

Билеты к экзамену по функциональному программированию

Тамарин Вячеслав

January 28, 2021

1 Сравнение функционального и императивного подходов к программированию

1.1 Императивное программирование

Вычисление (программа) описывается в терминах **инструкций**, изменяющих **состояние** вычислителя.

- Инструкции выполняются последовательно;
- Состояние изменяется инструкциями **присваивания** значений изменяемым переменным;
- Если механизм условного исполнения (if, switch);
- Инструкции можно повторять с помощью циклов (while, for);
- Типы данных описываются с оглядкой на их физическое представление в памяти.

Такой стиль иногда называют стилем фон Неймана.

```
1 long factorial(int n) {  
2     long res = 1;  
3     for (int i = 0; i < n; i++)  
4         res *= i;  
5     return res;  
6 }
```

Выполнение программы — переход вычислителя из начального состояния в конечное с помощью последовательных инструкций.

Часть конечного состояния может интерпретироваться как результат вычислений.

1.2 Функциональное программирование

Функциональная программа — **выражение**, ее выполнение — вычисление (**редукция**) этого выражения.

```
1 factorial n = if n == 0 then 1 else n * factorial(n-1)
```

Выполнение программы — редукция выражения с помощью **подстановки** определений функций в места их „вызова” с заменой формальных параметров на фактические.

```
1 factorial 3  
2 -> if 3 == 0 then 1 else 3 * factorial (3 - 1)  
3 -> ...  
4 -> 3 * factorial (3 - 1)  
5 -> 3 * if (3 - 1) == 0 then 1 else (3 - 1) * factorial ((3 - 1) - 1)  
6 -> ...  
7 -> 3 * 2 * 1 * 1  
8 -> 6
```

- Нет состояний — нет изменяемых переменных;
- Нет переменных — нет присваивания;
- Нет циклов, так как нет различий между итерациями – состояниями
- Последовательность не важна, поскольку выражения независимы.
- Рекурсия — замена циклов;
- Функции высших порядков;
- Сопоставление с образцом;
- Все функции — чистые.

1.3 Сильные стороны ФП

- Регулярный синтаксис, удобство анализа кода;
- Мощная типизация, при этом можно практически не использовать типы в коде за счет эффективным алгоритмам вывода типов;
- Возможность генерации программ по набору свойств;
- Эффективная доказуемость свойств программ алгебраическими методами;
- Высокоурвневые оптимизации на базе эквивалентных преобразований.

2 Основы λ -исчисления. λ -термы, свободные и связанные переменные.

λ -исчисление — формальная система, лежащая в основе ФП. Разработано Алонзо Чёрчем в 1930-х для формализации и анализа понятия вычислимости.

В λ -исчислении тремя основными понятиями являются:

- применение (аппликация, application) — задает синтаксис применения функции к ее фактическим аргументам. Применение функции F к аргументу X записывается как FX ;
- лямбда-абстракция (abstraction) — описывает синтаксис определения функции на основе параметризованного выражения, представляющего ее тело. Абстракция по x : $\lambda x. F$;
- редукция (reduction) — определяет отношение вычисления, основывающегося на подстановке фактических параметров вместо формальных.

Определение 1: λ -термы

Множество λ -термов Λ индуктивно строится из переменных $V = \{x, y, z, \dots\}$ с помощью применения и абстракции:

$$\begin{aligned} x \in V &\implies x \in \Lambda \\ M, N \in \Lambda &\implies (MN) \in \Lambda \\ M \in \Lambda, x \in V &\implies (\lambda x. M) \in \Lambda \end{aligned}$$

В абстрактном синтаксисе

$$\Lambda ::= V \mid (\Lambda\Lambda) \mid (\lambda V. \Lambda).$$

Терм может быть *переменной* (например, $x \in \Lambda$), *аппликацией* (например, $(xy) \in \Lambda$) или *лямбда-абстракцией* (например, $\lambda x. ((x(yx))) \in \Lambda$).

Определение 2: Подтермы

Множество **подтермов** терма Q определяется индуктивно:

$$\begin{aligned}\text{subterms}(x) &= \{x\} \\ \text{subterms}(MN) &= \{MN\} \cup \text{subterms}(M) \cup \text{subterms}(N) \\ \text{subterms}(\lambda x. M) &= \{\lambda x. M\} \cup \text{subterms}(M)\end{aligned}$$

2.1 Соглашения

Общеприняты следующие соглашения для термов:

- Внешние скобки опускаются

- Применение ассоциативно *влево*:

$$FXYZ \equiv (((FX)Y)Z).$$

- Абстракция ассоциативна *вправо*:

$$\lambda x y z. M \equiv (\lambda x. (\lambda y. (\lambda z. M))).$$

- Тело абстракции простирается вправо насколько это возможно:

$$\lambda x. MNK \equiv \lambda x. (MNK).$$

Определение 3: Свободные переменные

Множество **свободных переменных** в терме Q ($FV(Q)$) определяется следующим образом:

$$\begin{aligned}FV(x) &= \{x\} \\ FV(MN) &= FV(M) \cup FV(N) \\ FV(\lambda x. M) &= FV(M) \setminus \{x\}\end{aligned}$$

Определение 4: Связанные переменные

Множество **связанных переменных** в терме Q ($BV(Q)$) определяется следующим образом:

$$\begin{aligned}BV(x) &= \emptyset \\ BV(MN) &= BV(M) \cup BV(N) \\ BV(\lambda x. M) &= BV(M) \cup \{x\}\end{aligned}$$

2.2 Комбинаторы

Определение 5: Комбинатор

M — **замкнутый λ -терм (комбинатор)**, если $FV(M) = \emptyset$.

Множество замкнутых λ -термов обозначается Λ^0 .

1. $I = \lambda x. x$
2. $\omega = \lambda x. xx$
3. $\Omega = \omega\omega = (\lambda x. xx) (\lambda x. xx)$
4. $K = \lambda xy. x$
5. $K_* = \lambda xy. y$
6. $C = \lambda fxy. fyx$
7. $B = \lambda fgx. f(gx)$
8. $S = \lambda fgx. fx(gx)$

3 Подстановка λ-терма. Лемма подстановки.

Определение 6: Подстановка

Подстановка терма N вместо свободных вхождений переменной x в терм M ($[x \mapsto N]M$) задается следующими правилами:

$$\begin{aligned} [x \mapsto N]x &= N \\ [x \mapsto N]y &= y \\ [x \mapsto N]PQ &= ([x \mapsto N]P) ([x \mapsto N]Q) \\ [x \mapsto N]\lambda x. P &= \lambda x. P \\ [x \mapsto N]\lambda y. P &= \lambda y. [x \mapsto N]P, & \text{если } y \notin FV(N) \\ [x \mapsto N]\lambda y. P &= \lambda x. [x \mapsto N]([y \mapsto z]P), & \text{если } y \in FV(N) \end{aligned}$$

Предполагается, что x и y различны, а z — „свежая“ переменная, то есть $z \notin FV(P) \cup FV(N)$.

Пример 1 (Пример подстановки).

$$[x \mapsto uv]((\lambda x. (\lambda x. xz)x)x) = (\lambda x. (\lambda x. xz)x)(uv).$$

Лемма 1 (подстановки). Пусть $M, N, L \in \Lambda$, $x \neq y$ и $x \notin FV(L)$. Тогда

$$[y \mapsto L][x \mapsto N]M = [x \mapsto [y \mapsto L]N][y \mapsto L]M.$$

4 α- и β-конверсии. η-конверсия и экстенциональная эквивалентность.

4.1 α-эквивалентность

Позволяет менять переменную на „свежую“. Главная аксиома α-преобразования:

$$\lambda x. M =_{\alpha} \lambda y. [x \mapsto y]M, \text{ если } y \notin FV(M). \quad (\text{правило } \alpha)$$

Аналогично β-эквивалентности можно определить совместимость.

4.2 β-эквивалентность

Основная схема аксиом: для любых $M, N \in \Lambda$:

$$(\lambda x. M)N =_{\beta} [x \mapsto N]M. \quad (\text{правило } \beta)$$

Чтобы сделать $=_{\beta}$ отношением эквивалентности добавим логические аксиомы и правила:

$$\begin{aligned} M &=_{\beta} M \\ M =_{\beta} N &\Rightarrow M =_{\beta} N \\ M =_{\beta} N, N =_{\beta} L &\Rightarrow M =_{\beta} L \end{aligned}$$

Также определим правила совместимости:

$$\begin{aligned} M =_{\beta} N &\Rightarrow ZM =_{\beta} ZN \\ M =_{\beta} N &\Rightarrow MZ =_{\beta} NZ \\ M =_{\beta} N &\Rightarrow \lambda x. M =_{\beta} \lambda x. N \end{aligned} \quad (\text{правило } \xi)$$

Если $M =_{\beta} N$ доказуемо в λ-исчислении, пишут

$$\lambda \vdash M =_{\beta} N.$$

4.3 η -эквивалентность

Схема аксиом η -преобразования:

$$\lambda x. Mx =_{\eta} M, \text{ если } x \notin FV(M). \quad (\text{правило } \eta)$$

Аналогично можем определить правила и совместимость, как для β .

Смысл этой эквивалентности в том, что поведение данных двух термов одинаково; для произвольного N будет верно

$$(\lambda x. M)N =_{\beta} MN.$$

Принцип экстенциональности Две функции считаются экстенционально эквивалентными, если они дают одинаковый результат при одинаковом входе:

$$\forall N: FN =_{\beta} GN.$$

Выберем $u \notin FV(F) \cup FV(G)$, теперь

$$Fu =_{\beta} Gu \implies \lambda y. Fu =_{\beta} \lambda y. Gu \implies F =_{\beta\eta} G$$

5 Кодирование булевых значений, кортежей в чистом бестиповом λ -исчислении.

5.1 Булевы значения

$$\begin{aligned} \text{tru} &\equiv \lambda tf. t \\ \text{fls} &\equiv \lambda tf. f \end{aligned}$$

5.2 Булевы операции

$$\begin{aligned} \text{if} &\equiv \lambda bxy. bxy \\ \text{not} &\equiv \lambda b. b \text{ fls } \text{tru} \\ \text{and} &\equiv \lambda xy. xy \text{ fls} \\ \text{or} &\equiv \lambda xy. x \text{ tru } y \end{aligned}$$

5.3 Кортежи

Пара и стандартные операции:

$$\begin{aligned} \text{pair} &\equiv \lambda xyf. fxy \\ \text{fst} &\equiv \lambda p. p \text{ tru} \\ \text{snd} &\equiv \lambda p. p \text{ fls} \end{aligned}$$

6 Кодирование чисел Чёрча в чистом бестиповом λ -исчислении.

$$\begin{aligned} 0 &\equiv \lambda sz. z \\ 1 &\equiv \lambda sz. sz \\ 2 &\equiv \lambda sz. s(sz) \\ 3 &\equiv \lambda sz. s(s(sz)) \\ 4 &\equiv \lambda sz. s(s(s(sz))) \\ &\dots \end{aligned}$$

Определим $F^n(X)$, где $n \in \mathbb{N}$, $F, Z \in \Lambda$:

$$\begin{aligned} F^0(X) &\equiv X \\ F^{n+1}(X) &\equiv F(F^n(X)) \end{aligned}$$

Теперь число Чёрча принимает следующий вид

$$n \equiv \lambda sz. s^n(z).$$

Функция проверки на ноль:

$$iszero \equiv \lambda n. n(\lambda x. fls) tru.$$

Переход к следующему числу:

$$succ \equiv \lambda nsz. s(ns).$$

Сложение:

$$plus \equiv \lambda mnsz. ms(ns).$$

Умножение:

$$\begin{aligned} mult &\equiv \lambda mn. m(plus\ n)0 \\ mult &\equiv \lambda mnsz. m(ns)x \end{aligned}$$

7 Теорема о неподвижной точке. Y-комбинатор.

7.1 Решение уравнений на термы

Например, хотим найти F такой, что $\forall M, N, L: \lambda \vdash FMNL = ML(NL)$.

$$\begin{aligned} FMNL &= ML(NL) \\ FMNL &= (\lambda l. ML(Nl))L \\ FMN &= \lambda l. ML(Nl) \\ FM &= \lambda n. \lambda l. ML(nl) \\ F &= \lambda mnl. ml(nl) \end{aligned}$$

А что делать, если уравнение рекурсивное? Например, $FM = MF$.

7.2 Теоремы о неподвижной точке

Теорема 1. Для любого λ -терма F существует неподвижная точка:

$$\forall F \in \Lambda \exists X \in \Lambda: \lambda \vdash FX = X.$$

Proof. Пусть $W \equiv \lambda x. F(xx)$ и $X \equiv WW$. Тогда

$$X \equiv WW \equiv (\lambda x. F(xx))W = F(WW) \equiv FX.$$

□

Теорема 2. Существует комбинатор неподвижной точки Y такой, что

$$\forall F: F(YF) = YF.$$

Proof. Пусть $Y = \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))$. Тогда

$$YF \equiv (\lambda x. F(xx))(\lambda x. F(xx)) = F(\underbrace{(\lambda x. F(xx))(\lambda x. F(xx))}_{YF}) \equiv F(YF).$$

□

7.3 Рекурсия

Y-комбинатор позволяет ввести рекурсию в λ -исчисление. Например, можно задать факториал рекурсивно:

$$\text{fac} = \lambda n. \text{if}(\text{iszero } n)1(\text{mult } n(\text{fac}(\text{pred } n))).$$

Можно переписать:

$$\text{fac} = \underbrace{(\lambda fn. \text{if}(\text{iszero } n)1(\text{mult } n(f(\text{pred } n))))}_{\text{fac'}} \text{fac}.$$

Тогда fac — неподвижная точка для fac' , поэтому $\text{fac} = Y \text{fac}'$.

8 Редексы. Одношаговая и многошаговая редукция. Нормальная форма. Редукционные графы.

Если мы хотим доказать равенство, то эффективным способом будет сокращение всех редексов:

$$\text{KI} \equiv (\lambda xy. x) (\lambda z. z) = \lambda yz. z$$

$$\text{IK}_* \equiv (\lambda x. x) \text{IK}_* = \text{IK}_* = (\lambda x. x) (\lambda yz. z) = \lambda yz. z$$

Но как доказать неравенство?

8.1 Редукция

Определение 7: Совместимое отношение

Бинарное отношение \mathcal{R} над Λ называют **совместимым** (с операциями λ -исчисления), если для всех $M, N, Z \in \Lambda$:

$$\begin{aligned} M \mathcal{R} N &\Rightarrow (ZM) \mathcal{R} (ZN), \\ &\quad (MZ) \mathcal{R} (NZ), \\ &\quad (\lambda x. M) \mathcal{R} (\lambda x. N) \end{aligned}$$

Определение 8

Совместимое отношение эквивалентности называют **отношением конгруэнтности** над Λ .

Совместимое, рефлексивное и транзитивное отношение называют **отношением редукции** над Λ .

Определение 9

Бинарное отношение β -редукции за один шаг \rightarrow_β над Λ :

$$\begin{aligned} (\lambda x. M)N &\rightarrow_\beta [x \mapsto N]M \\ M \rightarrow_\beta N &\Rightarrow ZM \rightarrow_\beta ZN \\ M \rightarrow_\beta N &\Rightarrow MZ \rightarrow_\beta NZ \\ M \rightarrow_\beta N &\Rightarrow \lambda x. M \rightarrow_\beta \lambda x. N \end{aligned}$$

Это правило и совместимость.

Определение 10: β -редукция

Бинарное отношение β -редукции \rightarrow_β над Λ определяется индуктивно:

$$\begin{aligned} M &\rightarrow_\beta M && (\text{refl}) \\ M \rightarrow_\beta N &\Rightarrow M \rightarrow_\beta N && (\text{ini}) \\ M \rightarrow_\beta N, N \rightarrow_\beta L &\Rightarrow M \rightarrow_\beta L && (\text{trans}) \end{aligned}$$

Это отношение — транзитивное, рефлексивное замыкание \rightarrow_β , поэтому является отношением редукции.

Определение 11: Конвертируемость

Бинарное отношение **конвертируемости** $=_\beta$ над Λ определяется индуктивно:

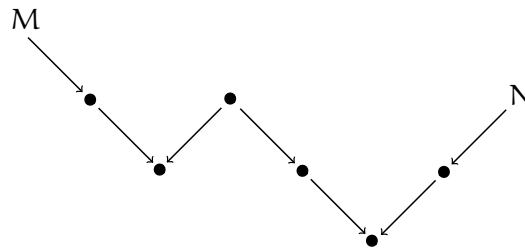
$$\begin{aligned} M \twoheadrightarrow_\beta N &\Rightarrow M =_\beta N && (\text{ini}) \\ M =_\beta N &\Rightarrow N =_\beta M && (\text{sym}) \\ M =_\beta N, N =_\beta L &\Rightarrow M =_\beta L && (\text{trans}) \end{aligned}$$

Это отношение является отношением конгруэнтности.

Утверждение 1. $M =_\beta N \iff \lambda \vdash M = N$.

Proof. Индукция по количеству шагов, расписываем по определению. □

Два термина M, N связаны отношением $=_\beta$, если есть связывающая цепочка стрелок \twoheadrightarrow_β :



Редукционный граф для термина $M \in \Lambda$ (нотация $G_\beta(M)$) — ориентированный мультиграф с вершинами в $\{N \mid M \twoheadrightarrow_\beta N\}$, ребро проводится между N и L , если $N \twoheadrightarrow_\beta L$.

Отношение β -эквивалентности не является разрешимым в общем случае (то есть для пары термов не существует универсального алгоритма, проверяющего эквивалентность).

8.2 Нормальная форма

Это то, что лежит в самом низу на картинке.

Определение 12

λ -терм M *находится* в β -нормальной форме (β -NF), если в нем нет подтермов, являющихся β -редексами.

λ -терм M *имеет* β -нормальную форму, если для некоторого N выполняется $M =_\beta N$ и N находится в β -NF.

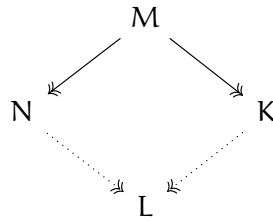
Пример 2.

- $\lambda x y. x(\lambda z. zx)y$ находится в β -NF
- $(\lambda x. xx)y$ не находится в β -NF, но имеет в качестве β -NF-терм yy
- Ω не имеет нормальной формы

9 Теорема Чёрча-Россера и ее следствия.

Теорема 3 (Черч, Россер). Если $M \twoheadrightarrow_\beta N$, $M \twoheadrightarrow_\beta K$, то существует такой L , что $N \twoheadrightarrow_\beta L$ и $K \twoheadrightarrow_\beta L$.

Иначе говоря, β -редукция обладает *свойством ромба* или *конфлюентностью*:



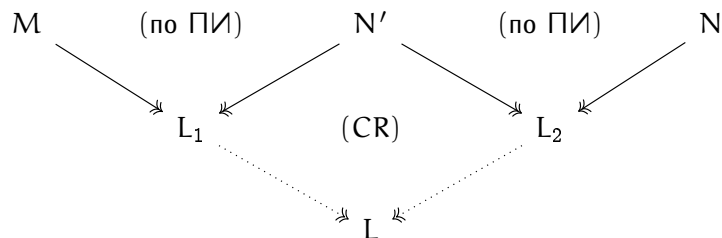
Следствие 1 (о существовании общего редукта). Если $M =_{\beta} N$, то существует L такой, что $M \rightarrow_{\beta} L$ и $N \rightarrow_{\beta} L$.

Proof. Индукция по генерации $=_{\beta}$. Разберем три случая вывода $M =_{\beta} N$:

$M \rightarrow_{\beta} N$. Возьмем $L = N$.

$N =_{\beta} M$. По предположению индукции есть общий β -редукт L_1 для N и M . Можем взять $L = L_1$.

$M =_{\beta} N'$, $N' =_{\beta} N$. Тогда



□

Следствие 2 (о редуцируемости к β -NF). Если M имеет N в качестве β -NF, то $M \rightarrow_{\beta} N$.

Следствие 3 (о единственности β -NF). λ -терм имеет не более одной β -NF.

10 Стратегии редукции. Теорема о нормализации. Механизмы вызова в функциональных языках.

Рассмотрим три варианта терма:

- Переменная v , редукция завершена, ничего интересного
- Абстракция $\lambda x. M$, просто редуцируем M
- Аппликация MN . Разбираем аппликацию влево (в M) до не-аппликации. Пока получаем аппликацию, разделяем дальше. Два варианта остановки:
 - Переменная. Тогда мы получили такой вид

$$(\dots ((vN_1)N_2) \dots N_k).$$

Теперь нам нужно просто проредуцировать каждый N_i , общая структура от этого не изменится.

- Абстракция. Получили:

$$(\dots (((\lambda x. M')N_1)N_2) \dots N_k).$$

Будем работать дальше. Здесь есть две стратегии:

- * **Нормальная стратегия:** сразу сокращаем редекс $(\lambda x. M)N_1$.

- * **Аппликативная стратегия:** сначала редуцируем отдельно все N_i слева направо до нормальной формы N'_i , а уже потом сокращаем редекс $(\lambda x. M)N'_1$.
Также можно редуцировать только N_1 и сразу сократить.

Разбор терма можно представить в виде дерева, где узлы $@$ задают аппликацию, а узлы λ — абстракцию.

Теорема 4. Лямбда-терм может иметь одну из двух форм:

$$\begin{aligned} \lambda \vec{x}. y \vec{N} &\equiv \lambda x_1 x_2 \dots x_n. y N_1, N_2, \dots N_k, \quad n, k \geq 0 & (\text{HNF}) \\ \lambda \vec{x}. (\lambda z. M) \vec{N} &\equiv \lambda x_1 x_2 \dots x_n. (\lambda z. M) N_1, N_2, \dots N_k, \quad n \geq 0, k > 0 \end{aligned}$$

Определение 13

Головная нормальная форма — форма следующего вида:

$$\lambda \vec{x}. y \vec{N} \equiv \lambda x_1 x_2 \dots x_n. y N_1, N_2, \dots N_k, \quad n, k \geq 0. \quad (\text{HNF})$$

Слабая головная нормальная форма (WHNF) — это HNF или лямбда-абстракция (то есть не редекс на верхнем уровне).

Переменная y называется **головной переменной**. Редекс $(\lambda z. M)N_1$ называется **головным редексом**.

Переменная y может совпадать с одним из x_i .

В Haskell вычисления как раз форсируются до слабой нормальной формы.

Теорема 5 (о нормализации). Если терм M имеет нормальную форму, то последовательное сокращение самого левого внешнего редекса приводит к этой нормальной форме.

То есть, если HNF есть, то нормальная стратегия гарантировано приводит к ней.

10.1 Механизмы вызова в ФЯ

Аппликативная стратегия более эффективная, хоть и не всегда приводит к HNF, поэтому применяется во многих языках программирования.

Пусть N — очень большой терм, который вычисляется сутки. Если мы запустим нормальную стратегию на

$$(\lambda x. Fx(Gx)x)N,$$

получим $FN(GN)N$ и N придется в дальнейших редукциях сократить трижды. Но для такого нормальная стратегия не вычислит N ни разу:

$$(\lambda x y. y)N \longrightarrow_{\beta} \lambda y. y.$$

Аппликативная стратегия будет вычислять N только один раз в каждом из примеров.

В „ленивых” языках программирования (Haskell, Clean) используется похожая на нормальную стратегия, а для решения проблем с эффективностью используют механизм *разделения*: вместо непосредственной подстановки терма подставляют указатель на терм, который хранится в памяти как *отложенное вычисление*.

Механизм вызова в функциональных языках:

- *вызов по значению* — аппликативный порядок редукций до WHNF
- *вызов по имени* — нормальный порядок редукций до WHNT
- *вызов по необходимости* — „вызов по имени” с разделением.

11 Функция предшествования для чисел Черча. Комбинатор примитивной рекурсии.

11.1 Функция предшествования

Введем вспомогательные функции:

$$\begin{aligned}zp &\equiv \text{pair } 00 \\sp &\equiv \lambda p. \text{pair}(\text{snd } p)(\text{succ}(\text{snd } p))\end{aligned}$$

Вторая функция позволяет, приняв пару $(_, j)$, вернуть пару $(j, j + 1)$:

$$sp(\text{pair } i \ j) = \text{pair } j \ (j + 1).$$

Если взять композицию $m > 0$ раз от zp :

$$\begin{aligned}sp^0(zp) &= \text{pair } 00 \\sp^m(zp) &= \text{pair}(m - 1) \ m\end{aligned}$$

Тогда можно определить функцию предшествования:

$$\text{pred} = \lambda m. \text{fst}(m \ sp \ zp).$$

Тогда

$$\text{minus} = \lambda m \ n. n \ \text{pred } m.$$

11.2 Комбинатор примитивной рекурсии

Обобщим конструкцию. Второй элемент пары оставим счетчиком, а в первом храним результат.

$$\begin{aligned}xz &\equiv \lambda x. \text{pair } x \ 0 \\fs &\equiv \lambda f \ p. \text{pair}(f(\text{fst } p)(\text{snd } p))(\text{succ}(\text{snd } p)) \\rec &\equiv \lambda m \ f \ x. \text{fst}(m(fs \ f)(xz \ x))\end{aligned}$$

Теперь можно выразить pred :

$$\text{pred}' = \lambda n. \text{rec } n \ (K_*) \ 0.$$

И факториал:

$$\text{fac} = \lambda n. \text{rec } n \ (\lambda x \ y. \text{mult } x(\text{succ } y)) \ 1.$$

11.3 Списки

$$\begin{aligned}\text{nil} &\equiv \lambda c \ n. n \\cons &\equiv \lambda e \ l \ c \ n. c \ e(l \ c \ n)\end{aligned}$$

Теперь

$$\begin{aligned}[] &= \text{nil} \\[5, 3, 2] &= \text{cons } 5(\text{cons } 3(\text{cons } 2 \ \text{nil})) = \lambda c \ n. c \ 5(c \ 3(c \ 2 \ n)) \\cons &\equiv \lambda l. l(\lambda h \ t. \text{fls}) \ \text{tru}.\end{aligned}$$

12 Просто типизированное λ -исчисление в стиле Карри. Предтермы. Утверждения о типизации. Контексты. Правила типизации.

12.1 Понятие типа

Система типов — гибко управляемый синтаксический метод доказательства отсутствия в программе определенных видов поведения при помощи классификации выражений языка по разновидностям вычисляемых ими значений (Бенджамин Пирс).

В λ -исчислении:

- выражения — λ -термы
- вычисления — их редукция
- значения — $(\text{WH})\text{NF}$

Типы рассматриваются как *синтаксические* конструкции, приписываемые термам по определенным правилам:

$$M: \sigma.$$

12.2 Для чего нужны типы?

- Частичная спецификация:

$$f: \mathbb{N} \rightarrow \mathbb{N}, \quad g: (\forall n: \mathbb{N}. \exists m: \mathbb{N}. m \leq n).$$

- Проверка типов помогает отлавливать простые ошибки

12.3 Тип функции

Стрелка — базовый способ конструирования типа, задающая функциональный тип. Например, для функции $I \equiv \lambda x. x$ может быть приписан тип:

$$I: \alpha \rightarrow \alpha.$$

В общем случае типом функции из α в β будет $\alpha \rightarrow \beta$.

Если $y: \alpha$ является аргументом функции $f: \alpha \rightarrow \beta$, то возвращаемое значение $f y$ имеет тип β .

Гипотеза о типе переменных записывают в контексте:

$$y: \alpha \vdash (Iy): \alpha.$$

12.4 Системы в стиле Карри

Оставляем термы такими же, что и в бестиповой теории. Каждый терм может обладать множеством различных типов (пустое, одно-, многоэлементное, бесконечное).

12.5 Просто типизированное λ -исчисление

Определение 14

Множество типов \mathbb{T} системы λ_{\rightarrow} определяется индуктивно:

$$\begin{array}{ll} \alpha, \beta \dots \in \mathbb{T} & \text{(переменные типа)} \\ \sigma, \tau \in \mathbb{T} \Rightarrow (\sigma \rightarrow \tau) \in \mathbb{T} & \text{(типы пространства функций)} \end{array}$$

В абстрактном синтаксисе: $\mathbb{T} ::= \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$, где $\mathbb{V} = \{\alpha, \beta \dots\}$.

Стрелка *правоассоциативна*: если $\sigma_1, \dots, \sigma_n \in \mathbb{T}$, то

$$\sigma_1 \rightarrow \sigma_2 \rightarrow \dots \rightarrow \sigma_n \equiv (\sigma_1 \rightarrow (\sigma_2 \rightarrow \dots \rightarrow (\sigma_{n-1} \rightarrow \sigma_n) \dots)).$$

Всякий тип может быть записан в виде:

$$\sigma_1 \rightarrow \dots \rightarrow \sigma_n \rightarrow \alpha.$$

Определение 15: Предтермы

Множество **предтермов** Λ строится из переменных из $V = \{x, y, z, \dots\}$ с помощью аппликации и абстракции:

$$\begin{aligned} x \in V &\implies x \in \Lambda \\ M, N \in \Lambda &\implies (MN) \in \Lambda \\ M \in \Lambda, x \in V &\implies (\lambda x. M) \in \Lambda \end{aligned}$$

В абстрактном стиле: $\Lambda ::= V \mid (\Lambda\Lambda) \mid (\lambda V. \Lambda)$.

Предтермы системы в стиле Карри — термы бестипового λ -исчисления.

Определение 16

Утверждение о типизации $\lambda \rightarrow$ по Карри имеет вид $M : \tau$, где субъект $M \in \Lambda$ и предикат $\tau \in \mathbb{T}$.

Пример 3. Примеры утверждений типизации

$$\begin{aligned} \lambda x. x : \alpha \rightarrow \alpha \\ \lambda x. x : (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta \\ \lambda x y. x : \alpha \rightarrow \beta \rightarrow \alpha \end{aligned}$$

Определение 17: Объявление

Объявление — утверждение типизации с термовой переменной в кресле субъекта.

Пример 4. Примеры объявлений

$$\begin{aligned} x : \alpha \\ y : \beta \\ f : (\alpha \rightarrow \beta) \rightarrow \gamma \end{aligned}$$

Определение 18: Контекст

Контекст — множество объявлений с *различными* переменными в качестве субъекта:

$$\Gamma = \{x_1 : \sigma_1, x_2 : \sigma_2, \dots, x_n : \sigma_n\}.$$

Контекст иногда называют *базисом* или *окружением*.

Фигурные скобки иногда опускают:

$$\Gamma = x : \alpha, f : \alpha \rightarrow \beta, g : (\alpha \rightarrow \beta) \rightarrow \gamma \equiv x^\alpha, f^{\alpha \rightarrow \beta}, g^{(\alpha \rightarrow \beta) \rightarrow \gamma}.$$

Контексты можно расширять, добавляя объявление новой переменной.

Также контексты можно рассматривать как частичные функции из V в множество типов \mathbb{T} .

12.6 Правила типизации

Определение 19

Утверждение $M: \tau$ называется **выводимым** в контексте Γ , обозначается

$$\Gamma \vdash M: \tau,$$

если его вывод может быть произведен по правилам:

$$\begin{aligned} x^\sigma \in \Gamma &\implies \Gamma \vdash x: \sigma \\ \Gamma \vdash M: \sigma \rightarrow \tau, \Gamma \vdash N: \sigma &\implies \Gamma \vdash MN: \tau \\ \Gamma, x^\sigma \vdash M: \tau &\implies \Gamma \vdash \lambda x. M: \sigma \rightarrow \tau \end{aligned}$$

Если существуют Γ и τ такие, что $\Gamma \vdash M: \tau$, то предтерм M называют **(допустимым) термом**.

Запись в виде дерева вывода:

$$\begin{array}{ll} \Gamma \vdash x: \sigma, \text{ если } x^\alpha \in \Gamma & \text{(аксиома)} \\ \frac{\Gamma \vdash M: \sigma \rightarrow \tau \quad \Gamma \vdash N: \sigma}{\Gamma \vdash MN: \tau} & (\rightarrow \text{Elim}) \\ \frac{\Gamma, x^\alpha \vdash M: \tau}{\Gamma \vdash \lambda x. M: \sigma \rightarrow \tau} & (\rightarrow \text{Intro}) \end{array}$$

13 Просто типизированное λ -исчисление в стиле Черча. Предтермы. Утверждения о типизации. Контексты. Правила типизации.

13.1 Системы в стиле Черча

Термы — аннотированные версии бестиповых термов. Каждый терм имеет обычно уникальный тип, выводимый из способа, которым терм аннотирован.

Определение 20: Предтермы

Множество **предтермов** $\Lambda_{\mathbb{T}}$ строится из переменных из $V = \{x, y, z, \dots\}$ с помощью аппликации и аннотированной типами абстракции:

$$\begin{aligned} x \in V &\implies x \in \Lambda_{\mathbb{T}} \\ M, N \in \Lambda_{\mathbb{T}} &\implies (MN) \in \Lambda_{\mathbb{T}} \\ M \in \Lambda_{\mathbb{T}}, x \in V, \sigma \in \mathbb{T} &\implies (\lambda x^\sigma. M) \in \Lambda_{\mathbb{T}} \end{aligned}$$

В абстрактном стиле: $\Lambda_{\mathbb{T}} ::= V \mid (\Lambda_{\mathbb{T}} \Lambda_{\mathbb{T}}) \mid (\lambda V^{\mathbb{T}}. \Lambda_{\mathbb{T}})$.

Определение 21

Утверждение о типизации λ_{\rightarrow} по Черчу имеет вид $M: \tau$, где субъект $M \in \Lambda^{\mathbb{T}}$ и предикат $\tau \in \mathbb{T}$.

Пример 5. Примеры утверждений типизации

$$\begin{aligned} \lambda x^\alpha. x: \alpha \rightarrow \alpha \\ \lambda x^{\alpha \rightarrow \beta}. x: (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta \\ \lambda x^\alpha y^\beta. x: \alpha \rightarrow \beta \rightarrow \alpha \end{aligned}$$

13.2 Правила типизации

Определение 22

Утверждение $M: \tau$ называется **выводимым** в контексте Γ , обозначается

$$\Gamma \vdash M: \tau,$$

если его вывод может быть произведен по правилам:

$$\begin{aligned} x^\sigma \in \Gamma &\implies \Gamma \vdash x: \sigma \\ \Gamma \vdash M: \sigma \rightarrow \tau, \Gamma \vdash N: \sigma &\implies \Gamma \vdash MN: \tau \\ \Gamma, x^\sigma \vdash M: \tau &\implies \Gamma \vdash \lambda x. M: \sigma \rightarrow \tau \end{aligned}$$

Если существуют Γ и τ такие, что $\Gamma \vdash M: \tau$, то предтерм M называют **(допустимым) термом**.

Запись в виде дерева вывода:

$$\begin{array}{l} \Gamma \vdash x: \sigma, \text{ если } x^\sigma \in \Gamma \quad \text{(аксиома)} \\ \hline \Gamma \vdash M: \sigma \rightarrow \tau \quad \Gamma \vdash N: \sigma \quad \text{---} \quad \Gamma \vdash MN: \tau \quad \text{---} \quad (\rightarrow \text{Elim}) \\ \hline \Gamma, x^\sigma \vdash M: \tau \quad \text{---} \quad \Gamma \vdash \lambda x^\sigma. M: \sigma \rightarrow \tau \quad \text{---} \quad (\rightarrow \text{Intro}) \end{array}$$

14 Свойства систем просто типизированного λ -исчисления

Лемма 2 (об инверсии или о генерации).

- $\Gamma \vdash x: \sigma \implies x^\sigma \in \Gamma$
- $\Gamma \vdash \lambda x. M: \tau \implies \exists \sigma: [\Gamma \vdash M: \sigma \rightarrow \tau \wedge \Gamma \vdash N: \sigma]$
- Карри: $\Gamma \vdash \lambda x. M: \rho \implies \exists \sigma, \tau: [\Gamma, x^\sigma \vdash M: \tau \wedge \rho \equiv \sigma \rightarrow \tau]$
- Черч: $[\Gamma \vdash \lambda x^\sigma. M: \rho \implies \exists \tau: [\Gamma, x^\sigma \vdash M: \tau \wedge \rho \equiv \sigma \rightarrow \tau]]$

Лемма 3 (о типизируемости подтерма). Пусть M' — подтерм M . Тогда для некоторых Γ' и σ'

$$\Gamma \vdash M: \sigma \implies \Gamma' \vdash M': \sigma'.$$

То есть, если терм имеет тип, что и подтерм имеет тип.

14.1 Леммы о контекстах

Лемма 4 („разбавления“). Пусть Γ и Δ — контексты, причем $\Gamma \subset \Delta$. Тогда

$$\Gamma \vdash M: \sigma \implies \Delta \vdash M: \sigma.$$

Расширение контекста не влияет на выводимость утверждения типизации.

Лемма 5 (о свободных переменных). Свободные переменные типизированного терма должны присутствовать в контексте:

$$\Gamma \vdash M: \sigma \implies \text{FV}(M) \subset \text{dom}(\Gamma).$$

Лемма 6 (сужения). Сужение контекста до множества свободных переменных терма не влияет на выводимость утверждения типизации:

$$\Gamma \vdash M: \sigma \implies \Gamma|_{\text{FV}(M)} \vdash M: \sigma.$$

Вместе эти леммы отвечают на вопрос: какой контекст требуется, чтобы произвести присваивание типов? Ответ следующий: в контексте обязательно должны присутствовать свободные переменные типизируемого терма; все остальные переменные опциональны, не влияют на типизацию и могут быть безболезненно отброшены по лемме сужения.

Рассмотрим предтерм xx . Пусть это терм. Тогда есть Γ и τ :

$$\Gamma \vdash xx : \tau.$$

По лемме об инверсии существует σ , что правый подтерм $x : \sigma$, левый подтерм (тоже x) имеет тип $\sigma \rightarrow \tau$.

По лемме о контекстах $x \in \text{dom}(\Gamma)$ и должен иметь там единственное связывание по определению контекста. То есть $\sigma = \sigma \rightarrow \tau$ — получили, что тип является подвыражением себя, чего не может быть, так как типы конечны.

Тогда ω , Ω и Y не имеют типа по лемме о типизируемости подтерма.

Определение 23

Для $\sigma, \tau \in \mathbb{T}$ **подстановку** τ вместо α в σ обозначим $[\alpha \mapsto \tau]\sigma$.

Пример 6.

$$[\alpha \mapsto (\gamma \rightarrow \gamma)]\alpha \rightarrow \beta \rightarrow \alpha = (\gamma \rightarrow \gamma) \rightarrow \beta \rightarrow \gamma \rightarrow \gamma.$$

Лемма 7 (подстановки типа). Подстановка в выводимое утверждение типизации некоторого типа вместо переменной типа порождает выводимое утверждение типизации.

Карри:

$$\Gamma \vdash M : \sigma \implies [\alpha \mapsto \tau]\Gamma \vdash M : [\alpha \mapsto \tau]\sigma.$$

Черч:

$$\Gamma \vdash M : \sigma \implies [\alpha \mapsto \tau]\Gamma \vdash [\alpha \mapsto \tau]M : [\alpha \mapsto \tau]\sigma.$$

Пример 7. Подстановка $[\alpha \mapsto (\gamma \rightarrow \gamma)]$ в утверждение типизации для системы Черча

$$x^\alpha \vdash \lambda y^\alpha z^\beta. x : \alpha \rightarrow \beta \rightarrow \alpha$$

осуществляется и в тип, и в терм, и в контекст и дает новое утверждение типизации

$$x^{\gamma \rightarrow \gamma} \vdash (\lambda y^{\gamma \rightarrow \gamma} z^\beta. x) : (\gamma \rightarrow \gamma) \rightarrow \beta \rightarrow \gamma \rightarrow \gamma.$$

Поскольку первое утверждение типизации выводимо, то и второе тоже выводимо.

Лемма 8 (подстановке терма). Пусть $\Gamma, x^\sigma \vdash M : \tau$ и $\Gamma \vdash N : \sigma$, тогда

$$\Gamma \vdash [x \mapsto N]M : \tau.$$

Подходящая по типу подстановка терма сохраняет тип.

Пример 8. Берем выводимое утверждение типизации

$$x^{\gamma \rightarrow \gamma} \vdash \lambda y^\beta. x : \beta \rightarrow \gamma \rightarrow \gamma.$$

и подставляем в него вместо свободной переменной x типа $\gamma \rightarrow \gamma$ терм $\lambda z^\gamma. z$ подходящего типа $\gamma \rightarrow \gamma$. Получаем

$$\vdash \lambda y^\beta z^\gamma. z : \beta \rightarrow \gamma \rightarrow \gamma.$$

Теорема 6 (о редукции субъекта). Пусть $M \rightarrow_\beta N$, тогда

$$\Gamma \vdash M : \sigma \implies \Gamma \vdash N : \sigma.$$

Тип терма сохраняется про β -редукциях.

Proof. Следует из леммы о подстановке терма. □

Следствие 4. Множество типизируемых в λ_{\rightarrow} термов замкнуто относительно редукции.

В обратную сторону теорема и следствие не верны для λ_{\rightarrow} .

Теорема 7 (о единственности типа для λ_{\rightarrow} по Черчу). Пусть $\Gamma \vdash M: \sigma$ и $\Gamma \vdash M: \tau$. Тогда $\sigma \equiv \tau$.
Терм в λ_{\rightarrow} по Черчу имеет единственный тип.

Следствие 5. Пусть $\Gamma \vdash M: \sigma$, $\Gamma \vdash N: \tau$ и $M =_{\beta} N$. Тогда $\sigma \equiv \tau$.
Типизируемые β -конвертируемые термы имеют одинаковый тип в λ_{\rightarrow} по Черчу.

Пример 9 (Контрпример для системы по Карри). Следующие два типа подходят для $K = \lambda x y. x$ по Карри:

$$\begin{aligned} \vdash \lambda x y. x: \alpha \rightarrow (\delta \rightarrow \gamma \rightarrow \delta) \rightarrow \alpha \\ \vdash \lambda x y. x: (\gamma \rightarrow \gamma) \rightarrow \beta \rightarrow \gamma \rightarrow \gamma \end{aligned}$$

15 Связь между системами Карри и Черча. Проблемы разрешимости. Сильная и слабая нормализация.

Зададим для Черча *стирающее отображение* $|\cdot|: \Lambda_{\mathbb{T}} \rightarrow \Lambda$:

$$\begin{aligned} |x| &\equiv x \\ |MN| &\equiv |M||N| \\ |\lambda x^{\sigma}. M| &\equiv \lambda x. |M| \end{aligned}$$

Все атрибутированные типами термы из версии по Черчу „проектируются” в термы в версии по Карри:

$$M \in \Lambda_{\mathbb{T}} \wedge \Gamma \vdash M: \sigma \implies \Gamma \vdash |M|: \sigma.$$

Также можно „поднять” в обратную сторону из Карри в Черча, правильно подобрав типы:

$$M \in \Lambda \wedge \Gamma \vdash M: \sigma \implies \exists N \in \Lambda_{\mathbb{T}}: [\Gamma \vdash N: \sigma \wedge |N| \equiv |M|].$$

Для произвольного типа $\sigma \in \mathbb{T}$ обитаемость в версии по Карри равносильна обитаемости в версии по Черчу.

15.1 Проблемы разрешимости

Для любой системы типов важную роль имеют проблемы разрешимости основных задач: есть ли алгоритм, решающий данную задачу? Для λ_{\rightarrow} все эти задачи разрешимы. Первые две задачи будут

$\vdash M: \sigma?$	Задача проверки типа	ЗПТ, ТСП
$\vdash M: ?$	Задача синтеза типа	ЗСТ, TSP / TAP
$\vdash?: \sigma$	Задача обитаемости типа	ЗОТ, TIP

решены с помощью теоремы Хиндли–Миллера.

Определение 24

Терм называется **слабо (weak) нормализуемым (WN)**, если существует последовательность редукций, приводящих его к нормальной форме.

Терм называется **сильно (strong) нормализуемым (SN)**, если любая последовательность редукций, приводящих его к нормальной форме.

Пример 10. KIK сильно нормализуем, $KI\Omega$ слабо нормализуем, Ω не нормализуем вообще.

Определение 25

Систему типов называют **слабо нормализуемой**, если все ее допустимые термы слабо нормализуемы. Систему типов называют **сильно нормализуемой**, если все ее допустимые термы сильно нормализуемы.

Теорема 8 (о нормализации λ_{\rightarrow}). Обе системы (по Карри и по Черчу) сильно нормализуемы.

То есть любой допустимый терм в λ_{\rightarrow} всегда редуцируется к нормальной форме независимо от выбранной стратегии редукции.

16 Понятие главного (наиболее общего) типа. Подстановка типа и их композиция.

Для систем Карри и Черча верна *лемма подстановки типа*:

$$\Gamma \vdash M : \sigma \implies [\alpha \mapsto \tau] \Gamma \vdash [\alpha \mapsto \tau] M : [\alpha \mapsto \tau] \sigma.$$

В версии Черча λ_{\rightarrow} термы атрибутированы типами, поэтому тип терма единственен:

$$\begin{aligned} \lambda f^{\sigma \rightarrow \tau \rightarrow \rho} g^{\sigma \rightarrow \tau} z^{\sigma}. f z(g z) : (\sigma \rightarrow \tau \rightarrow \rho) \rightarrow (\sigma \rightarrow \tau) \rightarrow \sigma \rightarrow \rho \\ \lambda f^{\sigma \rightarrow \tau \rightarrow \sigma} g^{\sigma \rightarrow \tau} z^{\sigma}. f z(g z) : (\sigma \rightarrow \tau \rightarrow \sigma) \rightarrow (\sigma \rightarrow \tau) \rightarrow \sigma \rightarrow \sigma \\ \lambda f^{(\tau \rightarrow \rho) \rightarrow \tau \rightarrow \rho} g^{(\tau \rightarrow \rho) \rightarrow \tau} z^{(\tau \rightarrow \rho)}. f z(g z) : ((\tau \rightarrow \rho) \rightarrow \tau \rightarrow \rho) \rightarrow ((\tau \rightarrow \rho) \rightarrow \tau) \rightarrow (\tau \rightarrow \rho) \rightarrow \rho \end{aligned}$$

Любой из этих типов можно приписать терму $S = \lambda f g z. f z(g z)$ в версии Карри.

Однако, первый „лучше” в том смысле, что остальные получаются из него подстановками типа вместо типовых переменных. Он называется **главным (principle)**. Также он отличается тем, что в нем меньше всего стрелок и максимальное разнообразие типов.

16.1 Вывод главного типа

Пусть у нас есть комбинатор

$$S = \lambda x y. y(\lambda z. y x).$$

Припишем всем термовым переменным типовую метапеременную:

$$\lambda x^{\alpha} y^{\beta}. y^{\beta}(\lambda z^{\gamma}. y^{\beta} x^{\alpha}).$$

Припишем типовую переменную метапеременную всем аппликативным подтермам: $(y x)'$: δ , $y(\lambda z. y z)$: ε

$$\lambda x^{\alpha} y^{\beta}. \underbrace{y^{\beta}(\lambda z^{\gamma}. \overbrace{y^{\beta} x^{\alpha}}^{\delta})}_{\varepsilon}.$$

Теперь можем связать некоторые метапеременные уравнениями, необходимыми для типизируемости терма:

$$\beta \sim \alpha \rightarrow \delta, \quad \beta \sim (\gamma \rightarrow \delta) \rightarrow \varepsilon.$$

Найдем главный унификатор для типовых переменных (подстановку), дающий решения уравнений:

$$\alpha := \gamma \rightarrow \delta, \quad \beta := (\gamma \rightarrow \delta) \rightarrow \varepsilon, \quad \delta := \varepsilon.$$

Главный тип

$$\lambda x y. y(\lambda z. y z) : (\gamma \rightarrow \varepsilon) \rightarrow ((\gamma \rightarrow \varepsilon) \rightarrow \varepsilon) \rightarrow \varepsilon.$$

Определение 26: Подстановка типа

Подстановка типа — операция $S: \mathbb{T} \rightarrow \mathbb{T}$ такая, что

$$S(\sigma \rightarrow \tau) \equiv S(\sigma) \rightarrow S(\tau).$$

Можно представлять типы как вершины бинарного дерева, пара ребер соответствует подстановке. Обычно подстановка тождественна на всех типовых переменных, кроме конечного носителя

$$\text{sup}(S) = \{\alpha \mid S(\alpha) \neq \alpha\}.$$

Пример подстановки:

$$S = [\alpha := \gamma \rightarrow \beta, \beta := \alpha \rightarrow \gamma].$$

Тождественная подстановка (то есть с пустым носителем) обозначают \square .

Все подстановки происходят *параллельно*: для $\tau = \alpha \rightarrow \beta \rightarrow \gamma$:

$$\begin{aligned} S(\tau) &= [\alpha := \gamma \rightarrow \beta, \beta := \alpha \rightarrow \gamma](\alpha \rightarrow \beta \rightarrow \gamma) = \\ &= (\gamma \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma) \rightarrow \gamma \end{aligned}$$

Определение 27: Композиция подстановок

Композиция двух подстановок — подстановка с носителем, являющимся объединением носителей, над которым последовательно выполнены обе подстановки.

Подстановки образуют моноид относительно композиции и с \square в роли нейтрального элемента.

Определение 28: Унификатор

Унификатор для типов σ и τ — подстановка S такая, что $S(\sigma) \equiv S(\tau)$.

Пример 11. Пусть $\sigma = \alpha \rightarrow \beta \rightarrow \gamma$, $\tau = (\delta \rightarrow \epsilon) \rightarrow \phi$. Унификатор:

$$S = [\alpha := \delta \rightarrow \epsilon, \phi := \beta \rightarrow \gamma].$$

В результате такой подстановки

$$S(\sigma) \equiv S(\tau) \equiv (\delta \rightarrow \epsilon) \rightarrow (\beta \rightarrow \gamma).$$

Когда выделяем одну элементарную подстановку нужно сразу выполнить ее повсюду.

Определение 29: Главный унификатор

Унификатор S — **главный унификатор** для σ и τ , если для любого другого унификатора S' существует подстановка T такая, что

$$S' \equiv T \circ S.$$

17 Алгоритм унификации

Теорема 9 (Робинсон, 1965). Существует алгоритм унификации \mathcal{U} , который для заданных типов σ и τ возвращает:

- главный унификатор S для σ и τ , если σ и τ могут быть унифицированы;
- сообщение об ошибке в противном случае.

Алгоритм $\mathcal{U}(\sigma, \tau)$ позволяет искать минимальное решение уравнения на типы $\sigma \sim \tau$.

Ключевой момент всех рассуждений про унификацию:

$$\sigma_1 \rightarrow \sigma_2 \equiv \tau_1 \rightarrow \tau_2 \iff \sigma_1 \equiv \tau_1 \wedge \sigma_2 \equiv \tau_2.$$

17.1 Алгоритм

Опишем алгоритм:

$$\begin{aligned}
 \mathcal{U}(\alpha, \alpha) &= [] \\
 \mathcal{U}(\alpha, \tau) \mid \alpha \in \text{FV}(\tau) &= \text{error} \\
 \mathcal{U}(\alpha, \tau) \mid \alpha \notin \text{FV}(\tau) &= [\alpha := \tau] \\
 \mathcal{U}(\sigma_1 \rightarrow \sigma_2, \alpha) &= \mathcal{U}(\alpha, \sigma_1, \sigma_2) \\
 \mathcal{U}(\sigma_1 \rightarrow \sigma_2, \tau_1 \rightarrow \tau_2) &= \mathcal{U}(\mathcal{U}_2 \sigma_1, \mathcal{U}_2 \tau_1) \circ \mathcal{U}_2 \quad \text{where } \mathcal{U}_2 = \mathcal{U}(\sigma_2, \sigma_1)
 \end{aligned}$$

- $\mathcal{U}(\sigma, \tau)$ завершается, так как деревья типа конечны и количество типовых переменных сокращается на 1 через конечное число шагов: интересен только последний шаг (остальные сразу точно завершаются), но мы дойдем до переменных, так как увеличение происходит из-за обход по поддереву, а его высота каждый раз уменьшается, с некоторого момента, общая высота станет уменьшаться; когда мы дошли до переменной, их количество уменьшится.
- $\mathcal{U}(\sigma, \tau)$ унифицирует: по индукции, если S унифицирует (σ, τ) , то $S \circ [\alpha := \rho]$ унифицирует $(\sigma \rightarrow \alpha, \tau \rightarrow \rho)$.
- $\mathcal{U}(\sigma, \tau)$ выдает главный унификатор.

18 Алгоритм построения системы ограничений.

Наша первая цель — построить систему ограничений на типы для терма M (возможно незамкнутого).

Для типизации таких термов необходим контекст Γ , в котором объявляются типы всех свободных переменных.

Для подстановки S , унифицирующей систему уравнений на типы

$$E = \{\sigma_1 \sim \tau_1, \dots, \sigma_n \sim \tau_n\},$$

введем обозначение $S \models E$.

Теорема 10 (Теорема о существовании системы ограничений). Для любых терма $M \in \mathcal{L}$, контекста Γ ($\text{FV}(M) \subseteq \text{dom}(\Gamma)$) и типа $\sigma \in \mathbb{T}$ существует конечное множество уравнений на типы $E = E(\Gamma, M, \sigma)$, такое что для некоторой подстановки S :

- $S \models E(\Gamma, M, \sigma) \implies S(\Gamma) \vdash M : S(\sigma)$
- $S(\Gamma) \vdash M : S(\sigma) \implies S' \models E(\Gamma, M, \sigma)$, для некоторого S' , имеющего тот же эффект, что и S , на типовых переменных в Γ и σ .

18.1 Алгоритм

$$\begin{aligned}
 E(\Gamma, x, \sigma) &= \{\sigma \Gamma(x)\} \\
 E(\Gamma, MN, \sigma) &= E(\Gamma, M, \alpha \rightarrow \sigma) \\
 E(\Gamma, \lambda x. M, \sigma) &= E(\Gamma \cup \{x : \alpha\}, M, \beta) \cup \{\alpha \rightarrow \beta \sigma\}
 \end{aligned}$$

В первом равенстве контекст Γ рассматривается как функция из множества переменных в множество типов.

Переменные α во втором и третьем равенствах и β в третьем всякий раз должны быть „свежими“!

19 Главная пара и главный тип. Теорема Хиндли–Миллера.

Определение 30: Главная пара

Главная пара для $M \in \Lambda$ — пара (Γ, σ) такая, что

- $\Gamma \vdash M: \sigma$
- $\Gamma' \vdash M: \sigma' \implies \exists S: S(\Gamma) \subseteq \Gamma' \wedge S(\sigma) \equiv \sigma'$

Пример 12. Для $M = \lambda x. x y$ имеем

$$\begin{aligned} PP(M) &= (y: \alpha, (\alpha \rightarrow \beta) \rightarrow \beta \\ y: \alpha &\vdash (\lambda x. x y): (\alpha \rightarrow \beta) \rightarrow \beta \end{aligned}$$

Теорема 11 (Теорема Хиндли – Милнера). Существует алгоритм PP, возвращающий для $M \in \Lambda$ главную пару (Γ, σ) , если M имеет тип и сообщение об ошибке в противном случае.

Пусть

$$\begin{aligned} FV(M) &= \{x_1, \dots, x_n\} \\ \Gamma_0 &= x_1: \alpha_1, \dots, x_n: \alpha_n, \quad \sigma_0 = \beta. \end{aligned}$$

19.1 Алгоритм PP

$$\begin{aligned} PP(M) \mid \mathcal{U}(E(\Gamma_0, M, \sigma_0)) &\equiv \text{error} = \text{error} \\ PP(M) \mid \mathcal{U}(E(\Gamma_0, M, \sigma_0)) &\equiv S = (S(\Gamma_0), S(\sigma_0)) \end{aligned}$$

Стартуем с произвольных переменных типа, приписанных свободным переменным типизируемого термина M и всему терму.

19.2 Главный тип (Principle Type)

Определение 31: Главный тип

Для $M \in \Lambda_0$ **главным типом** называют тип σ , такой что

- $M: \sigma$
- $M: \sigma' \implies \exists S: S(\sigma) \equiv \sigma'$

Следствие 6 (Следствие теоремы Хиндли – Милнера). Существует алгоритм PT, возвращающий для $M \in \Lambda_0$ главный тип σ , если M имеет тип и сообщение об ошибке в противном случае.