Функциональное программирование

Денис Николаевич Москвин

6 сентября 2022 г.

Содержание

Все презентации можно найти тут тык.

1. Лямбда-исчисление

1.1. Функциональная модель вычислений

Типы прогрмаммирования:

1. **Императивное** – инструкции выполняются последовательно (общение с вычислителем). Результат – выполнение последней инструкции.

2. **Функциональное** – программа это выражение, его выполнение это вычисление (редукция) выражения. Результат – отсутствие редексов (подвыражения, которые могут быть вычислены непосредственно).

***Определение* 1.1.** Связывание – символ равенства ($a = 2 \cdot 7 + 1$), имя слева становится редексом (будет происходить подстановка наряду со встроенными правилами).

Пример.

$$z \cdot 4 + 1 \to (2 \cdot 7 + 1) \cdot 4 + 1 \to \dots$$

***Определение* 1.2.** Рекурсивное связывание – символ равенства ($x = 2 \cdot x + 1$), имя слева становится редексом, такие выражения расходятся (так как нет терминирующего условия).

Пример.

$$x \cdot 2 \to (2 \cdot x + 1) \cdot 2 \to \dots$$

***Определение* 1.3.** Лямбда абстракция (анонимная функция) – $\lambda \underbrace{y}_{\text{абстрактор}} \to \underbrace{2 \cdot y + 3}_{\text{тело}}$, чтобы применить функцию к аргументу, то мы записываем справа от тело аргумент.

***Определение* 1.4.** Вычисление (β-редукция) – просто подстановка вместо абстрактора самого аргумента.

Пример. Заведем функцию:

$$f = \lambda y \to 2 \cdot y + 1$$

Стретегии редукции:

1. В Haskell используется **ленивая** стратегия: сокращается самый левый внешний редекс: $(\lambda y \to 2 \cdot y + 3)(4 + 6) \to_\beta 2 \cdot (4 + 6) + 3 \to (8 + 12) + 3 \to 20 + 3 \to 23$

2. Была еще **энергетическая**, но я не успел. Кто-нибудь добавьте, если хочется.

Пример. Тут был пример с факториалом.

***Определение* 1.5.** Функция нескольких переменных – $\lambda n \to 2 \cdot m + 3 \cdot n$, тут свободная переменная это m, можно продолжить выражение, чтобы полуяиться замкунутое выражения (все переменные связанные): $\lambda m \to (\lambda n \to 2 \cdot m + 3 \cdot n)$.

Вызываем функцию так: $(\lambda m \to (\lambda n \to 2 \cdot m + 3 \cdot n)15)4$ – вместо m подставится 15, а вместо n – 4.

1.2. Чистое λ-исчисление

***Определение* 1.6.** λ-терм – переменная, либо апликация, либо абстракция.

$$x \in V \implies x \in \Lambda$$

$$M, N \in \Lambda \implies (MN) \in \Lambda$$

$$M \in \Lambda, x \in V \implies (\lambda x, M) \in \Lambda$$

Пример. λ-термы:

1. x
2. (x z)
3. (λx. (xz))
4. ((λx. (xz)) y)
5. ...

Каждый следующий терм содержит предыдущий как подтерм.

Замечание. Имеются следующий обозначения:

1. Внешние скобки опускаются
2. Применение ассоциативно влево: $FXYZ == ((FX)Y)Z$
3. Абстракция ассоциативна вправо: $\lambda xyz == \lambda x.(\lambda y.(\lambda z.M))$

***Определение* 1.7.** β-редукция – $(\lambda x.M)N \to_\beta [x \mapsto N]M$ – подстановка N вместо x в M.

***Определение* 1.8.** Применение вида $(\lambda x.M)N$, в которой левый аппликанд является абстракцией, называют β-редексом.

***Определение* 1.9.** Шаг вычисления по приведенному выше правилу называют сокращением редекса.

***Определение* 1.10.** В чистом λ-исчислении нет ничего кроме переменных, применения, абстракции и редукции.

todo

***Определение* 1.11.** Множество $FV(T)$ свободных переменных в терме T:

$$FV(x) \implies \{x\}$$

$$FV(MN) \implies FV(M) \cup FV(N)$$

$$FV(\lambda x.M) = FV(M) \setminus \{x\}$$

***Определение* 1.12.** Множество $BV(T)$ связных переменных в терме T:

$$BV(x) \implies \varnothing$$

$$BV(MN) \implies BV(M) \cup BV(N)$$

$$BV(\lambda x.M) \implies BV(M) \cup \{x\}$$

***Определение* 1.13.** M – замкнутый λ-терм (комбинатор), если $FV(M) = \varnothing$. Множество замкнутых λ-термов обозначается Λ^0.

Пример. I - комбинатор.

$$I = \lambda x.x$$

$$IM \to_I (\lambda x.x)M \to_\beta M$$

Пример.

$$\omega = \lambda x.xx$$

$$\Omega = \omega\omega \to_\Omega (\lambda x.xx)\omega \to_\beta \omega\omega$$

***Определение* 1.14.** α-редукция – $\lambda x.x \to_\alpha \lambda y.y$

1.3. Отношение эквивалентности на термах

***Определение* 1.15.** Подстановка – $[x \mapsto N]M$. Правила подстановки:

1. $[x \mapsto N]x = N$
2. $[x \mapsto N]y = y$
3. $[x \mapsto N](PQ) = ([x \mapsto N]P)([x \mapsto N]Q)$
4. $[x \mapsto N](\lambda x.P) = \lambda x.P$
5. $[x \mapsto N](\lambda y.P) = \lambda y.[x \mapsto N]P : y \notin FV(N)$ – y - не свободная переменная.
6. $[x \mapsto N](\lambda y.P) = \lambda y'.[x \mapsto N]([y \mapsto y']P) : y \in FV(N)$ – иначе.

Лемма. (О подстановке) Подстановки не коммутируют, однако верна $M, N, L \in \Lambda$. Предположим, что $x \not\equiv y$ и $x \notin FV(L)$. Тогда $[y \mapsto L]([x \mapsto N]M) \equiv [x \mapsto [y \mapsto L]N]([y \mapsto L]M)$

Доказательство. Нудная индукция по всем 6ти случаям, с разбором всех подслучаев. □

***Определение* 1.16.** β-эквивалентность (хотим все свойства отношения эквивалентности). $\forall M, N \in \Lambda : (\lambda x.M) =_\beta [x \mapsto N]M$.

Логические аксиомы этого правила:

1. $M =_\beta M$
2. $M =_\beta N \implies N =_\beta M$
3. $M =_\beta N, N =_\beta L \implies M =_\beta L$

Правила совместимости:

1. $M =_\beta M' \implies MZ =_\beta M'Z$
2. $M =_\beta M' \implies ZM =_\beta ZM'$

3. $M =_\beta M' \implies \lambda x.M =_\beta \lambda x.M'$

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

***Определение* 1.17.** α-эквивалентность – $\lambda x.M =_\alpha \lambda y.[x \mapsto y]M$, если $y \notin FV(M)$ (переименование переменных). (там еще была табличка с аксиомами, посмотрите в презентации)

***Определение* 1.18.** η-эквивалентность – $\lambda x.Mx =_\eta M$. Смысл в том, что апликативное поведение термов слева и справа от знака равенства одинаоково: для произвольного N верно: $(\lambda x.Mx)N =_\beta MN$.