Билеты к экзамену по функциональному программированию

Тамарин Вячеслав

January 24, 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:

$$\begin{array}{lcl} \mathtt{subterms}(x) & = & \{x\} \\ \mathtt{subterms}(MN) & = & \{MN\} \cup \mathtt{subterms}(M) \cup \mathtt{subterms}(N) \\ \mathtt{subterms}(\lambda x.\, M) & = & \{\lambda x.\, M\} \cup \mathtt{subterms}(M) \end{array}$$

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:

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

Определение 4: Связвнные переменные

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

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

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:

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

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:

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

Аналогично β-эквивалентности можно определить совместимость.

4.2 β-эквивалентность

4.2.1 β-редукция

Одношаговая β-редукция для редекса $(\lambda x.\, M)N$:

$$(\lambda x.\, M)N \longrightarrow_\beta [x \mapsto N]M.$$

Сокращение β-редекса — результат подстановки в правой части:

$$Iz = (\lambda x.\, x)z \longrightarrow_\beta [x \mapsto z]x = z.$$

Дополнение определения одношаговой β-редукции:

$$\begin{array}{lcl}
M \longrightarrow_\beta N & \Rightarrow & ZM \longrightarrow_\beta ZN \\
M \longrightarrow_\beta N & \Rightarrow & MZ \longrightarrow_\beta NZ \\
M \longrightarrow_\beta N & \Rightarrow & \lambda x.\, M \longrightarrow_\beta \lambda x.\, N
\end{array}$$

4.2.2 Многошаговая β-редукция

Это рефлексивное транзитивное замыкание одношаговой:

$$\begin{aligned} M \longrightarrow_\beta N \quad &\Rightarrow \quad M \twoheadrightarrow_\beta N \\ &\quad\;\;\; M \twoheadrightarrow_\beta M \\ M \twoheadrightarrow_\beta N,\ N \twoheadrightarrow_\beta L \quad &\Rightarrow \quad M \twoheadrightarrow_\beta L \end{aligned} \tag{правила совместимости}$$

Определение 7: β-эквивалентность

Отношение **β-эквивалентности (β-конвертируемости)** (нотация $=_\beta$) — симметричное транзитивное замыкание многошаговой редукции:

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

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

$$(\lambda x.\, M)N =_\beta [x \mapsto N]M.$$

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

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

4.3 η-эквивалентность

Схема аксиом η-преобразования:

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

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

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