \vdash A : \varphi \to \pi \qquad \Gamma \vdash B : \psi \to \pi}{\Gamma \vdash \mathtt{case}\ T\ A\ B : \pi}$

Таблица 1: Соответствие правил вывода.

Интуиционистская логика	λ-исчисление
Выражение	Тип
Доказательство	Терм (программа)
Предположение	Свободная переменная
Импликация	Абстракция (функция)

Таблица 2: Соответствие сущностей.

i. Пусть $\varphi \in \Gamma$. Тогда по индукционному предположению $\Delta \vdash M : \psi$. Возьмём новую переменную $x \notin \mathrm{dom}(\Delta)$. Тогда можно изменить построенное доказательство и получить $\Delta, x : \varphi \vdash M : \psi$.

ii. Пусть $\varphi \notin \Gamma$. Тогда по индукционному предположению $\Delta, x : \varphi \vdash M : \psi$.

Тогда:

$$\frac{\Delta, x : \varphi \vdash M : \psi}{\Delta \vdash \lambda x^{\varphi}.M : \varphi \to \psi}$$

□

Вывод типа

> Помните, что в λ-исчислении нет смысла? Здесь смысл отрицательный, скорее.
>
> Д.Г.

В λ-исчислении выделяют 3 задачи:

(а) Проверка типа: верно ли $\Gamma \vdash M : \sigma$?

(б) Вывод типа: существуют ли такие Γ и σ, что $\Gamma \vdash M : \sigma$?

(в) Обитаемость типа: существуют ли такие Γ и M, что $\Gamma \vdash M : \sigma$?

Первые две задачи возникают в контексте дизайна языков программирования. Понятно, что задача проверки типа сводится к задаче вывода типа.

Из изоморфизма Карри-Ховарда ясно, что задача обитаемости типа эквивалентна задаче проверки выражения ИИВ на доказуемость, а задача проверки типа может быть рассмотрена как верификация доказательства.

В этом разделе мы будем рассматривать задачу вывода типа.

Определение (алгебраический терм)**.**

$$A ::= x \mid f\,(A, \ldots, A)$$

где $x \in X$, f — функциональный символ.

Уравнение в алгебраических термах: $A = A$.

Определение (подстановка)**.** Подстановкой называется функция из A в A:

$$S : A \to A$$

Причём S — id[2] почти везде (везде, кроме конечного количества).

Определение (естественное обобщение)**.** Естественное обобщение — такая подстановка $S : A \to A$, получаемая из $S : X \to A$, что $S\,(f\,(A_1 \ldots A_n)) = f\,(S(f_1) \ldots S(f_n))$

Определение (унификатор)**.** S — унификатор (решение уравнения) $P = Q$, если $S(P) = S(Q)$.

Пример. Пусть

$$\sigma = \beta \to \alpha \to \beta \qquad \tau = (\gamma \to \gamma) \to \delta$$

Их унификатор это

$$S = [\beta \coloneqq \gamma \to \gamma, \delta \coloneqq \alpha \to \gamma \to \gamma] \qquad S(\sigma) \equiv S(\tau) = (\gamma \to \gamma) \to \alpha \to \gamma \to \gamma$$

Задача решения уравнения в алгебраических термах — унификация.

Определение (композиция)**.** $(S \circ T)(A) = S(T(A))$

Определение (частный случай)**.** T — частный случай U, если существует такое S, что $T = S \circ U$.

[2] id — общепринятое обозначение функции $f(x) = x$.

Определение (наиболее общий унификатор)**.** Наиболее общий унификатор U для уравнения $A = B$ — такой унификатор, что:

1. $U(A) = U(B)$.
2. Любой другой унификатор — частный случай U.

Определение (несовместная система)**.** Назовём систему несовместной, если выполнено одно из условий:

1. В ней есть уравнение вида $f(\ldots) = g(\ldots)$.
2. В ней есть уравнение вида $x = f(\ldots\ x\ \ldots)$.

Определение (эквивалентные системы)**.** Назовём две системы эквивалентными, если они имеют одинаковые решения.

Утверждение 2.14. *Для любой системы*

$$\begin{cases} A_1 = B_1 \\ \quad\vdots \\ A_n = B_n \end{cases}$$

найдётся эквивалентная ей система из одного уравнения:

$$f(A_1 \ldots A_n) = f(B_1 \ldots B_n)$$

где f — новый символ.

Определение (разрешённая система)**.** Назовём систему разрешённой, если:

1. Все уравнения имеют вид $x = A$.
2. Если переменная встречается в левой части, то ни в каком уравнении она не встречается в правой части.
3. Переменная может встречаться в левой части только одного уравнения.

По системе в разрешённой форме $\{x_i = A_i\}$ мы можем построить решение S, определив $S(x_i) = A_i$ для каждого i.

Утверждение 2.15. *Построенный по системе в разрешённой форме унификатор S — наиболее общий унификатор.*

Утверждение 2.16. *Несовместная система не имеет решений.*

Рассмотрим следующие преобразования, не меняющие свойств системы:

Выражения	Условия	Новые выражения
$T = x$, T не переменная		$x = T$
$T = T$		убрать это уравнение
$f(A_1, \ldots A_n) = g(B_1, \ldots B_n)$	$f = g$	$A_1 = B_1 \ldots A_n = B_n$
	$f \neq g$	система несовместна
$x = T$, $R = S$, x входит в S или R	T не содержит x	$x = T$, $R\,[x \coloneqq T] = S\,[x \coloneqq T]$
	T содержит x	система несовместна

Утверждение 2.17. *Последовательное применение преобразований либо за конечное число шагов приведёт систему в разрешённый вид, либо покажет, что она несовместна.*

Доказательство. Пусть (n_1, n_2, n_3) — характеристика системы, где n_1 — количество переменных, не входящих систему только слева от знака равенства один раз, n_2 — общее количество вхождений функциональных символов в S, n_3 — количество выражений вида $T = x$ или $T = T$. Каждое преобразование уменьшает эту тройку (если сравнивать лексикографически). □

Теорема 2.18. *Задача вывода типа в λ-исчислении разрешима.*

Доказательство. Опишем алгоритм. Пусть нам дан λ-терм M. Рекурсивно построим по нему систему уравнений E_m и тип выражения τ_m:

Вид M	E_m	τ_m
x	$\varnothing$	α_x — новая переменная
PR	$E_p \cup E_r \cup \{\tau_p = \tau_r \to \pi\}$	π — новая переменная
$\lambda x.P$	E_p	$\tau_x \to \tau_p$

Решим построенную систему уравнений. Пусть унификатор S — решение системы уравнений E_m. Тогда тип M это $S(\tau_m)$.

Можно показать, что алгоритм корректный. □

Отметим важную для реализации деталь. Одной переменной должен соответствовать только один тип. Однако не всегда переменные, называющиеся одинаково, мы считаем одинаковыми. В выражении $\lambda f x.f\,x\,x$ важно, чтобы выделенные переменные получили один тип. Но в выражении $\lambda x.x\lambda x.x\lambda x.x$ типы выделенных переменных должны быть разными. Его тип должен быть такой же, как и, например, тип выражения $\lambda x.x\lambda y.y\lambda z.z$. Можно сформулировать правило: построенная система уравнений должна оставаться корректной для любого выражения, α-эквивалентного исходному.

Пример. Рассмотрим выражение $\overline{2} = \lambda f.\lambda x.f(fx)$. Сначала построим по нему систему уравнений (M — аргумент функции, столбики справа — рекурсивные вызовы, E и τ — результат функции):

$M = \lambda f x.f(fx)$	$M = \lambda x.f(fx)$	$M = f(fx)$	$M = f$ $E = \varnothing$ $\tau = \alpha$	
			$M = fx$	$M = f$ $E = \varnothing$ $\tau = \alpha$
				$M = x$ $E = \varnothing$ $\tau = \beta$
			$E = \{\alpha = \beta \to \gamma\}$ $\tau = \gamma$	
$E = \begin{cases}\alpha = \beta \to \gamma \\ \alpha = \gamma \to \delta\end{cases}$ $\tau = \alpha \to (\beta \to \delta)$	$E = \begin{cases}\alpha = \beta \to \gamma \\ \alpha = \gamma \to \delta\end{cases}$ $\tau = \beta \to \delta$	$E = \begin{cases}\alpha = \beta \to \gamma \\ \alpha = \gamma \to \delta\end{cases}$ $\tau = \delta$		

Затем решим полученную систему уравнений:

$$\begin{cases}\alpha = \beta \to \gamma \\ \alpha = \gamma \to \delta\end{cases} \Longrightarrow \begin{cases}\alpha = \beta \to \gamma \\ \beta \to \gamma = \gamma \to \delta\end{cases} \Longrightarrow \begin{cases}\alpha = \beta \to \gamma \\ \beta = \gamma \\ \gamma = \delta\end{cases} \Longrightarrow \begin{cases}\alpha = \beta \to \delta \\ \beta = \delta \\ \gamma = \delta\end{cases} \Longrightarrow \begin{cases}\alpha = \delta \to \delta \\ \beta = \delta \\ \gamma = \delta\end{cases}$$

Получаем подстановку $S = [\alpha := \delta \to \delta, \beta := \delta, \gamma := \delta]$. Применяя её к τ получаем тип выражения:

$$S(\tau) = (\delta \to \delta) \to (\delta \to \delta)$$
$$\lambda f.\lambda x.f(fx) : (\delta \to \delta) \to (\delta \to \delta)$$

Логика второго порядка и полиморфизм

> Полиморфизм — способность некоторых организмов существовать в состояниях с различной внутренней структурой или в разных внешних формах.
>
> Википедия

Может показаться, что, вооружившись простыми типами, можно начать писать настоящие программы. Разочарование нас ждёт на первой же попытке. Мы уже определяли функцию степени для чисел Чёрча:

$$\mathsf{Pow} = \lambda ab.ba =_\eta \lambda abfx.bafx.$$

Давайте попытаемся её типизировать. С помощью алгоритма вывода типов мы недавно выяснили, что числам Чёрча подходит тип

$$(\delta \to \delta) \to \delta \to \delta.$$

Функция степени принимает два числа, возвращает одно число. Соединение этих двух фактов может показаться довольно естественным:

$$\mathsf{Pow} : ((\delta \to \delta) \to \delta \to \delta) \to ((\delta \to \delta) \to \delta \to \delta) \to (\delta \to \delta) \to \delta \to \delta.$$

Однако этот тип неверный. Если запустить алгоритм вывода типов на выражении Pow, то мы получим тип

$$\alpha \to (\alpha \to \beta) \to \beta,$$

и при попытке унификации этого типа с предполагаемым мы получим несовместную систему, то есть типы действительно разные.

Причина этого ясна. Внутри Pow в b передаётся a, но у a тип не $\delta \to \delta$, как того просит тип b. Это, конечно, можно исправить, но результат выйдет пугающий:

$$\mathsf{Pow} : ((\delta \to \delta) \to \delta \to \delta) \to (((\delta \to \delta) \to \delta \to \delta) \to (\delta \to \delta) \to \delta \to \delta) \to (\delta \to \delta) \to \delta \to \delta.$$

Более того, теперь типы чисел a и b стали разные.

Проблема в том, что числа Чёрча имеют разные типы в зависимости от того, как они используются. В программе давать одинаковым сущностям разные типы в зависимости от того, как они будут использоваться впоследствии, было бы очень неудобно. В привычных нам языках программирования для решения подобных проблем есть различные механизмы для обеспечения *полиморфизма*. В этом разделе мы будем внедрять такой механизм в λ-исчисление.

Интуиционистское исчисление предикатов второго порядка

Определение (грамматика ИИП второго порядка)**.**

$$\Phi ::= (\Phi) \mid p \mid \Phi \to \Phi \mid \forall p.\Phi \underbrace{\mid \exists p.\Phi \mid \bot \mid \Phi \,\&\, \Phi \mid \Phi \vee \Phi}_{\text{несущественные}}$$

Определение (правила вывода в ИИП второго порядка)**.** К правилам обычного ИИВ добавляются правила вывода для квантора всеобщности:

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall p.\varphi}\ (p \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma \vdash \forall p.\varphi}{\Gamma \vdash \varphi\,[p := \sigma]}$$

И для квантора существования:

$$\frac{\Gamma \vdash \varphi\,[p := \psi]}{\Gamma \vdash \exists p.\varphi} \qquad \frac{\Gamma \vdash \exists p.\varphi \qquad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}\ (p \notin \mathrm{FV}(\Gamma, \psi)),$$

где $\mathrm{FV}(\Gamma)$ — множество свободных переменных всех выражений из Γ.

Напомним, что в логике второго порядка переменные под кванторами соответствуют любым выражениям, а не только значениям из предметного множества.

Последние четыре связки можно выразить через первые. $\bot$ — это критерий противоречивости, из него можно вывести любое утверждение. Это можно выразить так:

$$\bot \equiv \forall p.p$$

Такое выражение естественным образом соответствует нашим требованиям, так как, пользуясь правилом вывода для квантора всеобщности, из такого $\bot$ можно вывести любое выражение.

Приведём выражение для конъюнкции:

$$\varphi \,\&\, \psi \equiv \forall a.((\varphi \to \psi \to a) \to a)$$

Его можно читать как «если всё, что выводимо из $\{\varphi, \psi\}$, выполняется, то тогда выполняется $\varphi\&\psi$». Также для такого выражения можно проверить, что правила вывода конъюнкции действительно выводятся через такое определение. Например, выведем φ из $\varphi \,\&\, \psi$:

$$\dfrac{\dfrac{\Gamma \vdash \forall\gamma.(\varphi \to \psi \to \gamma) \to \gamma}{\Gamma \vdash (\varphi \to \psi \to \varphi) \to \varphi} \qquad \dfrac{\dfrac{\overline{\Gamma, \varphi, \psi \vdash \varphi}}{\Gamma, \varphi \vdash \psi \to \varphi}}{\Gamma \vdash \varphi \to \psi \to \varphi}}{\Gamma \vdash \varphi}$$

Выражение для дизъюнкции:

$$\varphi \vee \psi \equiv \forall a.(\varphi \to a) \to (\psi \to a) \to a$$

Формулу для дизъюнкции можно читать как «$\varphi \vee \psi$ выполняется, если любое a, которое выводимо как из φ, так и из ψ, выполняется». Ещё её можно понимать как следствие из формулы де Моргана $\varphi \vee \psi = \neg(\neg\varphi \,\&\, \neg\psi)$.

Формула для квантора существования получается из того, что $\exists x.\tau = \neg\forall x.\neg\tau$:

$$\exists x.\tau \equiv \forall a.(\forall x.\tau \to a) \to a$$

Система F

Определение (тип в системе F)**.**

$$\tau = \begin{cases} \alpha, \beta, \gamma, \ldots & \text{(атомарный тип)} \\ \tau \to \tau & \\ \forall\alpha.\tau & (\alpha \text{ — переменнная}) \end{cases}$$

Определение (система F)**.** Грамматика выражения в системе F:

$$\mathbf{\Lambda} ::= x \mid \lambda p^{\alpha}.\mathbf{\Lambda} \mid \mathbf{\Lambda}\mathbf{\Lambda} \mid\ (\mathbf{\Lambda}) \mid \Lambda\alpha.\mathbf{\Lambda} \mid \mathbf{\Lambda}\tau$$

$\Lambda\alpha.\mathbf{\Lambda}$ — полиморфная абстракция, $\mathbf{\Lambda}\tau$ — применение типа. Правила вывода:

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}\ (x \notin \mathrm{dom}(\Gamma))$$

$$\frac{\Gamma \vdash M : \tau \to \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma} \qquad \frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x^{\tau}.M : \tau \to \sigma}\ (x \notin \mathrm{dom}(\Gamma))$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \Lambda\alpha.M : \forall\alpha.\sigma}\ (\alpha \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma \vdash M : \forall\alpha.\sigma}{\Gamma \vdash M\tau : \sigma[\alpha := \tau]}$$

Полиморфная абстракция — это явное указание того, что вместо каких-то типов мы можем подставить любые выражения.

Пример. Левая проекция:

	Просто типизированное λ-исчисление	Система F
Тип	$\pi_1 : \alpha \,\&\, \beta \to \alpha$	$\pi_1 : \forall\alpha.\forall\beta.\alpha \,\&\, \beta \to \alpha$
Выражение	$\pi_1 = \lambda p.p\,\mathcal{T}$	$\pi_1 = \Lambda\alpha.\Lambda\beta.\lambda p^{\alpha\&\beta}.p\alpha\,\mathcal{T}$

В системе F явно указывается, что типы элементов пары могут быть любыми, как и в типе проекции, так и в её выражении.

Определение (β-редукция в F)**.**

1. Типовая редукция: $(\Lambda\alpha.M^\sigma)\,\tau \to_\beta M[\alpha := \tau] : \sigma[\alpha := \tau]$
2. Классическая β-редукция: $(\lambda x^\sigma.M)^{\sigma\to\tau}\,X \to_\beta M[x := X] : \tau$

Выразим несколько функций:

1. Левая инъекция:
$$\mathrm{inj}_1 = \Lambda\varphi.\Lambda\psi.\lambda x^\varphi.\underline{\Lambda\alpha.\lambda f^{\varphi\to\alpha}.\lambda g^{\psi\to\alpha}.fx}$$
Можно заметить, что подчёркнутый терм имеет тип $\varphi \vee \psi$.
2. Для ясности приведём (довольно бесполезное) выражение `case`:
$$\texttt{case}\ T^{\varphi\vee\psi}\ A^{\varphi\to\alpha}\ B^{\psi\to\alpha} = T\alpha AB$$
Результат всего выражения имеет тип α, который мы явно передали в выражение абстрактного типа данных T.
3. Конструктор пары:
$$\left\langle a^\varphi, b^\psi \right\rangle = \Lambda\varphi.\Lambda\psi.\lambda a^\varphi.\lambda b^\psi.\Lambda\alpha.\lambda f^{\varphi\to\psi\to\alpha}.fab$$
4. Правая проекция:
$$\pi_2 = \Lambda\varphi.\Lambda\psi.\lambda p^{\varphi\&\psi}.p\psi\left(\lambda x^\varphi\lambda y^\psi.y\right)$$

В системе F можно задать общий тип для чёрчевского нумерала: $\forall\alpha.(\alpha\to\alpha)\to\alpha\to\alpha$. Тип α можно менять в зависимости от того, как нумерал используется.

Теорема 3.1 (изоморфизм Карри-Ховарда для системы F)**.** $\Gamma \vdash_F M : \tau$ *т.и.т.т., когда* $|\Gamma| \vdash \tau$ *в интуиционистском исчислении предикатов второго порядка.*

Доказательство изоморфизма для системы F абсолютно аналогичное. Рассмотрим, например, следующее верное утверждение в ИИП второго порядка: «из лжи следует что угодно», или $\forall\beta.\bot\to\beta \equiv \forall\beta.(\forall\alpha.\alpha)\to\beta$. Если мы выпишем λ-выражение с таким типом, то станет очевидно, как доказывать утверждение. Пример такого выражения:

$$\Lambda\beta.\lambda a^{\forall\alpha.\alpha}.a\beta : \forall\beta.(\forall\alpha.\alpha)\to\beta$$

Вывод типа выражения, как и раньше, легко выписывается исходя из самого выражения, так как λ-выражение каждого из видов, описанных в грамматике, можно получить только одним правилом.

$$\dfrac{\dfrac{\dfrac{}{a : \forall\alpha.\alpha \vdash a : \forall\alpha.\alpha}}{a : \forall\alpha.\alpha \vdash a\beta : \beta}}{\vdash \lambda a^{\forall\alpha.\alpha}.a\beta : (\forall\alpha.\alpha)\to\beta}}{\vdash \Lambda\beta.\lambda a^{\forall\alpha.\alpha}.a\beta : \forall\beta.(\forall\alpha.\alpha)\to\beta}$$

Другое λ-выражение с таким же типом это $\Lambda\beta.\lambda a^{\forall\alpha.\alpha}.a((\forall\alpha.\alpha)\to\beta)a$.

Теорема 3.2. F *сильно нормализуема.*

Экзистенциальные типы

Допустим, у нас есть абстрактный тип данных «стек» α и операции над ним:
$\texttt{empty} : \alpha$
$\texttt{push} : \alpha \,\&\, \nu \to \alpha$
$\texttt{pop} : \alpha \to \alpha \,\&\, \nu$
Можно попробовать сказать это так: $\texttt{stack} = \alpha \,\&\, (\alpha \,\&\, \nu \to \alpha) \,\&\, (\alpha \to \alpha \,\&\, \nu)$, где вместо α будет описание хранилища для стека. Однако, проблема в том, что мы можем захотеть скрыть полный тип α, так как в нём будут описаны детали реализации. Нам хотелось бы дать только интерфейс стека. Это можно сказать так: $\exists\alpha.\alpha \,\&\, (\alpha \,\&\, \nu \to \alpha) \,\&\, (\alpha \to \alpha \,\&\, \nu)$. То есть существует какое-то α, реализующее описанный интерфейс.

По аналогии с правилом удаления квантора существования, можно определить правила вывода для выражений экзистенциальных типов:

$$\frac{\Gamma \vdash M : \varphi[\alpha := \theta]}{\Gamma \vdash (\texttt{pack}\ M, \theta\ \texttt{to}\ \exists\alpha.\varphi) : \exists\alpha.\varphi} \qquad \frac{\Gamma \vdash M : \exists\alpha.\varphi \qquad \Gamma, x : \varphi \vdash N : \psi}{\Gamma \vdash \texttt{abstype}\ \alpha\ \texttt{with}\ x : \varphi\ \texttt{in}\ M\ \texttt{is}\ N : \psi}\ (\alpha \notin \mathrm{FV}(\Gamma, \psi))$$

Если вспомнить, что квантор существования выразим через квантор всеобщности, то можно записать выражения `pack` и `abstype` без расширения языка:

$$\texttt{pack}\ M, \theta\ \texttt{to}\ \exists\alpha.\varphi = \Lambda\beta.\lambda x^{\forall\alpha.\varphi\to\beta}.x\theta M$$
$$\texttt{abstype}\ \alpha\ \texttt{with}\ x : \varphi\ \texttt{in}\ M\ \texttt{is}\ N : \psi = M\psi(\Lambda\alpha.\lambda x^{\varphi}.N)$$

Покажем, что $\texttt{abstype}\ \alpha\ \texttt{with}\ x : \varphi\ \texttt{in}\ (\texttt{pack}\ M, \theta\ \texttt{to}\ \exists\alpha.\varphi)\ \texttt{is}\ N$ редуцируется в $N[\alpha := \theta][x := M]$:

$$\begin{aligned}
\texttt{abstype}\ \alpha\ \texttt{with}\ x : \varphi\ \texttt{in}\ (\texttt{pack}\ M, \theta\ \texttt{to}\ \exists\alpha.\varphi)\ \texttt{is}\ N &= (\Lambda\beta.\lambda x^{\forall\alpha.\varphi\to\beta}.x\theta M)\psi(\Lambda\alpha.\lambda x^{\varphi}.N) \\
&\to_\beta (\lambda x^{\forall\alpha.\varphi\to\psi}.x\theta M)(\Lambda\alpha.\lambda x^{\varphi}.N) \\
&\to_\beta (\Lambda\alpha.\lambda x^{\varphi}.N)\theta M \\
&\to_\beta (\lambda x^{\varphi}.N)[\alpha := \theta]M \\
&\to_\beta N[\alpha := \theta][x := M]
\end{aligned}$$

Пример. **TODO** буде незабаром.

Утверждение 3.3. *F неразрешима.*

Ни одна из задач λ-исчисления в системе F не разрешима, даже задача проверки типизации. Доказать это можно через сведение к проблеме останова.

Итак, мы попытались добавить к типизированному λ-исчислению полиморфизм и абстрактные типы данных и получили слишком сложный язык. Давайте попробуем немного его упростить, чтобы с ним можно было работать.

Типовая система Хиндли-Милнера

Определение (ранг типа)**.**

$$\mathrm{rk}(\tau) = \begin{cases} \max(\mathrm{rk}(\sigma) + 1, \mathrm{rk}(\rho)) & \tau \equiv \sigma \to \rho, \text{ если } \sigma \text{ содержит } \forall \\ \mathrm{rk}(\rho) & \tau \equiv \sigma \to \rho, \text{ если } \sigma \text{ не содержит } \forall \\ 0 & \tau \equiv \alpha \\ \max(\mathrm{rk}(\rho), 1) & \tau \equiv \forall\alpha.\rho \end{cases}$$

Пример. Ранг экзистенциального типа:

$$\begin{aligned}
\mathrm{rk}(\exists\alpha.\beta) &= \mathrm{rk}(\forall\gamma.(\forall\alpha.\beta \to \gamma) \to \gamma) \\
&= \max(\mathrm{rk}((\forall\alpha.\beta \to \gamma) \to \gamma), 1) \\
&= \max(\max(\mathrm{rk}(\forall\alpha.\beta \to \gamma) + 1, \mathrm{rk}(\gamma)), 1) \\
&= \max(\max(2, 0), 1) = 2
\end{aligned}$$

Определение (тип в системе Хиндли-Милнера)**.**
Тип (монотип) — выражение в грамматике $\tau ::= \alpha \mid \tau \to \tau \mid (\tau)$.
Типовая схема (политип) — выражение в грамматике $\sigma ::= \tau \mid \forall\alpha.\sigma$.

$\forall\alpha.\alpha \to \alpha$ — политип, $\forall\alpha.\alpha \to \forall\beta.\beta$ — некорректный в системе Хиндли-Милнера тип.

Утверждение 3.4. $\mathrm{rk}(\tau) = 0$, $\mathrm{rk}(\sigma) = 1$.

Определение (подтип)**.** σ_1 — подтип σ_2, если существует такая подстановка $[\alpha_1 := \theta_1, \alpha_2 := \theta_2 \ldots \alpha_n := \theta_n]$, что:

1. $\sigma_1 = \forall\beta_1 \ldots \forall\beta_k.\tau[\alpha_1 := \theta_1 \ldots \alpha_n := \theta_n]$, α_i не входит свободно в θ_j
2. $\sigma_2 = \forall\alpha_1 \ldots \forall\alpha_n\tau$

Определение (система Хиндли-Милнера)**.** Грамматика:

$$\Lambda ::= x \mid \lambda x.\Lambda \mid \Lambda\Lambda \mid (\Lambda) \mid \texttt{let } x = \Lambda \texttt{ in } \Lambda$$

Будем обозначать контекст Γ без типа x как Γ_x. Правила вывода:

$$\frac{}{\Gamma \vdash x : \sigma} \text{ (Тавтология, } x : \sigma \in \Gamma) \qquad \frac{\Gamma \vdash e : \sigma}{\Gamma \vdash e : \sigma'} \text{ (Уточнение, } \sigma' \text{ — подтип } \sigma)$$

$$\frac{\Gamma \vdash e : \sigma}{\Gamma \vdash e : \forall\alpha.\sigma} \text{ (Обобщение, } \alpha \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma_x, x : \tau' \vdash e : \tau}{\Gamma \vdash \lambda x.e : \tau' \to \tau} \text{ (Абстракция)}$$

$$\frac{\Gamma \vdash e : \tau' \to \tau \qquad \Gamma \vdash e' : \tau'}{\Gamma \vdash ee' : \tau} \text{ (Применение)} \qquad \frac{\Gamma \vdash e : \sigma \qquad \Gamma_x, x : \sigma \vdash e' : \tau}{\Gamma \vdash \texttt{let } x = e \texttt{ in } e' : \tau} \text{ (Let)}$$

Мы существенно ограничили набор возможных типов в нашем языке, однако, он всё ещё вполне сильный. В нём всё ещё есть полиморфизм.

Хотя в системе Хиндли-Милнера (как и во всех рассматриваемых нами типовых системах) нельзя типизировать $\mathcal{Y}$-комбинатор, можно добавить его, расширив язык. Давайте определим его как $\mathcal{Y}\, f = f\,(\mathcal{Y}\, f)$. Какой у него должен быть тип? Пусть $\mathcal{Y}$ принимает f типа α, и возвращает нечто типа β, то есть $\mathcal{Y} : \alpha \to \beta$. Функция f должна принимать то же, что возвращает $\mathcal{Y}$, так как результат $\mathcal{Y}$ передаётся в f, и возвращать она должна то же, что возвращает $\mathcal{Y}$, так как тип выражений с обоих сторон равенства должен быть одинаковый, то есть $f : \beta \to \beta$. Кроме того, α и тип f это одно и то же, $\alpha = \beta \to \beta$. После подстановки и заключения свободной переменной под квантор получаем $\mathcal{Y} : \forall\beta.(\beta \to \beta) \to \beta$.

Через такой $\mathcal{Y}$ можно определять рекурсивные функции, и они будут типизироваться.

Вывод типа

Утверждение 3.5. *Задача вывода типа в системе Хиндли-Милнера разрешима.*

Приведём алгоритм, решающий эту задачу. Он будет принимать выражение e в контексте Γ и возвращать такую подстановку S и тип τ, что

$$S(\Gamma) \vdash e : \tau$$

В алгоритме будем пользоваться унификацией ($U(\tau_1, \tau_2)$ — унификатор τ_1 и τ_2), определим замыкание всех несвязанных типовых переменных в контексте:

$$\overline{\Gamma}(\tau) = \forall\alpha_1 \ldots \forall\alpha_n.\tau$$

где $\alpha_i \in \mathrm{FV}(\tau)$ и $\alpha_i \notin \mathrm{FV}(\Gamma)$.

Алгоритм описан в таблице 3. Если какие-то условия не могут быть соблюдены, то тип выражения не может быть выведен.

Пример. **TODO**

Вид e	Условия	$W(\Gamma, e)$
x	$x : \forall\alpha_1 \dots \alpha_k.\tau' \in \Gamma$	$S = \mathrm{id}$
	β_i — новые переменные	$\tau = \tau'[\alpha_i \coloneqq \beta_i]$
$e_1 e_2$	$W(\Gamma, e_1) = (S_1, \tau_1)$	$S = V \circ S_1 \circ S_2$
	$W(S_1(\Gamma), e_2) = (S_2, \tau_2)$	$\tau = S(\beta)$
	$U(S_2(\tau_1), \tau_2 \to \beta) = V$, β — новый тип	
$\lambda x.e$	$W(\Gamma_x \cup \{x : \beta\}, e) = (S_1, \tau_1)$	$S = S_1$
	β — новый тип	$\tau = S(\beta) \to \tau_1$
`let` $x = e_1$ `in` e_2	$W(\Gamma, e_1) = (S_1, \tau_1)$	$S = S_2 \circ S_1$
	$W(S_1\Gamma_x \cup \{x : \overline{S_1\Gamma}(\tau_1)\}, e_2) = (S_2, \tau_2)$	$\tau = \tau_2$

Таблица 3: Алгоритм W.

Вывод типа с использованием ограничений

Ой **TODO** , **TODO** .

Алгоритм W довольно прост, однако он нагромождённый. В нём мы одновременно обрабатываем выражение, решаем, каким условиям должны удовлетворять их типы, и выводим эти типы. Из общих соображений хорошего стиля программирования было бы лучше, если бы мы разделили логику алгоритма вывода типа на подпрограммы.

Давайте выделим этап построения *ограничений* на тип выражения. Для начала рассмотрим просто типизированное λ-исчисление. Грамматика ограничений:

$$C ::= \tau = \tau \mid C \wedge C \mid \exists\alpha.C$$

Алгоритм построения ограничений:

$$\begin{aligned}
[\![\Gamma \vdash x : \tau]\!] &= \Gamma(x) = \tau \\
[\![\Gamma \vdash \lambda x.e : \tau]\!] &= \exists\alpha_1\alpha_2.([\![\Gamma, x : \alpha_1 \vdash e : \alpha_2]\!] \wedge \alpha_1 \to \alpha_2 = \tau) \\
[\![\Gamma \vdash e_1 e_2 : \tau]\!] &= \exists\alpha.([\![\Gamma \vdash e_1 : \alpha \to \tau]\!] \wedge [\![\Gamma \vdash e_2 : \alpha]\!])
\end{aligned}$$

Где $\alpha_1, \alpha_2, \alpha \notin \mathrm{FV}(\Gamma, \tau)$.

Определение (эрбановский универсум)**.** Пусть c_i — константы языка, f^i — функциональные символы языка.

$$\begin{aligned}
H_0 &= \{c_1 \dots c_k \dots\} \\
H_i &= H_{i-1} \cup \left\{f^i(h_1 \dots h_t) \mid h_t \in H_{i-1}\right\}
\end{aligned}$$

Тогда $H = \cup_{i=0}^{\infty} H_i$ — эрбановский универсум.

Пусть $\varphi : T \to H$ (T — множество типов).

Теорема 3.6. *φ — решение $[\![\Gamma \vdash e : \tau]\!]$ m.u.m.m., когда $(\varphi\Gamma, \varphi\tau)$ — типизация e.*

Пример. Построим ограничения на тип выражения $\lambda x.x$:

$$\begin{aligned}
[\![\vdash \lambda x.x : \sigma]\!] &= \exists\alpha_1\alpha_2.([\![x : \alpha_1 \vdash x : \alpha_2]\!] \wedge \alpha_1 \to \alpha_2 = \sigma) \\
&= \exists\alpha_1\alpha_2.(\{x : \alpha_1\}(x) = \alpha_2 \wedge \alpha_1 \to \alpha_2 = \sigma) \\
&= \exists\alpha_1\alpha_2.(\alpha_1 = \alpha_2 \wedge \alpha_1 \to \alpha_2 = \sigma)
\end{aligned}$$

Нам тут не нравится контекст **TODO** . Давайте избавимся от него. Добавим в язык ограничений ещё две конструкции:

$$C ::= \ldots \mid x = \tau \mid \textbf{def}\ \ x : \tau\ \ \textbf{in}\ \ C$$

Теперь решением будет не только $\varphi : T \to H$, но и $\psi : X \to H$ (X — множество переменных). Решение будет удовлетворять ограничению $x = \tau$, если $\psi x = \varphi\tau$. Решение будет удовлетворять ограничению $\textbf{def}\ \ x : \tau\ \ \textbf{in}\ \ C$, если C удовлетворено при φ и $\psi[x := \varphi(\tau)]$.

Перепишем алгоритм:

$$\begin{aligned}
[\![x : \tau]\!] &= x = \tau \\
[\![\lambda x.e : \tau]\!] &= \exists\alpha_1\alpha_2.(\textbf{def}\ \ x : \alpha_1\ \ \textbf{in}\ \ [\![e : \alpha_2]\!] \wedge \alpha_1 \to \alpha_2 = \tau) \\
[\![e_1 e_2 : \tau]\!] &= \exists\alpha.([\![e_1 : \alpha \to \tau]\!] \wedge [\![e_2 : \alpha]\!])
\end{aligned}$$

Теперь расширим наш алгоритм на систему Хиндли-Милнера. Языком ограничений будет

$$\begin{aligned}
C &::= \tau = \tau \mid C \wedge C \mid \exists\alpha.C \mid x \preceq \tau \mid \textbf{def}\ \ x : \varsigma\ \ \textbf{in}\ \ C \\
\varsigma &::= \forall\overline{\alpha}\ [C]\ .\tau
\end{aligned}$$

где $x \preceq \tau$ означает, что τ — подтип типовой схемы переменной x, а ς — типовая схема.

Сам алгоритм:

$$\begin{aligned}
[\![x : \tau]\!] &= x \preceq \tau \\
[\![\lambda x.e : \tau]\!] &= \exists\alpha_1\alpha_2.(\textbf{def}\ \ x : \alpha_1\ \ \textbf{in}\ \ [\![e : \alpha_2]\!] \wedge \alpha_1 \to \alpha_2 = \tau) \\
[\![e_1 e_2 : \tau]\!] &= \exists\alpha.([\![e_1 : \alpha \to \tau]\!] \wedge [\![e_2 : \alpha]\!]) \\
[\![\texttt{let}\ \ x = e_1\ \ \texttt{in}\ \ e_2]\!] &= \textbf{let}\ \ x : \forall\alpha[[\![e_1 : \alpha]\!].\alpha\ \ \textbf{in}\ \ [\![e_2 : \tau]\!]
\end{aligned}$$

где $\textbf{let}\ \ x : \varsigma\ \ \textbf{in}\ \ C \equiv \textbf{def}\ \ x : \varsigma\ \ \textbf{in}\ \ \exists\alpha.x \preceq \alpha \wedge C$.
TODO

Рекурсивные типы

Допустим, мы хотим представить список. Все знают, что список это

```
data List a = Nil | Cons a (List a)
```

Хотелось бы написать $\texttt{list}\,\alpha = \varnothing \vee (\alpha\ \&\ \texttt{list}\,\alpha)$, однако так нельзя. Текущими выразительными средствами мы это сделать не можем.

Есть два способа решить эту проблему.

Эквирекурсивные типы

Введём аналог комбинатора неподвижной точки для типов, μ-оператор. Список можно будет записать так: $\texttt{list}\,\alpha = \mu\beta.\varnothing \vee \alpha \wedge \beta$. Будем считать, что типы $\mu\alpha.T$ и $T[\alpha := \mu\alpha.T]$ равны. На основе такого равенства сделаем отношение эквивалентности ($=_\mu$). Введём правило:

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash e : \tau_2}\ \ (\tau_1 =_\mu \tau_2)$$

Такое решение было применено в Java. Там можно писать `Enum<E extends Enum<E>>`.

Изорекурсивные типы

Будем считать типы $\mu\alpha.T$ и $T[\alpha := \mu\alpha.T]$ изоморфными. Введём две операции:

$$\begin{aligned}
\texttt{fold} &: T[\alpha := \mu\alpha.T] \to \mu\alpha.T \\
\texttt{unfold} &: \mu\alpha.T \to T[\alpha := \mu\alpha.T]
\end{aligned}$$

Разница в том, что изорекурсивные типы нужно сворачивать и разворачивать вручную, мы не считаем их равными.

Типы и указатели на них в Си — пример изорекурсивных типов.

λ-куб

А где все?

Д.Г.

Зависимые типы

Рассмотрим такой код на Си:

```
int n;
scanf("%d", &n);
int a[n];
```

Возникает вопрос, какой тип у `a`. Во-первых, это массив. Во-вторых, это массив значений тип `int`. В треьтих, он размера `n`. Хотелось бы это как-то формализовать.

Вспомним логику первого порядка. В ней есть конструкции вида $\forall x.\varphi$ и $\exists x.\varphi$, и аксиомы для них:

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x.\varphi}\ (x \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma \vdash \forall x.\varphi}{\Gamma \vdash \varphi\,[x \coloneqq \sigma]}$$

$$\frac{\Gamma \vdash \varphi\,[x \coloneqq \psi]}{\Gamma \vdash \exists x.\varphi} \qquad \frac{\Gamma \vdash \exists x.\varphi \qquad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}\ (x \notin \mathrm{FV}(\Gamma, \psi))$$

В логике второго порядка переменные под кванторами могли принимать значения любых выражений, что давало там выразительную силу. Выражения зависели от типов.

Теперь же давайте скажем, что типы могут зависеть от значений объектов. Будем обозначать неизвестный тип символом $*$. Тогда можно записать «тип» массива так: `[]` $: * \to$ `int` $\to *$. Однако, называть это типом нельзя, иначе мы бы могли писать противоречивые конструкции вида $* : * \to *$. Это *род*, обозначать мы его будем символом $\square$. Это «тип» типового конструктора. Например, $* \to$ `int` $\to * : \square$. Вместе типы и рода это *сорта*.

Сейчас мы определим исчисление, в котором типы могут зависить от значений.

Определение (обобщённая типовая система)**.** Грамматика выражения:

$$T ::= x \mid c \mid TT \mid \lambda x : T.T \mid \Pi x : T.T$$

где под выражение x подходят все переменные, под c водходят все константы.

Есть две выделенные константы для сортов: $*$ и $\square$. Пусть $s, s_1, s_2 \in \{*, \square\}$. Общие правила вывода:

$$\frac{}{\vdash * : \square}\ (\text{Аксиома})$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\ (\text{Введение}, x \notin \Gamma) \qquad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}\ (\text{Ослабление}, x \notin \Gamma)$$

$$\frac{\Gamma \vdash F : (\Pi x : A.B) \qquad \Gamma \vdash a : A}{\Gamma \vdash (Fa) : B[x \coloneqq a]}\ (\text{Применение}) \qquad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'}\ (\text{Конверсия}, B =_\beta B')$$

Следующие специальные правила параметризуются парами сортов (s_1, s_2):

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x : A.B) : s_2}\ (\Pi\text{-правило})$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash b : B \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\lambda x : A.b) : (\Pi x : A.B)}\ (\lambda\text{-правило})$$

Π — это обобщение $\to$. $\Pi x : a.b$, где b зависит от x, это множество пар, каждая из которых сопоставляет типу a тип b. В частности, если $\varphi = \Pi x : \psi.\sigma$, и $x \notin \mathrm{FV}(\sigma)$, то аналогичной записью будет $\varphi = \psi \to \sigma$.

Рассмотрение обобщённой типовой системы мы оставим на потом, а пока рассмотрим ту же систему, в которой специальные правила ограничены парами $(s_1, s_2) \in \{(*, *), (*, \square)\}$. Такая система называется λP. Правила с $(s_1, s_2) = (*, *)$ это обычные правила для λ-выражений. Правила с $(s_1, s_2) = (*, \square)$ это правила для типов, зависящих от значений (λ-выражений).

В системе λP тип массива из `int` можно записать так: `int[]` $\equiv \Pi$`x` : `int`.`int[x]`. Мы явно указали, что тип зависит от значения типа `int`. То есть тип `int[]` это набор всех возможных пар из размера массива и массива соответствующей длины. Тогда если `int a[5]`, то `a` : `int[5]`.

Пример. Типизируем $\lambda x.x$ (опуская $\vdash * : \square$):

$$\dfrac{\dfrac{}{a : * \vdash a : *} \qquad \dfrac{\dfrac{}{x : a \vdash x : a}}{a : *, x : a \vdash x : a} \qquad \dfrac{\dfrac{}{a : * \vdash a : *}}{a : *, x : a \vdash a : *}}{a : * \vdash (\lambda x : a.x) : (\Pi x : a.a)}$$

TODO ещё надо

Обобщённая типовая система и λ-куб

Ранее мы рассматривали систему F, в которой был полиморфизм за счёт выражений, зависящих от типов. Нам ничего не мешает совместить полиморфизм с зависимыми типами. Кроме того, для формализации массивов из Си нам всё ещё не хватает типов, которые зависили бы от других типов. Их мы также можем добавить.

λ-куб — это систематизация этих трёх не зависящих друг от друга расширений просто типизированного λ-исчисления. Позволяя или запрещая различные пары сортов (s_1, s_2) можно получать разные типовые системы:

$\lambda\to$	$(*, *)$			
$\lambda 2$	$(*, *)$	$(\square, *)$		
$\lambda\underline{\omega}$	$(*, *)$		$(\square, \square)$	
$\lambda\omega$	$(*, *)$	$(\square, *)$	$(\square, \square)$	
λP	$(*, *)$			$(*, \square)$
$\lambda P2$	$(*, *)$	$(\square, *)$		$(*, \square)$
$\lambda P\underline{\omega}$	$(*, *)$		$(\square, \square)$	$(*, \square)$
$\lambda P\omega = \lambda C$	$(*, *)$	$(\square, *)$	$(\square, \square)$	$(*, \square)$

Эти пары имеют следующий смысл:

$(*, *)$	разрешены правила для выражений, зависящих от выражений
$(\square, *)$	разрешены правила для выражений, зависящих от типов
$(*, \square)$	разрешены правила для типов, зависящих от выражений
$(\square, \square)$	разрешены правила для типов, зависящих от типов

Система F, например, в таблице обозначена как $\lambda 2$. Системы, в которых мы можем писать нечто вроде `[]` : $* \to$ `int` $\to *$ это $\lambda P\underline{\omega}$ и λC.

Наглядно такую классификацию обычно представляют в виде куба (рисунок 4).

Пример. В $\lambda\to$ выводимо $A : * \vdash \Pi x : A.A : *$ (или $A \to A : *$):

$$\dfrac{\dfrac{\dfrac{}{\vdash * : \square}\ (\text{Аксиома})}{A : * \vdash A : *}\ (\text{Введение}) \qquad \dfrac{\dfrac{\dfrac{}{\vdash * : \square}\ (\text{Аксиома})}{A : * \vdash A : *}\ (\text{Введение}) \qquad \dfrac{\dfrac{}{\vdash * : \square}\ (\text{Аксиома})}{A : * \vdash A : *}\ (\text{Введение})}{A : *, x : A \vdash A : *}\ (\text{Ослабление})}{A : * \vdash \Pi x : A.A : *}\ (\text{П-правило})$$

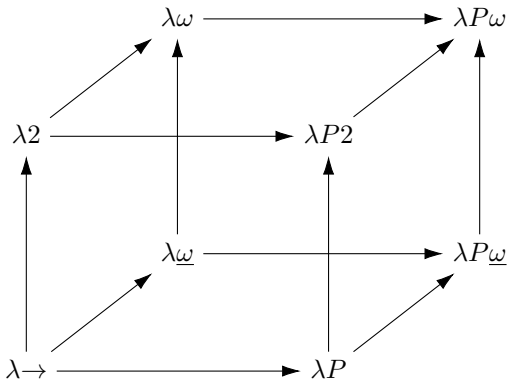

Рис. 4: λ-куб.

Выведем $A : *, b : A \vdash ((\lambda a : A.a)b) : A$ (из-за большого объёма две ветки вывода опущены):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash * : \square}\text{ (Аксиома)}}{A : * \vdash A : *}\text{ (Введение)} \qquad \dfrac{\dfrac{\dfrac{}{\vdash * : \square}\text{ (Аксиома)}}{A : * \vdash A : *}\text{ (Введение)}}{A : *, b : A \vdash b : A}\text{ (Введение)}}{A : *, b : A \vdash A : *}\text{ (Ослабление)}}{A : *, b : A, a : A \vdash a : A}\text{ (Введение)} \tag{1}$$

$$\dfrac{\dfrac{\dfrac{\text{(ослабление)}}{A : *, b : A \vdash A : *} \qquad \dfrac{(1)}{A : *, b : A, a : A \vdash a : A}}{A : *, b : A \vdash (\lambda a : A.a) : (\Pi x : A.A)} \qquad \dfrac{\dfrac{\text{(два ослабления)}}{A : *, b : A, a : A \vdash A : *} \qquad \dfrac{\text{(ослабление)}}{A : *, b : A \vdash b : A}}{A : *, b : A \vdash ((\lambda a : A.a)b) : A}$$

Два неуказаных правила это применение и λ-правило.
TODO

Линейные и уникальные типы

Пусть $A \to_\beta A'$. С одной стороны, порядок редукции не важен, $(\lambda x.xx)A \to_\beta (\lambda x.xx)A' \to_\beta A'A'$ и $(\lambda x.xx)A \to_\beta AA \to_\beta A'A \to_\beta A'A'$. По теореме Чёрча-Россера нормальная форма единственна, если существует. С другой стороны, реальный мир на самом деле не таков, в нём есть побочные эффекты.

Комбинаторная логика

Историческая справка. Известные нам комбинаторы $\mathcal{S}$ и $\mathcal{K}$ придумал Моисей Шейнфинкель, ещё до Карри с его λ-исчислением. У него было своё каррирование и комбинаторная логика. Хаскелл Карри придумал свою систему позже, однако независимо от Шейнфинкеля.

Моисей Шейнфинкель			Хаскелл Карри	
$\mathcal{I}$	(**I**dentität)	$\lambda x.x$	$\mathcal{B}$	$\lambda xyz.x(yz)$
$\mathcal{K}$	(**K**onstanz)	$\lambda xy.x$	$\mathcal{C}$	$\lambda xyz.xzy$
$\mathcal{S}$	(Ver**s**chmelzung)	$\lambda xyz.xz(yz)$	$\mathcal{K}$	$\lambda xy.x$
$\mathcal{T}$	(Ver**t**auschung)	$\lambda xyz.xzy$	$\mathcal{W}$	$\lambda xy.xyy$

С помощью каждой из этих двух систем можно выразить любое λ-выражение без свободных переменных. Пусть

$$\Lambda_{\mathcal{SK}} ::= x \mid \mathcal{S} \mid \mathcal{K} \mid \mathcal{I} \mid (\Lambda_{\mathcal{SK}}\Lambda_{\mathcal{SK}})$$

Докажем 1.11. Определим $\mathrm{T} : \Lambda \to \Lambda_{\mathcal{SK}}$:

$$\begin{aligned}
\mathrm{T}[x] &= x \\
\mathrm{T}[AB] &= \mathrm{T}[A]\,\mathrm{T}[B] \\
\mathrm{T}[\lambda x.P] &= \mathcal{K}\,\mathrm{T}[P], \text{ если } x \notin \mathrm{FV}(P) \\
\mathrm{T}[\lambda x.x] &= \mathcal{I} \\
\mathrm{T}[\lambda x.AB] &= \mathcal{S}\,\mathrm{T}[\lambda x.A]\,\mathrm{T}[\lambda x.B] \\
\mathrm{T}[\lambda x.\lambda y.A] &= \mathrm{T}[\lambda x.\,\mathrm{T}[\lambda y.A]]
\end{aligned}$$

$\mathrm{T}[\lambda$-выражение$]$ завершается и не содержит абстракций. Можно проверить, что $\mathrm{T}[A] =_\beta A$. □

Например, $\overline{4}$ после преобразования приведённым алгоритмом, очевидно, будет выглядеть так:

$$\overline{4} =_\beta \mathcal{S}(\mathcal{S}(\mathcal{K}\mathcal{S})(\mathcal{S}(\mathcal{K}\mathcal{K})\mathcal{I}))(\mathcal{S}(\mathcal{S}(\mathcal{K}\mathcal{S})(\mathcal{S}(\mathcal{K}\mathcal{K})\mathcal{I}))(\mathcal{S}(\mathcal{S}(\mathcal{K}\mathcal{S})(\mathcal{S}(\mathcal{K}\mathcal{K})\mathcal{I}))(\mathcal{S}(\mathcal{S}(\mathcal{K}\mathcal{S})(\mathcal{S}(\mathcal{K}\mathcal{K})\mathcal{I}))(\mathcal{K}\mathcal{I}))))$$

Для доказательства аналогичной теоремы для базиса $\mathcal{BCKW}$ достаточно выразить через них комбинаторы $\mathcal{S}$ и $\mathcal{K}$:

$$\mathcal{S} = \mathcal{B}(\mathcal{B}\mathcal{W})(\mathcal{B}\mathcal{B}\mathcal{C}) \qquad \mathcal{I} = \mathcal{C}\mathcal{K}\mathcal{K}$$

Давайте теперь проследим связь комбинаторов с логикой. Выведем типы у $\mathcal{S}$ и $\mathcal{K}$:

$$\begin{aligned}
&\mathcal{S} = \lambda xyz.xz(yz) : (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to (\alpha \to \gamma) \\
&\mathcal{K} = \lambda xy.x : \alpha \to \beta \to \alpha
\end{aligned}$$

Это похоже на вторую и первую схемы аксиом в ИИВ. Понятно, что изоморфизм Карри-Ховарда переносится и на комбинаторы, так как это сокращения для λ-выражений. Роль modus ponens будет исполнять редукция. Посмотрим на другие комбинаторы:

$$\begin{aligned}
\mathcal{I} &= \lambda x.x &&: \alpha \to \alpha \\
\mathcal{B} &= \lambda xyz.x(yz) &&: (\alpha \to \beta) \to (\beta \to \gamma) \to (\alpha \to \gamma) \\
\mathcal{C} &= \lambda xyz.xzy &&: (\alpha \to \beta \to \gamma) \to (\beta \to \alpha \to \gamma) \\
\mathcal{K} &= \lambda xy.x &&: \alpha \to \beta \to \alpha \\
\mathcal{W} &= \lambda xy.xyy &&: (\alpha \to \alpha \to \beta) \to (\alpha \to \beta)
\end{aligned}$$

У последних двух комбинаторов есть отличительная особенность. $\mathcal{K}$ «убивает» один аргумент, а $\mathcal{W}$ — дублирует. В логике это не даёт ничего плохого, однако в λ-исчислении такой эффект может быть нежелателен. Ведь не всегда в реальных программах мы можем произвольным образом дублировать какие-то объекты. Базис $\mathcal{BCKWI}$ по изоморфизму Карри-Ховарда порождает интуиционистскую логику. Можно рассматривать исчисления, прорджённые базисами $\mathcal{BCKI}$ и $\mathcal{BCI}$.

Прежде чем описывать новую логику, мы вспомним классическое ИИВ, однако представим правила вывода в немного другом виде:

$$\frac{}{A \vdash A} \qquad \frac{\Gamma, \Delta \vdash A}{\Delta, \Gamma \vdash A} \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B} \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash A \to B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \times B} \qquad \frac{\Gamma \vdash A \times B \qquad \Delta, A, B \vdash C}{\Gamma, \Delta \vdash C}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A + B} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A + B} \qquad \frac{\Gamma \vdash A + B \qquad \Delta, A \vdash C \qquad \Delta, B \vdash C}{\Gamma, \Delta \vdash C}$$

Линейные высказывания

Определение (Грамматика линейных высказываний)**.**

$$T ::= x \mid T \multimap T \mid T \otimes T \mid T \,\&\, T \mid T \oplus T \mid !T$$

Неформально, этим значкам можно предать следующий смысл (формально — смысла в этом нет):

1. $A \multimap B$ — Имеем возможноть превратить A в B
2. $A \otimes B$ — Можем получить и A, и B, одновременно
3. $A \,\&\, B$ — Можем получить либо A, либо B, по своему усмотрению
4. $A \oplus B$ — Можем получить либо A, либо B, но не по своему усмотрению
5. $!A$ — Имеем фабрику, производящая неограниченное количество A

Расширим язык новыми правилами работы с контекстом — заведем два вида контекстов: $\langle A \rangle$ — линейный, $[A]$ — интуиционистский. Если у мета-переменной контекста нет каких-либо скобок, то вид контекста нам не важен и может быть любым. Разница заключается в том, что на выражанения в линейном контексте налагаются новые правила, говорящие что мы не можем раздвоить или убрать в контексте эти выражения. Формально это отображается в следующих аксиомах:

$$\frac{}{\langle A \rangle \vdash A} \qquad \frac{}{[A] \vdash A} \qquad \frac{\Gamma, \Delta \vdash A}{\Delta, \Gamma \vdash A} \qquad \frac{\Gamma, [A], [A] \vdash B}{\Gamma, [A] \vdash B} \qquad \frac{\Gamma \vdash B}{\Gamma, [A] \vdash B}$$

Схема аксиом в ИИВ	Представление в линейной логике
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$	$\dfrac{\dfrac{}{\langle !A\rangle \vdash !A} \qquad \Gamma, [A] \vdash B}{\Gamma, \langle !A\rangle \vdash B}$ / $\Gamma \vdash !A \multimap B$
$\dfrac{\Gamma \vdash A \to B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B}$	$\dfrac{\Gamma \vdash !A \multimap B \qquad \dfrac{[\Delta] \vdash A}{[\Delta] \vdash !A}}{\Gamma, [\Delta] \vdash B}$

Таблица 4: Примеры перевода аксиом ИИВ в линейную логику.

Заведем также аксиомы для работы с самими выражениями:

$$\frac{[\Gamma] \vdash A}{[\Gamma] \vdash !A} \qquad \frac{\Gamma \vdash !A \qquad \Delta, [A] \vdash B}{\Gamma, \Delta \vdash B} \qquad \frac{\Gamma, \langle A\rangle \vdash B}{\Gamma \vdash A \multimap B} \qquad \frac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \qquad \frac{\Gamma \vdash A \otimes B \qquad \Delta, \langle A\rangle, \langle B\rangle \vdash C}{\Gamma, \Delta \vdash C}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B} \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash B}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B} \qquad \frac{\Gamma \vdash A \oplus B \qquad \Delta, \langle A\rangle \vdash C \qquad \Delta, \langle B\rangle \vdash C}{\Gamma, \Delta \vdash C}$$

Пример (законы Де-Моргана). В линейных высказываниях также верны законы Де-Моргана :

$$\langle !(A \,\&\, B)\rangle \vdash !A \otimes !B \qquad \langle !A \otimes !B\rangle \vdash !(A \,\&\, B)$$

Докажем один из них:

$$\dfrac{\dfrac{}{\langle !(A \,\&\, B)\rangle \vdash !(A \,\&\, B)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{[A \,\&\, B] \vdash A \,\&\, B}}{[A \,\&\, B] \vdash A}}{[A \,\&\, B] \vdash !A} \qquad \dfrac{\dfrac{\dfrac{}{[A \,\&\, B] \vdash A \,\&\, B}}{[A \,\&\, B] \vdash B}}{[A \,\&\, B] \vdash !B}}{[A \,\&\, B, A \,\&\, B] \vdash !A \otimes !B}}{\langle !(A \,\&\, B)\rangle \vdash !A \otimes !B}$$

Интуционисткую логику можно вложить в линейную логику путем определения интуционистских связок через линейные:

$$A \to B = !A \multimap B$$
$$A \times B = A \,\&\, B$$
$$A + B = !A \oplus !B$$

В качестве альтернативного варианта, $A \times B$ можно вложить как $!A \otimes !B$. Доказательства из ИИВ могут быть переписаны на язык линейной логики путем замены аксиомы ИИВ их аналогами из линенйной логики.

Линейные типы

Определим грамматику для нашего λ-исчисления:

$$\begin{aligned} s ::= {} & x \\ & \mid \lambda \langle x \rangle . s \mid s \langle s \rangle \\ & \mid\ !s \mid \text{case } s \text{ of } !x \to s \\ & \mid \langle s, s \rangle \mid \text{case } s \text{ of } \langle x, x \rangle \to s \\ & \mid \langle\!\langle s, s \rangle\!\rangle \mid \text{fst} \langle s \rangle \mid \text{snd} \langle s \rangle \\ & \mid \text{inl} \langle s \rangle \mid \text{inr} \langle s \rangle \mid \text{case } s \text{ of } \text{inl} \langle x \rangle \to s;\ \text{inr} \langle x \rangle \to s \end{aligned}$$

Будем типизировать это λ-исчисление линейными высказываниями. Выпишем аксиомы:

$$\frac{[\Gamma] \vdash s : A}{[\Gamma] \vdash\ !s :\ !A} \qquad \frac{\Gamma \vdash s :\ !A \qquad \Delta, [x : A] \vdash t : B}{\Gamma, \Delta \vdash \text{case } s \text{ of } !x \to t : B}$$

$$\frac{\Gamma \vdash s : A \qquad \Delta \vdash t : B}{\Gamma, \Delta \vdash \langle s, t \rangle : A \otimes B} \qquad \frac{\Gamma \vdash s : A \otimes B \qquad \Delta, \langle x : A \rangle, \langle y : B \rangle \vdash t : C}{\Gamma, \Delta \vdash \text{case } s \text{ of } \langle x, y \rangle \to t : C}$$

$$\frac{\Gamma \vdash s : A \qquad \Gamma \vdash t : B}{\Gamma \vdash \langle\!\langle s, t \rangle\!\rangle : A \,\&\, B} \qquad \frac{\Gamma \vdash s : A \,\&\, B}{\Gamma \vdash \text{fst} \langle s \rangle : A} \qquad \frac{\Gamma \vdash s : A \,\&\, B}{\Gamma \vdash \text{snd} \langle s \rangle : B}$$

$$\frac{\Gamma, \langle x : A \rangle \vdash u : B}{\Gamma, \lambda \langle x \rangle . u : A \multimap B} \qquad \frac{\Gamma \vdash s : A \multimap B \qquad \Delta \vdash t : A}{\Gamma, \Delta \vdash s \langle t \rangle : B}$$

$$\frac{\Gamma \vdash s : A}{\Gamma \vdash \text{inl} \langle s \rangle : A \oplus B} \qquad \frac{\Gamma \vdash s : B}{\Gamma \vdash \text{inr} \langle s \rangle : A \oplus B}$$

$$\frac{\Gamma \vdash s : A \oplus B \qquad \Delta, \langle x : A \rangle \vdash t : C \qquad \Delta, \langle y : B \rangle \vdash v : C}{\Gamma, \Delta \vdash \text{case } s \text{ of } \text{inl} \langle x \rangle \to t;\ \text{inr} \langle y \rangle \to v : C}$$

Вложим просто типизируемое λ-исчисление в линейные типы.

$$\begin{aligned} \lambda x.s &= \lambda \langle x' \rangle . \text{case } x' \text{ of } !x \to s \\ st &= s \langle !t \rangle \end{aligned}$$

Пример (Ломаем линейные типы). Пусть у нас есть выражения f, g типа $A \multimap B$. Возьмем $h = \lambda x. \langle f \langle x \rangle, g \langle x \rangle \rangle :\ !A \multimap B \otimes B$. Докажем, что h действительно существует в линейных типах:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[f : A \multimap B] \vdash f : A \multimap B}{[x : A], [f : A \multimap B] \vdash f \langle x \rangle : B} \qquad \dfrac{[g : A \multimap B] \vdash g : A \multimap B}{[x : A], [g : A \multimap B] \vdash g \langle x \rangle : B}}{[x : A], [f : A \multimap B], [x : A], [g : A \multimap B] \vdash \langle f \langle x \rangle, g \langle x \rangle \rangle : B \otimes B}}{\vdots}}{\langle x' :\ !A \rangle \vdash x' : A \qquad [f : A \multimap B], [g : A \multimap B], [x : A] \vdash \langle f \langle x \rangle, g \langle x \rangle \rangle : B \otimes B}}{\langle x' :\ !A \rangle, [f : A \multimap B], [g : A \multimap B] \vdash \text{case } x' \text{ of } !x \to \langle f \langle x \rangle, g \langle x \rangle \rangle : B \otimes B}}{[f : A \multimap B], [g : A \multimap B], \langle x' :\ !A \rangle \vdash \text{case } x' \text{ of } !x \to \langle f \langle x \rangle, g \langle x \rangle \rangle : B \otimes B}}{[f : A \multimap B], [g : A \multimap B] \vdash \lambda x. \langle f \langle x \rangle, g \langle x \rangle \rangle :\ !A \multimap B \otimes B}$$

В функции h обьект A был интуционистким, но функции f и g работают с ним, как с линейным. Как мы видим линейный тип не дает нам размножать обьект (получить несколько ссылок на него в разных местах), но при этом у нас нет гарантий того, что до этого этот обьект не был размножен.

Уникальные типы

Определение (Род)**.** Родом будем называть выражение, удовлетворяющие следующей граммтике:

$$\begin{array}{lll} \kappa ::= \mathcal{T} & & \text{Базовый тип} \\ \mid \mathcal{U} & & \text{Аттрибут уникальности} \\ \mid * & & \text{Тип} \\ \mid \kappa \to \kappa & & \end{array}$$

Определение (Типовые константы)**.**

`Int`, `Bool`	::	$\mathcal{T}$	Базовые типы
$\to$	::	$* \to * \to \mathcal{T}$	Конструктор функций
$\bullet, \times$	::	$\mathcal{U}$	Уникальный, не уникальный
$\vee, \wedge$	::	$\mathcal{U} \to \mathcal{U} \to \mathcal{U}$	Дизьюнкия, коньюнкция аттрибутов
$\neg$	::	$\mathcal{U} \to \mathcal{U}$	Отрицание аттрибута
`Attr`	::	$\mathcal{T} \to \mathcal{U} \to *$	Конструктор типа

С помощью типовых констант мы можем делать различные типы разных родов. Тип (рода $*$) состоит из базового типа (рода $\mathcal{T}$) и аттрибута уникальности (это тоже тип, только рода $\mathcal{U}$). Аттрибут показывает обязан ли обьект этого типа быть уникальным (всего на этот обьект может существовать только одна ссылка) — $\bullet$, или это не обязательно — $\times$. На аттрибутах заданы различные булевы операции, которые выполняются так, будто $\bullet \equiv true$, $\times \equiv false$.

Немного сахара:

$$\texttt{Attr}\ t\ \ u \equiv t^u \qquad \texttt{Attr}\ (a \to b)\ \ u \equiv a \xrightarrow{u} b$$

Грамматика выражений будет похожа на классическую грамматику выражений, за исключением того, что у переменных будет указываться аттрибут ($x^{\odot}$), если на переменную есть только 1 ссылка, или аттрибут ($x^{\otimes}$), если на переменную имеется несколько ссылок.

$$e ::= \lambda x.e \mid ee \mid x^{\odot} \mid x^{\otimes}$$

В доказательствах будем использовать следующую нотацию: $\Gamma \vdash e : \tau|_{fv}$. Γ и fv отображают переменные в типы. Разница заключается в том, что Γ отображает в типы рода $*$, а fv отображает в типы рода $\mathcal{U}$.

Введём правила вывода. Эти правила похожи на правила из системы типов Хиндли-Милнера, за исключением того, что они будут также нести информацию о уникальности типов и переменных.

$$\frac{}{\Gamma, x : t^u \vdash x^{\odot} : t^u|_{x:u}}\ \texttt{Var}^{\odot} \qquad \frac{}{\Gamma, x : t^{\times} \vdash x^{\otimes} : t^{\times}|_{x:\times}}\ \texttt{Var}^{\otimes}$$

В обоих правилах мы вводим в контекст новую переменную. В $\texttt{Var}^{\odot}$ мы вводим уникальную перемменную, а в $\texttt{Var}^{\otimes}$ — расшаренную. Заметим, что в $\texttt{Var}^{\odot}$ хоть переменная и уникальна, тип ее может быть и не уникальным, так как потом эта переменная может перестать быть уникальной (например, её скопировали в процессе программы). В $\texttt{Var}^{\otimes}$ тип переменной обязан быть не уникальным.

$$\frac{\Gamma, x : a \vdash e : b|_{fv} \qquad fv' \equiv \triangleright_x fv}{\Gamma \vdash \lambda x.e : a \xrightarrow{\vee fv'} b|_{fv'}}\ \texttt{Abs}$$

Операция $\triangleright_x fv$ выкидывает из отображения fv все пары $x : \mathcal{U}$. Запись $\vee fv'$ означает дизьюнкцию всех типов из отобораения fv'.

Рассмотрим правило `Abs`. В замыкании функции $\lambda x.e$ находятся все свободные переменные из выражения e, за исключением связанной переменной x. Если в замыкании функции находится уникальная переменная, то функция должна быть уникальна, следовательно, атрибут уникальности функции вычисляется так: $\vee \left(\triangleright_x fv\right)$, что и написано в правиле.

$$\frac{\Gamma \vdash e_1 : a \xrightarrow{u} b|_{fv_1} \qquad \Gamma \vdash e_2 : a|_{fv_2}}{\Gamma \vdash e_1 e_2 : b|_{fv_1 \cup fv_2}}\ \texttt{App}$$

Тут обычная аппликация Хиндли-Милнера, в которой мы запоминаем уникальность свободных переменных из обоих ветвей. Уникальность функции нам не интересна так как мы можем вычислять функции вне зависимости от их уникальности.

Пример. Рассмотрим функцию $\texttt{dup} \equiv \lambda x.\, \langle x^{\otimes}, x^{\otimes} \rangle$ типа $a^{\times} \xrightarrow{\times} a^{\times}\ \&^{u}\ a^{\times}$. Покажем, что такая функция существует в нашей системе типов:

$$\dfrac{\dfrac{\dfrac{}{x : a^{\times} \vdash x^{\otimes} : x : \times}\ \texttt{Var}^{\otimes} \qquad \dfrac{}{x : a^{\times} \vdash x^{\otimes} : x : \times}\ \texttt{Var}^{\otimes}}{x : a^{\times} \vdash \langle x^{\otimes}, x^{\otimes} \rangle : a^{\times}\ \&^{u}\ a^{\times} | x : \times}\ \texttt{Abs}}{\vdash \lambda x.\, \langle x^{\otimes}, x^{\otimes} \rangle : a^{\times} \xrightarrow{\times} a^{\times}\ \&^{u}\ a^{\times} |}$$

Так как функция `dup` дублирует аргументы, она принимает переменную неуникального типа. Компилятор это может легко проверить, так как в теле функции есть 2 ссылки на x, и именно поэтому она помечена как $\otimes$. Также функция `dup` полиморфна по аттрибуту уникальности возвращаемого значения, так как мы можем попросить у функции как и уникальную пару, так и расшаренную.

Пример. Давайте по другому определим функцию $\texttt{dup}'\ x \equiv (\lambda f.\, \langle f^{\otimes}\ \bot, f^{\otimes}\ \bot \rangle)\, (\texttt{const}\ x^{\odot})$. Очевидно, она делает тоже самое, что и функция `dup`.

Давайте установим тип функции `const`. Сначала запишем ее тип без указания аттрибутов уникальности над стрелочками: $t^{u} \to s^{v} \to t^{u}$. Если мы частично применим функцию `const`, передав в нее переменную уникального типа, то мы должны получить функцию $(\texttt{const}\ x)$ уникального типа. Если бы это было не так, то мы смогли бы полученную функцию применить в нескольких местах, и, следственно, смогли получить несколько ссылок на результат функции. Но это противоречит типовой системе, так как результат (он же первый аргумент исходной функции) — уникален.

Значит, полный тип функции `const` равен $t^{u} \xrightarrow{\times} s^{v} \xrightarrow{u} t^{u}$.

В первой части функции $\texttt{dup}'$ лямбда-выражение принимает расшаренную функцию. То есть функция $(\texttt{const}\ x)$ не уникальна. Значит тип переменной x не уникален. Это является примером того, когда уникальная переменная имеет неуникальный тип.

Теорема Диаконеску и сетоиды

Теорема Диаконеску

Believe me.

Теорема 6.1. *Аксиома выбора влечёт закон исключённого третьего.*

Доказательство. Для любого утверждения P по аксиоме выделения мы можем построить два множества:

$$A = \{x \in \{0,1\} \mid (x = 0) \vee P\} \qquad B = \{x \in \{0,1\} \mid (x = 1) \vee P\}$$

По аксиоме выбора мы знаем, что их декартово произведение непусто. Иначе говоря, существует фукнция $f : \{A, B\} \to \{0,1\}$, что

$$f(A) \in A \,\&\, f(B) \in B$$

Это, по определению двух множеств, эквивалентно

$$(f(A) = 0 \vee P) \,\&\, (f(B) = 1 \vee P)$$

Из этого следует, что

$$(f(A) \neq f(B)) \vee P \tag{$*$}$$

Однако, по принципу объёмности $P \to (A = B)$. Значит, $P \to (f(A) = f(B))$. Значит,

$$(f(A) \neq f(B)) \to \neg P \tag{$**$}$$

Из $*$ и $**$ можно вывести $P \vee \neg P$. □

Сетоид

Определение.

$$\langle C : \mathrm{Type}, \mathrm{Eq} : C \to C \to \mathrm{Type}, P : \mathrm{IsEquivalence\ Eq} \rangle$$

Eq — отношение эквивалентности, P — доказательство этого факта.

Определение (экстенциональность)**.**

$$f : \langle A, =_A, P_A \rangle \to \langle B, =_B, P_B \rangle$$

f экстенсиональна (Ext f), если из $x =_A y$ следует $fx =_B fy$

Пример сетоида — целые числа:

$$\left\langle \langle p : \mathrm{Nat}, n : \mathrm{Nat} \rangle, \overline{Eq(a + d = b + c)}, \mathrm{IsEquivalence}Eq \right\rangle$$

Интенциональное исчисление — это исчисление, равенство в котором является структурным.

Экстенциональное исчисление — это исчисление, равенство в котором задаётся отношением эквивалентности.