Билеты к экзамену по функциональному программированию

Тамарин Вячеслав

January 25, 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.

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

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.

```
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.

```
factorial 3
-> if 3 == 0 then 1 else 3 * factorial (3 - 1)
-> ...
-> 3 * factorial (3 - 1)
-> 3 * if (3 - 1) == 0 then 1 else (3 - 1) * factorial ((3 - 1) - 1)
-> ...
-> 3 * 2 * 1 * 1
-> 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-абстрацкия (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 $V = \{x, y, z, \ldots\}$ 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:

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

2.1 Соглашения

Общеприняты следующие соглашения для термов:

- Внешние скобки опускаются
- Применение ассоциативно *влево*:
$$FXYZ \equiv (((FX)Y)Z).$$
- Абстракция ассоциативна *вправо*:
$$\lambda xyz.\, M \equiv (\lambda x.\, (\lambda y.\, (\lambda z.\, M))).$$
- 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: Связвнные переменные

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

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

2.2 Комбинаторы

Определение 5: Комбинатор

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

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

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 $M, N, L \in \Lambda$, $x \not\equiv y$ 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 экстенсиональная эквивалентность.

4.1 α-эквивалентность

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

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

Аналогично β-эквивалентности можно определить совместимость.

4.2 β-эквивалентность

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

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

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

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

Также определим правила совместимости:

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

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

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

4.3 η-эквивалентность

Схема аксиом η-преобразования:

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

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 экстенсиональности Две функции считаются **экстенсионально эквивалентными**, если они дают одинаковый результат при одинаковом входе:

$$\forall N\colon FN =_\beta GN.$$

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

$$Fy =_\beta Gy \Longrightarrow \lambda y.\, Fy =_\beta \lambda y.\, Gy \Longrightarrow F =_{\beta\eta} G$$

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

5.1 Булевы значения

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

5.2 Булевы операции

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

5.3 Кортежи

Пара и стандартные операции:

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

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:

$$\mathtt{iszero} \equiv \lambda n.\ n(\lambda x.\ \mathtt{fls})\,\mathtt{tru}.$$

Переход к следующему числу:

$$\mathtt{succ} \equiv \lambda nsz.\ s(nsz).$$

Сложение:

$$\mathtt{plus} \equiv \lambda mnsz.\ ms(nsz).$$

Умножение:

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

7 Теорема о неподвижной точке. Y-комбинатор.

7.1 Решение уравнений на термы

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

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

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:

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

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 $\mathbf{Y}$ такой, что

$$\forall F\colon F(\mathbf{Y}F) = \mathbf{Y}F.$$

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

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

□

7.3 Рекурсия

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

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

Можно переписать:

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

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

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

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

$$\begin{aligned} \mathrm{KI} &\equiv (\lambda xy.\, x)\,(\lambda z.\, z) = \lambda yz.\, z \\ \mathrm{IIK}_* &\equiv (\lambda x.\, x)\mathrm{IK}_* = \mathrm{IK}_* = (\lambda x.\, x)(\lambda yz.\, z) = \lambda yz.\, z \end{aligned}$$

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 $M, N, Z \in \Lambda$:

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

Определение 8

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

Определение 9

Бинарное отношение **β-редукции за один шаг** $\to_\beta$ над Λ:

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

Это правило и совместимость.

Определение 10: β-редукция

Бинарное отношение **β-редукции** $\twoheadrightarrow_\beta$ над Λ определяется индуктивно:

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

Это отношение — транзитивное, рефлексивное замыкание $\longrightarrow_\beta$, поэтому является отношением редукции.

Определение 11: Конвертируемость

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

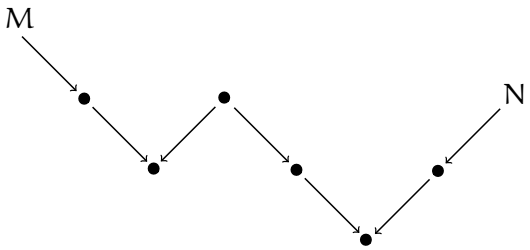
$$M \twoheadrightarrow_\beta N \Rightarrow M =_\beta N \quad \text{(ini)}$$
$$M =_\beta N \Rightarrow N =_\beta M \quad \text{(sym)}$$
$$M =_\beta N,\ N =_\beta L \Rightarrow M =_\beta L \quad \text{(trans)}$$

Это отношение является отношением конгруэнтности.

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

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

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



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

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.

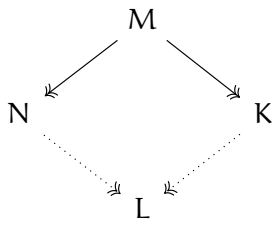
Пример 2.

- $\lambda xy.\, x(\lambda z.\, zx)y$ находится в β-NF
- $(\lambda x.\, xx)y$ не находится в β-NF, но имеет в качестве β-NFтерм 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*:



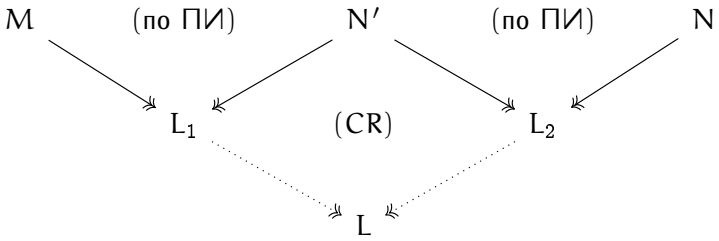
Следствие 1 (о существовании общего редукта). Если $M =_\beta N$, то существует L такой, что $M \twoheadrightarrow_\beta L$ и $N \twoheadrightarrow_\beta L$.

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

$M \twoheadrightarrow_\beta N$. 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 $M \twoheadrightarrow_\beta N$.

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_j', 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:

$$\lambda \vec{x}.\, y\vec{N} \equiv \lambda x_1 x_2 \dots x_n.\, yN_1, N_2, \dots N_k, \quad n, k \geqslant 0 \tag{HNF}$$
$$\lambda \vec{x}.\, (\lambda z.\, M)\vec{N} \equiv \lambda x_1 x_2 \dots x_n.\, (\lambda z.\, M)N_1, N_2, \dots N_k, \quad n \geqslant 0, k > 0$$

Определение 13

Головная нормальная форма — форма следующего вида:

$$\lambda \vec{x}.\, y\vec{N} \equiv \lambda x_1 x_2 \dots x_n.\, yN_1, N_2, \dots N_k, \quad n, k \geqslant 0. \tag{HNF}$$

Слабая головная нормальная форма (*W*HNF) — это HFN 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:

$$(\lambda xy.\, y)N \longrightarrow_\beta \lambda y.\, y.$$

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* — аппликативый порядок редукций до *W*HNF
- *вызов по имени* — нормальный порядок редукций до *W*HNT
- *вызов по необходимости* — „вызов по имени" с разделением.

11 Функция предшествования для чисел Черча. Комбинатор примитивной рекурсии.

11.1 Функция предшествования

Введем вспомогательные функции:

$$\begin{aligned} \mathtt{zp} &\equiv \mathtt{pair}\,0\,0 \\ \mathtt{sp} &\equiv \lambda\,\mathtt{p}.\,\mathtt{pair}(\mathtt{snd}\,\mathtt{p})(\mathtt{succ}(\mathtt{snd}\,\mathtt{p})) \end{aligned}$$

Вторая функция позволяет, приняв пару $(\_, \mathtt{j})$, вернуть пару $(\mathtt{j}, \mathtt{j}+1)$:

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

Если взять композицию $\mathtt{m} > 0$ раз от zp:

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

Тогда можно определить функцию предшествования:

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

Тогда

$$\mathtt{minus} = \lambda\,\mathtt{m}\,\mathtt{n}.\,\mathtt{n}\ \mathtt{pred}\,\mathtt{m}.$$

11.2 Комбинатор примитивной рекурсии

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

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

Теперь можно выразить pred:

$$\mathtt{pred}' = \lambda\,\mathtt{n}.\,\mathtt{rec}\,\mathtt{n}\,(\mathtt{K}_*)\,0.$$

И факториал:

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

11.3 Списки

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

Теперь

$$\begin{aligned} [\,] &= \mathtt{nil} \\ [5,3,2] &= \mathtt{cons}\,5(\mathtt{cons}\,3(\mathtt{cons}\,2\,\mathtt{nil})) = \lambda\,\mathtt{c}\,\mathtt{n}.\,\mathtt{c}\,5(\mathtt{c}\,3(\mathtt{c}\,2\,\mathtt{n})) \end{aligned}$$

$$\mathtt{cons} \equiv \lambda\,\mathtt{l}.\,\mathtt{l}\,(\lambda\,\mathtt{h}\,\mathtt{t}.\,\mathtt{fls})\,\mathtt{tru}.$$

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 — (WH)NF

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

$$M : \sigma.$$

12.2 Для чего нужны типы?

- Частичная спецификация:

$$f : \mathbb{N} \to \mathbb{N}, \qquad g : (\forall n : \mathbb{N}.\ \exists m : \mathbb{N}.\ m \leqslant n).$$

- 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:

$$I : \alpha \to \alpha.$$

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

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

Гипотеза о типе переменных записывают в контексте:

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

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 $\lambda_\to$ определяется индуктивно:

$$\alpha, \beta \ldots \in \mathbb{T} \qquad \text{(переменные типа)}$$
$$\sigma, \tau \in \mathbb{T} \Rightarrow (\sigma \to \tau) \in \mathbb{T} \qquad \text{(типы пространства функций)}$$

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

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

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

Всякий тип может быть записан в виде:

$$\sigma_1 \to \dots \to \sigma_n \to \alpha.$$

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 $\lambda_\to$ по Карри имеет вид $M\colon \tau$, где субъект $M \in \Lambda$ и предикат $\tau \in \mathbb{T}$.

Пример 3. Примеры утверждений типизации

$$\begin{aligned} &\lambda x.\, x\colon \alpha \to \alpha \\ &\lambda x.\, x\colon (\alpha \to \beta) \to \alpha \to \beta \\ &\lambda x\, y.\, x\colon \alpha \to \beta \to \alpha \end{aligned}$$

Определение 17: Объвление

Объявление — утверждение типизации с термовой переменной в кресле субъекта.

Пример 4. Примеры объявлений

$$\begin{aligned} &x\colon \alpha \\ &y\colon \beta \\ &f\colon (\alpha \to \beta) \to \gamma \end{aligned}$$

Определение 18: Контекст

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

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

Контекст иногда называют *базисом* или *окружением*.

Фигурные скобки иногда опускают:

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

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

$$\Gamma \vdash M\colon \tau,$$

если его вывод может быть произведен по правилам:

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

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

Запись в виде дерева вывода:

$$\Gamma \vdash x\colon \sigma,\ \text{если } x^\alpha \in \Gamma \qquad \text{(аксиома)}$$

$$\frac{\Gamma \vdash M\colon \sigma \to \tau \qquad \Gamma \vdash N\colon \sigma}{\Gamma \vdash MN\colon \tau} \qquad (\to \mathrm{Elim})$$

$$\frac{\Gamma,\ x^\alpha \vdash M\colon \tau}{\Gamma \vdash \lambda x.\, M\colon \sigma \to \tau} \qquad (\to \mathrm{Intro})$$

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 $V = \{x, y, z, \ldots\}$ 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 $\lambda_\to$ по Черчу имеет вид $M\colon \tau$, где субъект $M \in \Lambda^{\mathbb{T}}$ и предикат $\tau \in \mathbb{T}$.

Пример 5. Примеры утверждений типизации

$$\lambda x^\alpha.\, x\colon \alpha \to \alpha$$
$$\lambda x^{\alpha\to\beta}.\, x\colon (\alpha \to \beta) \to \alpha \to \beta$$
$$\lambda x^\alpha y^\beta.\, x\colon \alpha \to \beta \to \alpha$$

13.2 Правила типизации

Определение 22

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

$$\Gamma \vdash M\colon \tau,$$

если его вывод может быть произведен по правилам:

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

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

Запись в виде дерева вывода:

$$\Gamma \vdash x\colon \sigma,\ \text{если } x^\alpha \in \Gamma \qquad \text{(аксиома)}$$

$$\frac{\Gamma \vdash M\colon \sigma \to \tau \qquad \Gamma \vdash N\colon \sigma}{\Gamma \vdash MN\colon \tau} \qquad (\to \text{Elim})$$

$$\frac{\Gamma,\ x^\alpha \vdash M\colon \tau}{\Gamma \vdash \lambda x^\alpha.\, M\colon \sigma \to \tau} \qquad (\to \text{Intro})$$

14 Свойства систем просто типизированного λ-исчисления

Лемма 2 (об инверсии или о генерации).

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

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

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

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

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

Расширение контекста не влияет на выводимость утверждения типизации.

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

$$\Gamma \vdash M\colon \sigma \implies \mathrm{FV}(M) \subset \mathtt{dom}(\Gamma).$$

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

$$\Gamma \vdash M\colon \sigma \implies \Gamma|_{\mathrm{FV}(M)} \vdash M\colon \sigma.$$

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 $\sigma \to \tau$.

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

Тогда ω, Ω и $\mathbf{Y}$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.

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

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

Карри:

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

Черч:

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

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

$$x^{\alpha} \vdash \lambda y^{\alpha} z^{\beta}.\, x : \alpha \to \beta \to \alpha$$

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

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

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

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

и подставляем в него вместо свободной переменной x типа $\gamma \to \gamma$ терм $\lambda z^{\gamma}.\, z$ подходящего типа $\gamma \to \gamma$. Получаем

$$\vdash \lambda y^{\beta} z^{\gamma}.\, z : \beta \to \gamma \to \gamma.$$

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

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

Тип терма сохраняется про β-редукциях.

Proof. Следует из леммы о подстановке терма. □

Следствие 4. Множество типизируемых в $\lambda_{\to}$ термов замкнуто относительно редукции.

В обратную сторону теорема и следствие не верны для $\lambda_{\to}$.

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

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

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

$$\vdash \lambda x\, y.\, x\colon \alpha \to (\delta \to \gamma \to \delta) \to \alpha$$
$$\vdash \lambda x\, y.\, x\colon (\gamma \to \gamma) \to \beta \to \gamma \to \gamma$$

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

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

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

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

$$M \in \Lambda_{\mathbb{T}} \wedge \Gamma \vdash M\colon \sigma \Longrightarrow \Gamma \vdash |M|\colon \sigma.$$

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

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

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 $\lambda_{\to}$ все эти задачи разрешимы. Первые две задачи будут

$\vdash M\colon \sigma$?	Задача проверки типа	ЗПТ, TCP
$\vdash M\colon ?$	Задача синтеза типа	ЗСТ, TSP / TAP
$\vdash ?\colon \sigma$	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 $\lambda_{\to}$). Обе системы (по Карри и по Черчу) сильно нормализуемы.

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