Доп. 20 Лямбда-исчисление 2.0

η-редукция

$$\lambda x.fx \xrightarrow[\eta]{} f$$

$$(\lambda y.\ \lambda x.yx) \xrightarrow[\eta]{} \lambda y.y$$

λ-терм находится в нормальной форме, если дальше редуцировать его нельзя

Стратегии для нормализации λ-термов.

→ нормальная

→ β-редукции

→ call by name

→ call by value

Нормализуем: $(\lambda x.x)((\lambda x.x)(\lambda z.(\lambda x.x)\, z))$

1) Норм. порядок вычислений

$$(\lambda x.x)((\lambda x.x)(\lambda z.(\lambda x.x) z)) \to (\lambda x.x)(\lambda z.(\lambda x.x) z) \to (\lambda z.(\lambda x.x) z) \to \lambda z.z \to z$$

2) Полная β-редукция:

$(\lambda x.x)((\lambda x.x)(\lambda z.(\lambda x.x)z)) \to_\beta$

$\to_\beta (\lambda x.x)((\lambda x.x)(\lambda z.z)) \to_\beta$

$\to_\beta (\lambda x.x)(x\,[x := \lambda z.z]) =$

$= (\lambda x.x)(\lambda z.z) \to_\beta x\,(x := \lambda z.z) =$

$= \lambda z.z$

3) Call-by-name – только применение аппликации

$(\lambda x.x)((\lambda x.x)(\lambda z.(\lambda x.x)z)) \to$

$\to (\lambda x.x)(\lambda z.(\lambda x.x)z) \to$

$\to \lambda z.(\lambda x.x)z$ в этом типе вычисления нельзя сокращать (редуцировать) внутри абстракции (в данном случае внутри $(\lambda x.x)$)

4) Call-by-value – внутри абстр. сокр. нельзя
+ начинаем от самой вложенной аппликации

$(\lambda x.x)\underline{((\lambda x.x)(\lambda z.(\lambda x.x)z))} \to$

$\to \underline{(\lambda x.x)(\lambda z.(\lambda x.x)z)} \to$

$\to \lambda z.(\lambda x.x)z$

$(\lambda x.xx)(\lambda x.xx)$ – сам в себя

$(\lambda x.xxx)(\lambda x.xxx)$ – постоянно увеличивается

Программирование и лямбда-исчисление

$(\lambda x.(x+x+x))(10+5) \to$

1) call-by-name

$\to (10+5)+(10+5)+(10+5) \to 45$

БАН за такое!

2) call-by-value

$(\lambda x.(x+x+x))\ 15 \to 15+15+15 \to 45$

Теорема Чёрча-Россера

↳ Для любого выражения E и любых двух последовательностей редукции $E \to E_1$ и $E \to E_2$ существуют две последовательности редукции $E_1 \to E_3$ и $E_2 \to E_3$, приводящие к одному и тому же результату E_3.

$$\begin{array}{ccc}
\overset{E}{(\lambda x.\ ax)((\lambda y.by)\ c)} & \xrightarrow[\text{name}]{} & a((\lambda y.\ by)\ c) \quad E_1 \\
\downarrow \text{value} & & \downarrow \\
(\lambda x.\ ax)\ bc \quad E_2 & & a(bc) \quad E_3 \\
\downarrow & \nearrow & \\
a(bc) \quad E_3 & &
\end{array}$$

Комбинаторы

↳ Комбинатор — λ-терм без констант и свободных переменных

Комбинаторный базис

$I = \lambda x . x$ → комбинатор тождества (т.к. резуль- равен значению)

$K = \lambda xy . x$ → канцелятор

$S = \lambda xyz . (xz)(yz)$ → коннектор

Комбинаторное исчисление (комбинаторные редукции).

- $(Ia) \Rightarrow_c^I a \qquad (\lambda x . x) a \to a$
- $(Kab) \Rightarrow_c^K a \qquad (\lambda xy . x) ab \to a$
- $(Sabc) \Rightarrow_c^S (ac)(bc) \qquad (\lambda xyz . (xz)(yz)) \to (xz)(yz)$

Числа Чёрча

→ числа обозначают кол-во аппликаций в λ-терме

$\underline{0} = \lambda f x . x$

$\underline{1} = \lambda f x . f(x)$

$\underline{2} = \lambda f x . f(f(x))$

Арифметические операции

$plus(m, n) = m + n$

вложенность $m = \lambda f x . \underbrace{f(f(f \dots (x)))}_{m}$

вложенность $n = \lambda f x . \underbrace{f(f(f \dots (x)))}_{n}$

вложенность $m + n = \lambda f x . \underbrace{f(f(f}_{m} \dots \underbrace{f(f(f \dots (x)))}_{n})))$

$1 + 2$

$\underline{1} = \lambda f x . f(x)$

$\underline{2} = \lambda f x . f(f(x))$

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

$\lambda m . \lambda n . \lambda f x . \ m f (n f x)$

сложение

$mult(m, n) = m \cdot n$

$\underline{2} = \lambda f x.\ f(f(x))$

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

$\underline{6} = \lambda f x.\ f(f(f(f(f(f(x))))))$

$\lambda m.\ \lambda n.\ \lambda f x.\ m(n f x)$

{умножение}

$succ(n) = n + 1$

$succ(1) = 2 = \lambda f x. f(f(x))$

$\lambda n.\ \lambda f x.\ f(n f x)$

Можно обойтись без комбинатора I !

$S = \lambda xyz.\ (xz)(yz)$
$K = \lambda xy.\ x$
$I = \lambda x. x$

$(SKK)x = x$

$Ix = x$

$SKKx = (\lambda xyz.\ (xz)(yz))\ KKx =$
$= (Kx)\ (Kx) = (\lambda xy.x)xKx =$
$= (\lambda xy.x)x(\lambda xy.x)x =$