Лекция 9

Основы λ-исчисления

Функциональное и логическое программирование

КамчатГТУ, 2011 г.

1. Базис λ-исчисления
   - Определения
   - Правила вывода

2. λ-исчисление, как язык программирования
   - Управляющие операторы
   - Абстракция данных
   - Арифметика
   - Рекурсия

1. Базис λ-исчисления
   - Определения
   - Правила вывода

2. λ-исчисление, как язык программирования
   - Управляющие операторы
   - Абстракция данных
   - Арифметика
   - Рекурсия

Что мы уже знаем о λ-функциях?

- λ-функции – это основное средство создания функций в ФЯП.

Что мы уже знаем о λ-функциях?

- λ-функции – это основное средство создания функций в ФЯП.
- Определение λ-функции состоит из перечисления её формальных аргументов и тела функции:

$$(\lambda(x)\ (*\ x\ x)), \quad (\lambda(f\ g)\ (\lambda(x)\ (f\ (g\ x))))$$

Что мы уже знаем о λ-функциях?

- λ-функции – это основное средство создания функций в ФЯП.
- Определение λ-функции состоит из перечисления её формальных аргументов и тела функции:

$$(\lambda(x)\ (*\ x\ x)), \quad (\lambda(f\ g)\ (\lambda(x)\ (f\ (g\ x))))$$

- Какой именно символ используется в качестве формального аргумента не имеет значения:

$$(\lambda(x)\ (*\ x\ x)) \cong (\lambda(y)\ (*\ y\ y)).$$

Что мы уже знаем о λ-функциях?

- λ-функции – это основное средство создания функций в ФЯП.
- Определение λ-функции состоит из перечисления её формальных аргументов и тела функции:

$$(\lambda(x)\ (*\ x\ x)),\quad (\lambda(f\ g)\ (\lambda(x)\ (f\ (g\ x))))$$

- Какой именно символ используется в качестве формального аргумента не имеет значения:

$$(\lambda(x)\ (*\ x\ x)) \cong (\lambda(y)\ (*\ y\ y)).$$

- Вычисление λ-функции состоит в замене формальных аргументов фактическими:

$$((\lambda(x)\ (*\ x\ x))\ 2) = (*\ 2\ 2) = 4$$

$$((\lambda(f\ g)\ (\lambda(x)\ (f\ (g\ x))))\ not\ odd?) =$$
$$= (\lambda(x)\ (not\ (odd?\ x)))$$

Что мы уже знаем о λ-функциях?

- λ-функции – это основное средство создания функций в ФЯП.
- Определение λ-функции состоит из перечисления её формальных аргументов и тела функции:

$$(\lambda(x)\ (*\ x\ x)),\quad (\lambda(f\ g)\ (\lambda(x)\ (f\ (g\ x))))$$

- Какой именно символ используется в качестве формального аргумента не имеет значения:

$$(\lambda(x)\ (*\ x\ x)) \cong (\lambda(y)\ (*\ y\ y)).$$

- Вычисление λ-функции состоит в замене формальных аргументов фактическими:

$$((\lambda(x)\ (*\ x\ x))\ 2) = (*\ 2\ 2) = 4$$

$$((\lambda(f\ g)\ (\lambda(x)\ (f\ (g\ x))))\ not\ odd?) =$$
$$= (\lambda(x)\ (not\ (odd?\ x)))$$

- Функции эквивалентны, тогда и только тогда, когда их результаты совпадают на всей области определения:

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

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Базис λ-исчисления:

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Базис λ-исчисления:

1. символы переменных: $x, y, \ldots$;

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Базис λ-исчисления:

1. символы переменных: $x, y, \ldots$;
2. символ абстракции: λ;

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Базис λ-исчисления:

1. символы переменных: $x, y, ...$;
2. символ абстракции: λ;
3. скобки: (и).

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Базис λ-исчисления:

1. символы переменных: $x, y, \ldots$;
2. символ абстракции: λ;
3. скобки: (и).

Базовые операции:

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Базис λ-исчисления:

1. символы переменных: $x, y, \ldots$;
2. символ абстракции: λ;
3. скобки: (и).

Базовые операции:

абстракция — конструирование функции,

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Базис λ-исчисления:

1. символы переменных: $x, y, \ldots$;
2. символ абстракции: λ;
3. скобки: (и).

Базовые операции:

абстракция — конструирование функции,

аппликация — применение функции к аргументам.

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Базис λ-исчисления:

1. символы переменных: $x, y, \ldots$;
2. символ абстракции: λ;
3. скобки: (и).

Базовые операции:

абстракция — конструирование функции,

аппликация — применение функции к аргументам.

Определение

Базовая единица λ-исчисления — λ-терм.

$$\begin{array}{lcl} \langle\Lambda\rangle & ::= & \langle id\rangle \\ & | & \lambda.\langle id\rangle\,\langle\Lambda\rangle \\ & | & \langle\Lambda\rangle\,\langle\Lambda\rangle \end{array}$$

Определения

Определение

λ-исчисление – система для формализации и анализа понятия вычислимости.

Базис λ-исчисления:

1. символы переменных: $x, y, \ldots$;
2. символ абстракции: λ;
3. скобки: (и).

Базовые операции:

абстракция — конструирование функции,

аппликация — применение функции к аргументам.

Определение

Базовая единица λ-исчисления — λ-терм.

$$\begin{array}{lcl} \langle\Lambda\rangle & ::= & \langle id\rangle \\ & | & \lambda.\langle id\rangle\,\langle\Lambda\rangle \\ & | & \langle\Lambda\rangle\,\langle\Lambda\rangle \end{array}$$

Примеры:

x, $\lambda.x\,x$, $\lambda.x\,(f\,x)$, $(\lambda.x\,y)\,z$, $\lambda.f\,\lambda.x\,(f\,(fx))$

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ связывает аргумент x в теле E определяемой функции.

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ связывает аргумент x в теле E определяемой функции.

Символ x называется связанным.

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ связывает аргумент x в теле E определяемой функции.

Символ x называется связанным.

Не связанные символы в теле функции называются свободными:

$$\begin{aligned} FV(x) &= \{x\}, \text{если } x \text{ – переменная}, \\ FV(\lambda.x\,E) &= FV(E) \setminus \{x\}, \\ FV(A\,B) &= FV(A) \cup FV(B) \end{aligned}$$

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ **связывает** аргумент x в теле E определяемой функции.

Символ x называется **связанным**.

Не связанные символы в теле функции называются **свободными**:

$$\begin{aligned} FV(x) &= \{x\}, \text{если } x \text{ – переменная}, \\ FV(\lambda.x\,E) &= FV(E) \setminus \{x\}, \\ FV(A\,B) &= FV(A) \cup FV(B) \end{aligned}$$

Определение

λ-терм не имеющий свободных аргументов называется **замкнутым термом** или **комбина́тором**.

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ **связывает** аргумент x в теле E определяемой функции.

Символ x называется **связанным**.

Не связанные символы в теле функции называются **свободными**:

$$
\begin{aligned}
FV(x) &= \{x\}, \text{если } x \text{ – переменная},\\
FV(\lambda.x\,E) &= FV(E) \setminus \{x\},\\
FV(A\,B) &= FV(A) \cup FV(B)
\end{aligned}
$$

Определение

λ-терм не имеющий свободных аргументов называется **замкнутым термом** или **комбина́тором**.

Примеры:

Примеры комбинаторов:

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ связывает аргумент x в теле E определяемой функции.

Символ x называется связанным.

Не связанные символы в теле функции называются свободными:

$$\begin{aligned} FV(x) &= \{x\}, \text{если } x \text{ – переменная}, \\ FV(\lambda.x\,E) &= FV(E) \setminus \{x\}, \\ FV(A\,B) &= FV(A) \cup FV(B) \end{aligned}$$

Определение

λ-терм не имеющий свободных аргументов называется замкнутым термом или комбина́тором.

Примеры:

Примеры комбинаторов:

$I = \lambda.x\,x$

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ связывает аргумент x в теле E определяемой функции.

Символ x называется связанным.

Не связанные символы в теле функции называются свободными:

$$\begin{aligned} FV(x) &= \{x\}, \text{если } x \text{ – переменная}, \\ FV(\lambda.x\,E) &= FV(E) \setminus \{x\}, \\ FV(A\,B) &= FV(A) \cup FV(B) \end{aligned}$$

Определение

λ-терм не имеющий свободных аргументов называется замкнутым термом или комбина́тором.

Примеры:

Примеры комбинаторов:

$I = \lambda.x\,x$
$K = \lambda.x\,\lambda.y\,x$

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ **связывает** аргумент x в теле E определяемой функции.

Символ x называется **связанным**.

Не связанные символы в теле функции называются **свободными**:

$$
\begin{aligned}
FV(x) &= \{x\}, \text{если } x \text{ – переменная},\\
FV(\lambda.x\,E) &= FV(E) \setminus \{x\},\\
FV(A\,B) &= FV(A) \cup FV(B)
\end{aligned}
$$

Определение

λ-терм не имеющий свободных аргументов называется **замкнутым термом** или **комбина́тором**.

Примеры:

Примеры комбинаторов:

$I = \lambda.x\,x$

$K = \lambda.x\,\lambda.y\,x$

$\circ = \lambda.f\,\lambda.g\,\lambda.x\,(f\,(g\,x))$

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ связывает аргумент x в теле E определяемой функции.

Символ x называется связанным.

Не связанные символы в теле функции называются свободными:

$$FV(x) = \{x\}, \text{если } x \text{ – переменная},$$
$$FV(\lambda.x\,E) = FV(E) \setminus \{x\},$$
$$FV(A\,B) = FV(A) \cup FV(B)$$

Определение

λ-терм не имеющий свободных аргументов называется замкнутым термом или комбина́тором.

Примеры:

Примеры комбинаторов:

$I = \lambda.x\,x$

$K = \lambda.x\,\lambda.y\,x$

$\circ = \lambda.f\,\lambda.g\,\lambda.x\,(f\,(g\,x))$

$\omega = \lambda.x\,(x\,x)$

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ **связывает** аргумент x в теле E определяемой функции.

Символ x называется **связанным**.

Не связанные символы в теле функции называются **свободными**:

$$\begin{aligned} FV(x) &= \{x\}, \text{если } x \text{ – переменная}, \\ FV(\lambda.x\,E) &= FV(E) \setminus \{x\}, \\ FV(A\,B) &= FV(A) \cup FV(B) \end{aligned}$$

Определение

λ-терм не имеющий свободных аргументов называется **замкнутым термом** или **комбина́тором**.

Примеры:

Примеры комбинаторов:

$I = \lambda.x\,x$
$K = \lambda.x\,\lambda.y\,x$
$\circ = \lambda.f\,\lambda.g\,\lambda.x\,(f\,(g\,x))$
$\omega = \lambda.x\,(x\,x)$
$dup = \lambda.f\,\lambda.x\,(f\,(f\,x))$

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ **связывает** аргумент x в теле E определяемой функции.

Символ x называется **связанным**.

Не связанные символы в теле функции называются **свободными**:

$$\begin{aligned} FV(x) &= \{x\}, \text{если } x \text{ – переменная}, \\ FV(\lambda.x\,E) &= FV(E) \setminus \{x\}, \\ FV(A\,B) &= FV(A) \cup FV(B) \end{aligned}$$

Определение

λ-терм не имеющий свободных аргументов называется **замкнутым термом** или **комбина́тором**.

Примеры:

Примеры комбинаторов:

$I = \lambda.x\,x$
$K = \lambda.x\,\lambda.y\,x$
$\circ = \lambda.f\,\lambda.g\,\lambda.x\,(f\,(g\,x))$
$\omega = \lambda.x\,(x\,x)$
$dup = \lambda.f\,\lambda.x\,(f\,(f\,x))$

Пример незамкнутых λ-термов:

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ связывает аргумент x в теле E определяемой функции.

Символ x называется связанным.

Не связанные символы в теле функции называются свободными:

$$\begin{aligned} FV(x) &= \{x\}, \text{если } x \text{ – переменная},\\ FV(\lambda.x\,E) &= FV(E) \setminus \{x\},\\ FV(A\,B) &= FV(A) \cup FV(B) \end{aligned}$$

Определение

λ-терм не имеющий свободных аргументов называется замкнутым термом или комбина́тором.

Примеры:

Примеры комбинаторов:

$I = \lambda.x\,x$
$K = \lambda.x\,\lambda.y\,x$
$\circ = \lambda.f\,\lambda.g\,\lambda.x\,(f\,(g\,x))$
$\omega = \lambda.x\,(x\,x)$
$dup = \lambda.f\,\lambda.x\,(f\,(f\,x))$

Пример незамкнутых λ-термов:

$\lambda.x\,(f\,(g\,x))$

Свободные аргументы

Оператор абстракции $\lambda.x\,E$ **связывает** аргумент x в теле E определяемой функции.

Символ x называется **связанным**.

Не связанные символы в теле функции называются **свободными**:

$$
\begin{aligned}
FV(x) &= \{x\}, \text{если } x \text{ – переменная},\\
FV(\lambda.x\,E) &= FV(E) \setminus \{x\},\\
FV(A\,B) &= FV(A) \cup FV(B)
\end{aligned}
$$

Определение

λ-терм не имеющий свободных аргументов называется **замкнутым термом** или **комбина́тором**.

Примеры:

Примеры комбинаторов:

$I = \lambda.x\,x$
$K = \lambda.x\,\lambda.y\,x$
$\circ = \lambda.f\,\lambda.g\,\lambda.x\,(f\,(g\,x))$
$\omega = \lambda.x\,(x\,x)$
$dup = \lambda.f\,\lambda.x\,(f\,(f\,x))$

Пример незамкнутых λ-термов:

$\lambda.x\,(f\,(g\,x))$
$C_y = \lambda.x\,y$

Правила вывода в λ-исчислении

α-преобразование (α-эквивалентность)

Выражение $\lambda.x\,A$ можно преобразовать к виду $\lambda.y\,B$ заменой x на y в A, если y не является свободной в A:

$$\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,B,\ \text{если}\ B = A[x \to y]\ \text{и}\ y \notin FV(A).$$

Правила вывода в λ-исчислении

α-преобразование (α-эквивалентность)

Выражение $\lambda.x\,A$ можно преобразовать к виду $\lambda.y\,B$ заменой x на y в A, если y не является свободной в A:

$$\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,B, \text{ если } B = A[x \to y] \text{ и } y \notin FV(A).$$

Определение

Подстановкой $A[x \to y]$ (формальной заменой x на y в выражении A) называется рекурсивное преобразование :

Правила вывода в λ-исчислении

α-преобразование (α-эквивалентность)

Выражение $\lambda.x\,A$ можно преобразовать к виду $\lambda.y\,B$ заменой x на y в A, если y не является свободной в A:

$$\lambda.x\,A \leftrightarrow_{\alpha} \lambda.y\,B,\ \text{если } B = A[x \rightarrow y] \text{ и } y \notin FV(A).$$

Определение

Подстановкой $A[x \rightarrow y]$ (формальной заменой x на y в выражении A) называется рекурсивное преобразование :

$$x[x \rightarrow X] \equiv X$$

Правила вывода в λ-исчислении

α-преобразование (α-эквивалентность)

Выражение $\lambda.x\,A$ можно преобразовать к виду $\lambda.y\,B$ заменой x на y в A, если y не является свободной в A:

$$\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,B, \text{ если } B = A[x \to y] \text{ и } y \notin FV(A).$$

Определение

Подстановкой $A[x \to y]$ (формальной заменой x на y в выражении A) называется рекурсивное преобразование :

$$x[x \to X] \equiv X$$

$$y[x \to X] \equiv y, \text{ если } y \neq x$$

Правила вывода в λ-исчислении

α-преобразование (α-эквивалентность)

Выражение $\lambda.x\,A$ можно преобразовать к виду $\lambda.y\,B$ заменой x на y в A, если y не является свободной в A:

$$\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,B, \text{ если } B = A[x \to y] \text{ и } y \notin FV(A).$$

Определение

Подстановкой $A[x \to y]$ (формальной заменой x на y в выражении A) называется рекурсивное преобразование :

$$x[x \to X] \equiv X$$
$$y[x \to X] \equiv y, \text{ если } y \neq x$$
$$(A\,B)[x \to X] \equiv A[x \to X]\,B[x \to X]$$

Правила вывода в λ-исчислении

α-преобразование (α-эквивалентность)

Выражение $\lambda.x\,A$ можно преобразовать к виду $\lambda.y\,B$ заменой x на y в A, если y не является свободной в A:

$$\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,B, \text{ если } B = A[x \to y] \text{ и } y \notin FV(A).$$

Определение

Подстановкой $A[x \to y]$ (формальной заменой x на y в выражении A) называется рекурсивное преобразование :

$$\begin{aligned}
x[x \to X] &\equiv X \\
y[x \to X] &\equiv y, \text{ если } y \neq x \\
(A\,B)[x \to X] &\equiv A[x \to X]\,B[x \to X] \\
(\lambda.x\,A)[x \to X] &\equiv \lambda.x\,A
\end{aligned}$$

Правила вывода в λ-исчислении

α-преобразование (α-эквивалентность)

Выражение $\lambda.x\,A$ можно преобразовать к виду $\lambda.y\,B$ заменой x на y в A, если y не является свободной в A:

$$\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,B,\ \text{если } B = A[x \to y] \text{ и } y \notin FV(A).$$

Определение

Подстановкой $A[x \to y]$ (формальной заменой x на y в выражении A) называется рекурсивное преобразование :

$$\begin{aligned}
x[x \to X] &\equiv X \\
y[x \to X] &\equiv y,\ \text{если } y \neq x \\
(A\,B)[x \to X] &\equiv A[x \to X]\,B[x \to X] \\
(\lambda.x\,A)[x \to X] &\equiv \lambda.x\,A \\
(\lambda.y\,A)[x \to X] &\equiv \lambda.y\,(A[x \to X]) \\
&\text{если } y \neq x \text{ и } y \notin FV(X)
\end{aligned}$$

Правила вывода в λ-исчислении

α-преобразование (α-эквивалентность)

Выражение $\lambda.x\,A$ можно преобразовать к виду $\lambda.y\,B$ заменой x на y в A, если y не является свободной в A:

$$\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,B,\ \text{если } B = A[x \to y] \text{ и } y \notin FV(A).$$

β-редукция

Вычисление λ-терма заключается в замене формального аргумента фактическим:

$$(\lambda.x\,A)\,B \to_\beta A[x \to B]$$

Определение

Подстановкой $A[x \to y]$ (формальной заменой x на y в выражении A) называется рекурсивное преобразование :

$$\begin{aligned}
x[x \to X] &\equiv X \\
y[x \to X] &\equiv y,\ \text{если } y \neq x \\
(A\,B)[x \to X] &\equiv A[x \to X]\,B[x \to X] \\
(\lambda.x\,A)[x \to X] &\equiv \lambda.x\,A \\
(\lambda.y\,A)[x \to X] &\equiv \lambda.y\,(A[x \to X]) \\
&\text{если } y \neq x \text{ и } y \notin FV(X)
\end{aligned}$$

Правила вывода в λ-исчислении

α-преобразование (α-эквивалентность)

Выражение $\lambda.x\,A$ можно преобразовать к виду $\lambda.y\,B$ заменой x на y в A, если y не является свободной в A:

$$\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,B, \text{ если } B = A[x \to y] \text{ и } y \notin FV(A).$$

β-редукция

Вычисление λ-терма заключается в замене формального аргумента фактическим:

$$(\lambda.x\,A)\,B \to_\beta A[x \to B]$$

η-редукция

Закон функциональной экстенсивности:

$$\lambda.x\,(A\ x) \to_\eta A \text{ если } x \notin FV(A).$$

Определение

Подстановкой $A[x \to y]$ (формальной заменой x на y в выражении A) называется рекурсивное преобразование :

$$\begin{aligned}
x[x \to X] &\equiv X \\
y[x \to X] &\equiv y, \text{ если } y \neq x \\
(A\,B)[x \to X] &\equiv A[x \to X]\,B[x \to X] \\
(\lambda.x\,A)[x \to X] &\equiv \lambda.x\,A \\
(\lambda.y\,A)[x \to X] &\equiv \lambda.y\,(A[x \to X]) \\
&\text{если } y \neq x \text{ и } y \notin FV(X)
\end{aligned}$$

Нормальная форма и редекс

Определение

Редексом (**red**ucible **ex**pression) называется λ-терм, к которому можно применить β- или η-редукцию.

Нормальная форма и редекс

Определение

Редексом (**red**ucible **ex**pression) называется λ-терм, к которому можно применить β- или η-редукцию.

Определение

Говорят, что λ-терм находится в нормальной форме, если к нему нельзя применить никакое правило редукции.

Нормальная форма и редекс

Определение

Редексом (**red**ucible **ex**pression) называется λ-терм, к которому можно применить β- или η-редукцию.

Определение

Говорят, что λ-терм находится в нормальной форме, если к нему нельзя применить никакое правило редукции.

Вычислительный процесс в λ-исчислении

Процесс вычисления λ-выражений состоит в редукции выражения до тех пор пока оно включает в себя хотя бы один редекс.

Нормальная форма и редекс

Определение

Редексом (**red**ucible **ex**pression) называется λ-терм, к которому можно применить β- или η-редукцию.

Определение

Говорят, что λ-терм находится в нормальной форме, если к нему нельзя применить никакое правило редукции.

Вычислительный процесс в λ-исчислении

Процесс вычисления λ-выражений состоит в редукции выражения до тех пор пока оно включает в себя хотя бы один редекс.

Нормальная форма соответствует концу вычислений и результату.

Нормальная форма и редекс

Определение

Редексом (**red**ucible **ex**pression) называется λ-терм, к которому можно применить β- или η-редукцию.

Определение

Говорят, что λ-терм находится в нормальной форме, если к нему нельзя применить никакое правило редукции.

Вычислительный процесс в λ-исчислении

Процесс вычисления λ-выражений состоит в редукции выражения до тех пор пока оно включает в себя хотя бы один редекс.

Нормальная форма соответствует концу вычислений и результату.

Не всякое λ-выражение имеет нормальную форму. Например:

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

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

Примеры применений правил вывода

Примеры α-преобразований:

$$\lambda.x\, x \leftrightarrow_\alpha \lambda.y\, y$$

$$\lambda.x\, (f\, y\, x) \leftrightarrow_\alpha \lambda.z\, (f\, y\, z)$$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\, x \leftrightarrow_\alpha \lambda.y\, y$

$\lambda.x\, (f\, y\, x) \leftrightarrow_\alpha \lambda.z\, (f\, y\, z)$ $\lambda.x\, (f\, y\, x) \not\leftrightarrow_\alpha \lambda.y\, (f\, y\, y)$

$\lambda.x\, \lambda.x\, x \leftrightarrow_\alpha \lambda.x\, \lambda.y\, y$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\, x \leftrightarrow_\alpha \lambda.y\, y$

$\lambda.x\, (f\, y\, x) \leftrightarrow_\alpha \lambda.z\, (f\, y\, z)$ $\lambda.x\, (f\, y\, x) \not\leftrightarrow_\alpha \lambda.y\, (f\, y\, y)$

$\lambda.x\, \lambda.x\, x \leftrightarrow_\alpha \lambda.x\, \lambda.y\, y$ $\lambda.x\, \lambda.x\, x \not\leftrightarrow_\alpha \lambda.y\, \lambda.x\, y$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\, x \leftrightarrow_\alpha \lambda.y\, y$

$\lambda.x\, (f\, y\, x) \leftrightarrow_\alpha \lambda.z\, (f\, y\, z)$ $\lambda.x\, (f\, y\, x) \not\leftrightarrow_\alpha \lambda.y\, (f\, y\, y)$

$\lambda.x\, \lambda.x\, x \leftrightarrow_\alpha \lambda.x\, \lambda.y\, y$ $\lambda.x\, \lambda.x\, x \not\leftrightarrow_\alpha \lambda.y\, \lambda.x\, y$

Примеры β-редукции:

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\quad$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\quad$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \rightarrow_\beta y$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$$(\lambda.x\,x)\,y \rightarrow_\beta y$$

$$(\lambda.x\,(*\ x\ x))\,y \rightarrow_\beta (*\ y\ y)$$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \rightarrow_\beta y$

$(\lambda.x\,(*\ x\ x))\,y \rightarrow_\beta (*\ y\ y)$

$(\lambda.g\,\lambda.x\,(g\ x\ x))\ *$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\, x \leftrightarrow_\alpha \lambda.y\, y$

$\lambda.x\,(f\; y\; x) \leftrightarrow_\alpha \lambda.z\,(f\; y\; z)$ $\quad$ $\lambda.x\,(f\; y\; x) \not\leftrightarrow_\alpha \lambda.y\,(f\; y\; y)$

$\lambda.x\,\lambda.x\, x \leftrightarrow_\alpha \lambda.x\,\lambda.y\, y$ $\quad$ $\lambda.x\,\lambda.x\, x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\, y$

Примеры β-редукции:

$$(\lambda.x\, x)\, y \rightarrow_\beta y$$

$$(\lambda.x\,(*\; x\; x))\, y \rightarrow_\beta (*\; y\; y)$$

$$(\lambda.g\,\lambda.x\,(g\; x\; x))\; *$$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \rightarrow_\beta y$

$(\lambda.x\,(*\ x\ x))\,y \rightarrow_\beta (*\ y\ y)$

$(\lambda.g\,\lambda.x\,(g\ x\ x))\ * \rightarrow_\beta \lambda.x\,(*\ x\ x)$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \to_\beta y$

$(\lambda.x\,(*\ x\ x))\,y \to_\beta (*\ y\ y)$

$(\lambda.g\,\lambda.x\,(g\ x\ x))\ * \to_\beta \lambda.x\,(*\ x\ x)$

$(\lambda.x\,\lambda.g\,(g\ x\ x))\,2$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\quad$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\quad$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \to_\beta y$ $\quad$ $(\lambda.g\,\lambda.x\,(g\;x\;x))\;* \to_\beta \lambda.x\,(*\;x\;x)$

$(\lambda.x\,(*\;x\;x))\,y \to_\beta (*\;y\;y)$ $\quad$ $(\lambda.x\,\lambda.g\,(g\;x\;x))\,2$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\quad$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\quad$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \rightarrow_\beta y$ $\quad$ $(\lambda.g\,\lambda.x\,(g\;x\;x))\;* \rightarrow_\beta \lambda.x\,(*\;x\;x)$

$(\lambda.x\,(*\;x\;x))\,y \rightarrow_\beta (*\;y\;y)$ $\quad$ $(\lambda.x\,\lambda.g\,(g\;x\;x))\,2 \rightarrow_\beta \lambda.g\,(g\;2\;2)$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\qquad$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\qquad$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \rightarrow_\beta y$

$(\lambda.x\,(*\ x\ x))\,y \rightarrow_\beta (*\ y\ y)$

$(\lambda.g\,\lambda.x\,(g\ x\ x))\ * \rightarrow_\beta \lambda.x\,(*\ x\ x)$

$(\lambda.x\,\lambda.g\,(g\ x\ x))\,2 \rightarrow_\beta \lambda.g\,(g\ 2\ 2)$

Аппликативный порядок

$(\lambda.f\,\lambda.x\,(f\ (f\ x)))\,(\lambda.y\,(*\ y\ y))$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\qquad$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\qquad$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$$(\lambda.x\,x)\,y \rightarrow_\beta y$$

$$(\lambda.x\,(*\;x\;x))\,y \rightarrow_\beta (*\;y\;y)$$

Аппликативный порядок

$$(\lambda.f\,\lambda.x\,(f\;(f\;x)))\,(\lambda.y\,(*\;y\;y)) \rightarrow_\beta$$
$$\lambda.x\,((\lambda.y\,(*\;y\;y))\;((\lambda.y\,(*\;y\;y))\;x))$$

$$(\lambda.g\,\lambda.x\,(g\;x\;x))\;* \rightarrow_\beta \lambda.x\,(*\;x\;x)$$

$$(\lambda.x\,\lambda.g\,(g\;x\;x))\,2 \rightarrow_\beta \lambda.g\,(g\;2\;2)$$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\, x \leftrightarrow_\alpha \lambda.y\, y$

$\lambda.x\, (f\, y\, x) \leftrightarrow_\alpha \lambda.z\, (f\, y\, z)$ $\quad$ $\lambda.x\, (f\, y\, x) \not\leftrightarrow_\alpha \lambda.y\, (f\, y\, y)$

$\lambda.x\, \lambda.x\, x \leftrightarrow_\alpha \lambda.x\, \lambda.y\, y$ $\quad$ $\lambda.x\, \lambda.x\, x \not\leftrightarrow_\alpha \lambda.y\, \lambda.x\, y$

Примеры β-редукции:

$$(\lambda.x\, x)\, y \rightarrow_\beta y$$

$$(\lambda.x\, (*\ x\ x))\, y \rightarrow_\beta (*\ y\ y)$$

$$(\lambda.g\, \lambda.x\, (g\ x\ x))\ * \rightarrow_\beta \lambda.x\, (*\ x\ x)$$

$$(\lambda.x\, \lambda.g\, (g\ x\ x))\, 2 \rightarrow_\beta \lambda.g\, (g\ 2\ 2)$$

Аппликативный порядок

$$(\lambda.f\, \lambda.x\, (f\ (f\ x)))\, (\lambda.y\, (*\ y\ y)) \rightarrow_\beta$$

$$\lambda.x\, ((\lambda.y\, (*\ y\ y))\ ((\lambda.y\, (*\ y\ y))\ x)) \rightarrow_\beta$$

$$\lambda.x\, ((\lambda.y\, (*\ y\ y))\ (*\ x\ x))$$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\quad\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\quad\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \to_\beta y$

$(\lambda.x\,(*\ x\ x))\,y \to_\beta (*\ y\ y)$

$(\lambda.g\,\lambda.x\,(g\ x\ x))\ * \to_\beta \lambda.x\,(*\ x\ x)$

$(\lambda.x\,\lambda.g\,(g\ x\ x))\,2 \to_\beta \lambda.g\,(g\ 2\ 2)$

Аппликативный порядок

$(\lambda.f\,\lambda.x\,(f\ (f\ x)))\,(\lambda.y\,(*\ y\ y)) \to_\beta$

$\lambda.x\,((\lambda.y\,(*\ y\ y))\ ((\lambda.y\,(*\ y\ y))\ x)) \to_\beta$

$\lambda.x\,((\lambda.y\,(*\ y\ y))\ (*\ x\ x)) \to_\beta$

$\lambda.x\,(*\ (*\ x\ x)\ (*\ x\ x))$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\, x \leftrightarrow_\alpha \lambda.y\, y$

$\lambda.x\, (f\, y\, x) \leftrightarrow_\alpha \lambda.z\, (f\, y\, z)$ $\quad$ $\lambda.x\, (f\, y\, x) \not\leftrightarrow_\alpha \lambda.y\, (f\, y\, y)$

$\lambda.x\, \lambda.x\, x \leftrightarrow_\alpha \lambda.x\, \lambda.y\, y$ $\quad$ $\lambda.x\, \lambda.x\, x \not\leftrightarrow_\alpha \lambda.y\, \lambda.x\, y$

Примеры β-редукции:

$(\lambda.x\, x)\, y \rightarrow_\beta y$

$(\lambda.x\, (*\ x\ x))\, y \rightarrow_\beta (*\ y\ y)$

$(\lambda.g\, \lambda.x\, (g\ x\ x))\ * \rightarrow_\beta \lambda.x\, (*\ x\ x)$

$(\lambda.x\, \lambda.g\, (g\ x\ x))\, 2 \rightarrow_\beta \lambda.g\, (g\ 2\ 2)$

Аппликативный порядок

$(\lambda.f\, \lambda.x\, (f\ (f\ x)))\, (\lambda.y\, (*\ y\ y)) \rightarrow_\beta$

$\lambda.x\, ((\lambda.y\, (*\ y\ y))\ ((\lambda.y\, (*\ y\ y))\ x)) \rightarrow_\beta$

$\lambda.x\, ((\lambda.y\, (*\ y\ y))\ (*\ x\ x)) \rightarrow_\beta$

$\lambda.x\, (*\ (*\ x\ x)\ (*\ x\ x))$

Нормальный порядок

$(\lambda.f\, \lambda.x\, (f\ (f\ x)))\, (\lambda.y\, (*\ y\ y))$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \to_\beta y$

$(\lambda.x\,(*\ x\ x))\,y \to_\beta (*\ y\ y)$

$(\lambda.g\,\lambda.x\,(g\ x\ x))\ * \to_\beta \lambda.x\,(*\ x\ x)$

$(\lambda.x\,\lambda.g\,(g\ x\ x))\,2 \to_\beta \lambda.g\,(g\ 2\ 2)$

Аппликативный порядок

$(\lambda.f\,\lambda.x\,(f\ (f\ x)))\,(\lambda.y\,(*\ y\ y)) \to_\beta$

$\lambda.x\,((\lambda.y\,(*\ y\ y))\ ((\lambda.y\,(*\ y\ y))\ x)) \to_\beta$

$\lambda.x\,((\lambda.y\,(*\ y\ y))\ (*\ x\ x)) \to_\beta$

$\lambda.x\,(*\ (*\ x\ x)\ (*\ x\ x))$

Нормальный порядок

$(\lambda.f\,\lambda.x\,(f\ (f\ x)))\,(\lambda.y\,(*\ y\ y)) \to_\beta$

$\lambda.x\,((\lambda.y\,(*\ y\ y))\ ((\lambda.y\,(*\ y\ y))\ x))$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$(\lambda.x\,x)\,y \rightarrow_\beta y$

$(\lambda.x\,(*\ x\ x))\,y \rightarrow_\beta (*\ y\ y)$

$(\lambda.g\,\lambda.x\,(g\ x\ x))\ * \rightarrow_\beta \lambda.x\,(*\ x\ x)$

$(\lambda.x\,\lambda.g\,(g\ x\ x))\,2 \rightarrow_\beta \lambda.g\,(g\ 2\ 2)$

Аппликативный порядок

$(\lambda.f\,\lambda.x\,(f\ (f\ x)))\,(\lambda.y\,(*\ y\ y)) \rightarrow_\beta$

$\lambda.x\,((\lambda.y\,(*\ y\ y))\ ((\lambda.y\,(*\ y\ y))\ x)) \rightarrow_\beta$

$\lambda.x\,((\lambda.y\,(*\ y\ y))\ (*\ x\ x)) \rightarrow_\beta$

$\lambda.x\,(*\ (*\ x\ x)\ (*\ x\ x))$

Нормальный порядок

$(\lambda.f\,\lambda.x\,(f\ (f\ x)))\,(\lambda.y\,(*\ y\ y)) \rightarrow_\beta$

$\lambda.x\,((\lambda.y\,(*\ y\ y))\ ((\lambda.y\,(*\ y\ y))\ x)) \rightarrow_\beta$

$\lambda.x\,(*\ ((\lambda.y\,(*\ y\ y))\ x)\ ((\lambda.y\,(*\ y\ y))\ x))$

Примеры применений правил вывода

Примеры α-преобразований:

$\lambda.x\,x \leftrightarrow_\alpha \lambda.y\,y$

$\lambda.x\,(f\,y\,x) \leftrightarrow_\alpha \lambda.z\,(f\,y\,z)$ $\lambda.x\,(f\,y\,x) \not\leftrightarrow_\alpha \lambda.y\,(f\,y\,y)$

$\lambda.x\,\lambda.x\,x \leftrightarrow_\alpha \lambda.x\,\lambda.y\,y$ $\lambda.x\,\lambda.x\,x \not\leftrightarrow_\alpha \lambda.y\,\lambda.x\,y$

Примеры β-редукции:

$$(\lambda.x\,x)\,y \rightarrow_\beta y$$

$$(\lambda.x\,(*\ x\ x))\,y \rightarrow_\beta (*\ y\ y)$$

$$(\lambda.g\,\lambda.x\,(g\ x\ x))\ * \rightarrow_\beta \lambda.x\,(*\ x\ x)$$

$$(\lambda.x\,\lambda.g\,(g\ x\ x))\,2 \rightarrow_\beta \lambda.g\,(g\ 2\ 2)$$

Аппликативный порядок

$$(\lambda.f\,\lambda.x\,(f\ (f\ x)))\,(\lambda.y\,(*\ y\ y)) \rightarrow_\beta$$
$$\lambda.x\,((\lambda.y\,(*\ y\ y))\ ((\lambda.y\,(*\ y\ y))\ x)) \rightarrow_\beta$$
$$\lambda.x\,((\lambda.y\,(*\ y\ y))\ (*\ x\ x)) \rightarrow_\beta$$
$$\lambda.x\,(*\ (*\ x\ x)\ (*\ x\ x))$$

Нормальный порядок

$$(\lambda.f\,\lambda.x\,(f\ (f\ x)))\,(\lambda.y\,(*\ y\ y)) \rightarrow_\beta$$
$$\lambda.x\,((\lambda.y\,(*\ y\ y))\ ((\lambda.y\,(*\ y\ y))\ x)) \rightarrow_\beta$$
$$\lambda.x\,(*\ ((\lambda.y\,(*\ y\ y))\ x)\ ((\lambda.y\,(*\ y\ y))\ x)) \rightarrow_\beta$$
$$\lambda.x\,(*\ (*\ x\ x)\ (*\ x\ x))$$

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

При этом любая функция, вычислимая с помощью машины Тьюринга, может быть вычислена в рамках λ-исчисления и наоборот.

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

При этом любая функция, вычислимая с помощью машины Тьюринга, может быть вычислена в рамках λ-исчисления и наоборот.

Машина Тьюринга:

- Вычислительный процесс — последовательная смена состояния информационной среды: состояния машины и записи на ленте.

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

При этом любая функция, вычислимая с помощью машины Тьюринга, может быть вычислена в рамках λ-исчисления и наоборот.

Машина Тьюринга:

- Вычислительный процесс — последовательная смена состояния информационной среды: состояния машины и записи на ленте.
- Программа — правила смены состояния машины, текущей записи и позиции на ленте.

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

При этом любая функция, вычислимая с помощью машины Тьюринга, может быть вычислена в рамках λ-исчисления и наоборот.

Машина Тьюринга:

- Вычислительный процесс — последовательная смена состояния информационной среды: состояния машины и записи на ленте.
- Программа — правила смены состояния машины, текущей записи и позиции на ленте.
- Данные — исходная запись на ленте.

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

При этом любая функция, вычислимая с помощью машины Тьюринга, может быть вычислена в рамках λ-исчисления и наоборот.

Машина Тьюринга:

- Вычислительный процесс — последовательная смена состояния информационной среды: состояния машины и записи на ленте.
- Программа — правила смены состояния машины, текущей записи и позиции на ленте.
- Данные — исходная запись на ленте.
- Результат — запись на ленте после остановки машины.

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

При этом любая функция, вычислимая с помощью машины Тьюринга, может быть вычислена в рамках λ-исчисления и наоборот.

Машина Тьюринга:

- Вычислительный процесс — последовательная смена состояния информационной среды: состояния машины и записи на ленте.
- Программа — правила смены состояния машины, текущей записи и позиции на ленте.
- Данные — исходная запись на ленте.
- Результат — запись на ленте после остановки машины.

λ-исчисление:

- Вычислительный процесс — последовательность редукций λ-термов.

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

При этом любая функция, вычислимая с помощью машины Тьюринга, может быть вычислена в рамках λ-исчисления и наоборот.

Машина Тьюринга:

- Вычислительный процесс — последовательная смена состояния информационной среды: состояния машины и записи на ленте.
- Программа — правила смены состояния машины, текущей записи и позиции на ленте.
- Данные — исходная запись на ленте.
- Результат — запись на ленте после остановки машины.

λ-исчисление:

- Вычислительный процесс — последовательность редукций λ-термов.
- Программа — исходный λ-терм.

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

При этом любая функция, вычислимая с помощью машины Тьюринга, может быть вычислена в рамках λ-исчисления и наоборот.

Машина Тьюринга:

- Вычислительный процесс — последовательная смена состояния информационной среды: состояния машины и записи на ленте.
- Программа — правила смены состояния машины, текущей записи и позиции на ленте.
- Данные — исходная запись на ленте.
- Результат — запись на ленте после остановки машины.

λ-исчисление:

- Вычислительный процесс — последовательность редукций λ-термов.
- Программа — исходный λ-терм.
- Данные — фактические аргументы исходного λ-терма (так же λ-термы).

Тезис Чёрча–Тьюринга

Множество вычислимых функций и функций, вычислимых с помощью машины Тьюринга или λ-исчисления совпадают.

При этом любая функция, вычислимая с помощью машины Тьюринга, может быть вычислена в рамках λ-исчисления и наоборот.

Машина Тьюринга:

- Вычислительный процесс — последовательная смена состояния информационной среды: состояния машины и записи на ленте.
- Программа — правила смены состояния машины, текущей записи и позиции на ленте.
- Данные — исходная запись на ленте.
- Результат — запись на ленте после остановки машины.

λ-исчисление:

- Вычислительный процесс — последовательность редукций λ-термов.
- Программа — исходный λ-терм.
- Данные — фактические аргументы исходного λ-терма (так же λ-термы).
- Результат — нормальная форма аппликации исходного λ-терма и данных.

λ-исчисление, как язык программирования

Базис минимального Lisp-а (Scheme)

λ-исчисление, как язык программирования

Базис минимального LISP-а (SCHEME)

- символы: произвольные идентификаторы;

λ-исчисление, как язык программирования

Базис минимального LISP-а (SCHEME)

- символы: произвольные идентификаторы;
- оператор связывания: **`define`**;

λ-исчисление, как язык программирования

Базис минимального LISP-а (SCHEME)

- символы: произвольные идентификаторы;
- оператор связывания: **`define`**;
- оператор абстракции: $\boldsymbol{\lambda}$;

λ-исчисление, как язык программирования

Базис минимального Lisp-а (Scheme)

- символы: произвольные идентификаторы;
- оператор связывания: **`define`**;
- оператор абстракции: $\boldsymbol{\lambda}$;
- логические данные и операции: `#t`, `#f`, **`if`**, **`or`**, **`and`**, `not`, `eq?`;

λ-исчисление, как язык программирования

Базис минимального LISP-а (SCHEME)

- символы: произвольные идентификаторы;
- оператор связывания: **`define`**;
- оператор абстракции: $\boldsymbol{\lambda}$;
- логические данные и операции: `#t`, `#f`, **`if`**, **`or`**, **`and`**, `not`, `eq?`;
- точечные пары и списки: `cons`, `car`, `cdr`, `null`, `null?`, `pair?`;

λ-исчисление, как язык программирования

Базис минимального LISP-а (SCHEME)

- символы: произвольные идентификаторы;
- оператор связывания: **`define`**;
- оператор абстракции: **λ**;
- логические данные и операции: `#t`, `#f`, **`if`**, **`or`**, **`and`**, `not`, `eq?`;
- точечные пары и списки: `cons`, `car`, `cdr`, `null`, `null?`, `pair?`;
- числовые данные и операции: `+ - * / expt = <`, `number?`.

λ-исчисление, как язык программирования

Базис минимального LISP-а (SCHEME)

- символы: произвольные идентификаторы;
- оператор связывания: **define**;
- оператор абстракции: **λ**;
- логические данные и операции: #t, #f, **if**, **or**, **and**, not, eq?;
- точечные пары и списки: cons, car, cdr, null, null?, pair?;
- числовые данные и операции: + - * / expt = <, number?.

Базис чистого λ-исчисления

λ-исчисление, как язык программирования

Базис минимального LISP-а (SCHEME)

- символы: произвольные идентификаторы;
- оператор связывания: **`define`**;
- оператор абстракции: $\boldsymbol{\lambda}$;
- логические данные и операции: `#t`, `#f`, **`if`**, **`or`**, **`and`**, `not`, `eq?`;
- точечные пары и списки: `cons`, `car`, `cdr`, `null`, `null?`, `pair?`;
- числовые данные и операции: `+ - * / expt = <`, `number?`.

Базис чистого λ-исчисления

- символы: произвольные идентификаторы;

λ-исчисление, как язык программирования

Базис минимального LISP-a (SCHEME)

- символы: произвольные идентификаторы;
- оператор связывания: **define**;
- оператор абстракции: **λ**;
- логические данные и операции: `#t`, `#f`, **if**, **or**, **and**, `not`, `eq?`;
- точечные пары и списки: `cons`, `car`, `cdr`, `null`, `null?`, `pair?`;
- числовые данные и операции: `+ - * / expt = <`, `number?`.

Базис чистого λ-исчисления

- символы: произвольные идентификаторы;
- оператор абстракции: λ;

λ-исчисление, как язык программирования

Базис минимального Lisp-а (Scheme)

- символы: произвольные идентификаторы;
- оператор связывания: **`define`**;
- оператор абстракции: $\boldsymbol{\lambda}$;
- логические данные и операции: `#t`, `#f`, **`if`**, **`or`**, **`and`**, `not`, `eq?`;
- точечные пары и списки: `cons`, `car`, `cdr`, `null`, `null?`, `pair?`;
- числовые данные и операции: `+ - * / expt = <`, `number?`.

Базис чистого λ-исчисления

- символы: произвольные идентификаторы;
- оператор абстракции: λ;
- оператор связывания: $=$;

λ-исчисление, как язык программирования

Базис минимального Lisp-а (Scheme)

- символы: произвольные идентификаторы;
- оператор связывания: **`define`**;
- оператор абстракции: $\boldsymbol{\lambda}$;
- логические данные и операции: `#t`, `#f`, **`if`**, **`or`**, **`and`**, `not`, `eq?`;
- точечные пары и списки: `cons`, `car`, `cdr`, `null`, `null?`, `pair?`;
- числовые данные и операции: `+ - * / expt = <`, `number?`.

Базис чистого λ-исчисления

- символы: произвольные идентификаторы;
- оператор абстракции: λ;
- оператор связывания: $=$;

В базисе чистого бестипового λ-исчисления можно определить базис минимального Lisp-а без квантификаторов типа (`pair?`, `number?`) и предиката `eq?`.

λ-исчисление, как язык программирования

Базис минимального LISP-а (SCHEME)

- символы: произвольные идентификаторы;
- оператор связывания: **define**;
- оператор абстракции: $\boldsymbol{\lambda}$;
- логические данные и операции: `#t`, `#f`, **if**, **or**, **and**, `not`, `eq?`;
- точечные пары и списки: `cons`, `car`, `cdr`, `null`, `null?`, `pair?`;
- числовые данные и операции: `+ - * / expt = <`, `number?`.

Базис чистого λ-исчисления

- символы: произвольные идентификаторы;
- оператор абстракции: λ;
- оператор связывания: $=$;

В базисе чистого бестипового λ-исчисления можно определить базис минимального LISP-а без квантификаторов типа (`pair?`, `number?`) и предиката `eq?`.

Отдых и созерцание

$$\lambda.x\,x$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$
$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$
$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$
$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$

$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$

$$\mathbf{T} = \lambda.qr\,q$$

$$\mathbf{F} = \lambda.qr\,r$$

$$\mathbf{IF}\ \mathbf{T}\ A\ B \quad = \quad (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$

$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$

$$\mathbf{T} = \lambda.qr\,q$$

$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned} \mathbf{IF}\ \mathbf{T}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B \\ &\to_\beta p\ q\ r \end{aligned}$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$
$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$
$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned}\mathbf{IF}\ \mathbf{T}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B \\ &\to_\beta (\lambda.qr\,q)\ q\ r\end{aligned}$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$

$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$

$$\mathbf{T} = \lambda.qr\,q$$

$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned} \mathbf{IF}\ \mathbf{T}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B \\ &\rightarrow_\beta (\lambda.qr\,q)\ A\ r \end{aligned}$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$
$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$
$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned}\mathbf{IF}\ \mathbf{T}\ A\ B \quad &= \quad (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B \\ &\to_\beta \quad (\lambda.qr\,q)\ A\ B\end{aligned}$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$

$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$

$$\mathbf{T} = \lambda.qr\,q$$

$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned}\mathbf{IF}\ \mathbf{T}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B\\ &\to_\beta (\lambda.qr\,q)\ A\ B\\ &\to_\beta q\end{aligned}$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$
$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$
$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned} \mathbf{IF}\ \mathbf{T}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B \\ &\to_\beta (\lambda.qr\,q)\ A\ B \\ &\to_\beta A \end{aligned}$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$
$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$
$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned}\mathbf{IF}\ \mathbf{T}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B\\ &\to_\beta (\lambda.qr\,q)\ A\ B\\ &\to_\beta A\end{aligned}$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$

$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$

$$\mathbf{T} = \lambda.qr\,q$$

$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned}
\mathbf{IF}\ \mathbf{T}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B \\
&\to_\beta (\lambda.qr\,q)\ A\ B \\
&\to_\beta A
\end{aligned}$$

$$\mathbf{IF}\ \mathbf{F}\ A\ B = (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,r)\ A\ B$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$
$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$
$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned}\mathbf{IF}\ \mathbf{T}\ A\ B &= (\lambda.pqr\,{\color{red}(p\ q\ r)})\ (\lambda.qr\,q)\ A\ B\\ &\to_\beta (\lambda.qr\,q)\ A\ B\\ &\to_\beta A\end{aligned}$$

$$\begin{aligned}\mathbf{IF}\ \mathbf{F}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,r)\ A\ B\\ &\to_\beta (\lambda.qr\,r)\ A\ B\end{aligned}$$

Логические данные и операции

Формальные уравнения

$$\mathbf{IF}\ \mathbf{T}\ q\ r = q$$
$$\mathbf{IF}\ \mathbf{F}\ q\ r = r$$

Ленивая реализация

$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$
$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

$$\begin{aligned}\mathbf{IF}\ \mathbf{T}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,q)\ A\ B\\ &\to_\beta (\lambda.qr\,q)\ A\ B\\ &\to_\beta A\end{aligned}$$

$$\begin{aligned}\mathbf{IF}\ \mathbf{F}\ A\ B &= (\lambda.pqr\,(p\ q\ r))\ (\lambda.qr\,r)\ A\ B\\ &\to_\beta (\lambda.qr\,r)\ A\ B\\ &\to_\beta B\end{aligned}$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\, q$$

$$\mathbf{F} = \lambda.qr\, r$$

Логические операции

$$\mathbf{AND} = \lambda.qr\, (q\ r\ q)$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\, q$$
$$\mathbf{F} = \lambda.qr\, r$$

Логические операции

$$\mathbf{AND} = \lambda.qr\, (q\ r\ q)$$

$$\mathbf{AND\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{AND\ F\ T} = \mathbf{F\ T\ F} = \mathbf{F}$$
$$\mathbf{AND\ T\ F} = \mathbf{T\ F\ T} = \mathbf{F}$$
$$\mathbf{AND\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\, q$$
$$\mathbf{F} = \lambda.qr\, r$$

Логические операции

$$\mathbf{AND} = \lambda.qr\, (q\ r\ q)$$
$$\mathbf{OR} = \lambda.qr\, (q\ q\ r)$$

$$\mathbf{AND\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{AND\ F\ T} = \mathbf{F\ T\ F} = \mathbf{F}$$
$$\mathbf{AND\ T\ F} = \mathbf{T\ F\ T} = \mathbf{F}$$
$$\mathbf{AND\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\, q$$
$$\mathbf{F} = \lambda.qr\, r$$

Логические операции

$$\mathbf{AND} = \lambda.qr\, (q\ \ r\ \ q)$$
$$\mathbf{OR} = \lambda.qr\, (q\ \ q\ \ r)$$

$$\mathbf{AND\ T\ T = T\ T\ T = T}$$
$$\mathbf{AND\ F\ T = F\ T\ F = F}$$
$$\mathbf{AND\ T\ F = T\ F\ T = F}$$
$$\mathbf{AND\ F\ F = F\ F\ F = F}$$

$$\mathbf{OR\ T\ T = T\ T\ T = T}$$
$$\mathbf{OR\ F\ T = F\ F\ T = T}$$
$$\mathbf{OR\ T\ F = T\ T\ F = T}$$
$$\mathbf{OR\ F\ F = F\ F\ F = F}$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

Логические операции

$$\mathbf{AND} = \lambda.qr\,(q\ r\ q)$$
$$\mathbf{OR} = \lambda.qr\,(q\ q\ r)$$
$$\mathbf{NOT} = \lambda.q\,(q\ \mathbf{F}\ \mathbf{T})$$

$$\mathbf{AND\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{AND\ F\ T} = \mathbf{F\ T\ F} = \mathbf{F}$$
$$\mathbf{AND\ T\ F} = \mathbf{T\ F\ T} = \mathbf{F}$$
$$\mathbf{AND\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

$$\mathbf{OR\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{OR\ F\ T} = \mathbf{F\ F\ T} = \mathbf{T}$$
$$\mathbf{OR\ T\ F} = \mathbf{T\ T\ F} = \mathbf{T}$$
$$\mathbf{OR\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

Логические операции

$$\mathbf{AND} = \lambda.qr\,(q\ \ r\ \ q)$$
$$\mathbf{OR} = \lambda.qr\,(q\ \ q\ \ r)$$
$$\mathbf{NOT} = \lambda.q\,(q\ \mathbf{F}\ \mathbf{T})$$

$$\mathbf{AND\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{AND\ F\ T} = \mathbf{F\ T\ F} = \mathbf{F}$$
$$\mathbf{AND\ T\ F} = \mathbf{T\ F\ T} = \mathbf{F}$$
$$\mathbf{AND\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

$$\mathbf{OR\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{OR\ F\ T} = \mathbf{F\ F\ T} = \mathbf{T}$$
$$\mathbf{OR\ T\ F} = \mathbf{T\ T\ F} = \mathbf{T}$$
$$\mathbf{OR\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

$$\mathbf{NOT\ T} = \mathbf{T\ F\ T} = \mathbf{F}$$
$$\mathbf{NOT\ F} = \mathbf{F\ F\ T} = \mathbf{T}$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\,q$$
$$\mathbf{F} = \lambda.qr\,r$$

Логические операции

$$\mathbf{AND} = \lambda.qr\,(q\ r\ q)$$
$$\mathbf{OR} = \lambda.qr\,(q\ q\ r)$$
$$\mathbf{NOT} = \lambda.q\,(q\ \mathbf{F}\ \mathbf{T})$$
$$\mathbf{IMP} = \lambda.qr\,(r\ r\ (\mathbf{NOT}\ q))$$

$$\mathbf{AND\ T\ T = T\ T\ T = T}$$
$$\mathbf{AND\ F\ T = F\ T\ F = F}$$
$$\mathbf{AND\ T\ F = T\ F\ T = F}$$
$$\mathbf{AND\ F\ F = F\ F\ F = F}$$

$$\mathbf{OR\ T\ T = T\ T\ T = T}$$
$$\mathbf{OR\ F\ T = F\ F\ T = T}$$
$$\mathbf{OR\ T\ F = T\ T\ F = T}$$
$$\mathbf{OR\ F\ F = F\ F\ F = F}$$

$$\mathbf{NOT\ T = T\ F\ T = F}$$
$$\mathbf{NOT\ F = F\ F\ T = T}$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\, q$$
$$\mathbf{F} = \lambda.qr\, r$$

Логические операции

$$\mathbf{AND} = \lambda.qr\,(q\ r\ q)$$
$$\mathbf{OR} = \lambda.qr\,(q\ q\ r)$$
$$\mathbf{NOT} = \lambda.q\,(q\ \mathbf{F}\ \mathbf{T})$$
$$\mathbf{IMP} = \lambda.qr\,(r\ r\ (\mathbf{NOT}\ q))$$
$$\mathbf{XOR} = \lambda.qr\,(q\ (\mathbf{NOT}\ r)\ r)$$

$$\mathbf{AND\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{AND\ F\ T} = \mathbf{F\ T\ F} = \mathbf{F}$$
$$\mathbf{AND\ T\ F} = \mathbf{T\ F\ T} = \mathbf{F}$$
$$\mathbf{AND\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

$$\mathbf{OR\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{OR\ F\ T} = \mathbf{F\ F\ T} = \mathbf{T}$$
$$\mathbf{OR\ T\ F} = \mathbf{T\ T\ F} = \mathbf{T}$$
$$\mathbf{OR\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

$$\mathbf{NOT\ T} = \mathbf{T\ F\ T} = \mathbf{F}$$
$$\mathbf{NOT\ F} = \mathbf{F\ F\ T} = \mathbf{T}$$

Логические данные и операции

$$\mathbf{T} = \lambda.qr\, q$$
$$\mathbf{F} = \lambda.qr\, r$$

Логические операции

$$\mathbf{AND} = \lambda.qr\,(q\ r\ q)$$
$$\mathbf{OR} = \lambda.qr\,(q\ q\ r)$$
$$\mathbf{NOT} = \lambda.q\,(q\ \mathbf{F}\ \mathbf{T})$$
$$\mathbf{IMP} = \lambda.qr\,(r\ r\ (\mathbf{NOT}\ q))$$
$$\mathbf{XOR} = \lambda.qr\,(q\ (\mathbf{NOT}\ r)\ r)$$

Оператор выбора

```
(cond                (OR
 [p1  q1]             (AND p1  q1)
 [p2  q2])            (AND p2  q2))
```

$$\mathbf{AND\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{AND\ F\ T} = \mathbf{F\ T\ F} = \mathbf{F}$$
$$\mathbf{AND\ T\ F} = \mathbf{T\ F\ T} = \mathbf{F}$$
$$\mathbf{AND\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

$$\mathbf{OR\ T\ T} = \mathbf{T\ T\ T} = \mathbf{T}$$
$$\mathbf{OR\ F\ T} = \mathbf{F\ F\ T} = \mathbf{T}$$
$$\mathbf{OR\ T\ F} = \mathbf{T\ T\ F} = \mathbf{T}$$
$$\mathbf{OR\ F\ F} = \mathbf{F\ F\ F} = \mathbf{F}$$

$$\mathbf{NOT\ T} = \mathbf{T\ F\ T} = \mathbf{F}$$
$$\mathbf{NOT\ F} = \mathbf{F\ F\ T} = \mathbf{T}$$

Точечная пара

Формальные уравнения

$$\mathbf{CONS}\ (\mathbf{CAR}\ p)\ (\mathbf{CDR}\ p) = p$$
$$(\mathbf{CAR}\ (\mathbf{CONS}\ a\ b)) = a$$
$$(\mathbf{CDR}\ (\mathbf{CONS}\ a\ b)) = b$$

Точечная пара

Формальные уравнения

$$\mathbf{CONS}\ (\mathbf{CAR}\ p)\ (\mathbf{CDR}\ p) = p$$
$$(\mathbf{CAR}\ (\mathbf{CONS}\ a\ b)) = a$$
$$(\mathbf{CDR}\ (\mathbf{CONS}\ a\ b)) = b$$

$$\mathbf{CONS} = \lambda.ab\,(\lambda.m\,(m\ a\ b))$$
$$\mathbf{CAR} = \lambda.p\,(p\ (\lambda.xy\,x))$$
$$\mathbf{CDR} = \lambda.p\,(p\ (\lambda.xy\,y))$$

Точечная пара

Формальные уравнения

$$\mathbf{CONS}\ (\mathbf{CAR}\ p)\ (\mathbf{CDR}\ p) = p$$
$$(\mathbf{CAR}\ (\mathbf{CONS}\ a\ b)) = a$$
$$(\mathbf{CDR}\ (\mathbf{CONS}\ a\ b)) = b$$

$$\mathbf{CONS} = \lambda.ab\,(\lambda.m\,(m\ a\ b))$$
$$\mathbf{CAR} = \lambda.p\,(p\ (\lambda.xy\,x))$$
$$\mathbf{CDR} = \lambda.p\,(p\ (\lambda.xy\,y))$$

$$\mathbf{CONS}\ A\ B = \lambda.m\,(m\ A\ B)$$

Точечная пара

Формальные уравнения

$$\begin{aligned} \mathbf{CONS}\ (\mathbf{CAR}\ p)\ (\mathbf{CDR}\ p) &= p \\ (\mathbf{CAR}\ (\mathbf{CONS}\ a\ b)) &= a \\ (\mathbf{CDR}\ (\mathbf{CONS}\ a\ b)) &= b \end{aligned}$$

$$\begin{aligned} \mathbf{CONS} &= \lambda.ab\,(\lambda.m\,(m\ a\ b)) \\ \mathbf{CAR} &= \lambda.p\,(p\ (\lambda.xy\,x)) \\ \mathbf{CDR} &= \lambda.p\,(p\ (\lambda.xy\,y)) \end{aligned}$$

$$\mathbf{CONS}\ A\ B = \lambda.m\,(m\ A\ B)$$

$$\begin{aligned} \mathbf{CAR}\ (\mathbf{CONS}\ A\ B) &= (\lambda.p\,(p\ (\lambda.xy\,x)))(\lambda.m\,(m\ A\ B) \\ &\to_\beta (\lambda.m\,(m\ A\ B))\ (\lambda.xy\,x) \\ &\to_\beta (\lambda.xy\,x)\ A\ B \\ &\to_\beta A \end{aligned}$$

Точечная пара

Формальные уравнения

$$\begin{aligned}\mathbf{CONS}\ (\mathbf{CAR}\ p)\ (\mathbf{CDR}\ p) &= p\\ (\mathbf{CAR}\ (\mathbf{CONS}\ a\ b)) &= a\\ (\mathbf{CDR}\ (\mathbf{CONS}\ a\ b)) &= b\end{aligned}$$

$$\begin{aligned}\mathbf{CONS} &= \lambda.ab\,(\lambda.m\,(m\ a\ b))\\ \mathbf{CAR} &= \lambda.p\,(p\ (\lambda.xy\,x))\\ \mathbf{CDR} &= \lambda.p\,(p\ (\lambda.xy\,y))\end{aligned}$$

$$\mathbf{CONS}\ A\ B = \lambda.m\,(m\ A\ B)$$

$$\begin{aligned}\mathbf{CAR}\ (\mathbf{CONS}\ A\ B) \quad &= \quad (\lambda.p\,(p\ (\lambda.xy\,x)))(\lambda.m\,(m\ A\ B))\\ &\to_\beta \quad (\lambda.m\,(m\ A\ B))\ (\lambda.xy\,x)\\ &\to_\beta \quad (\lambda.xy\,x)\ A\ B\\ &\to_\beta \quad A\end{aligned}$$

$$\mathbf{CONS}\ A\ (\mathbf{CONS}\ B\ C) = \lambda.m\,(m\ A\ (\lambda.m'\,(m'\ B\ C)))$$

Точечная пара

Формальные уравнения

$$\begin{aligned} \mathbf{CONS}\ (\mathbf{CAR}\ p)\ (\mathbf{CDR}\ p) &= p \\ (\mathbf{CAR}\ (\mathbf{CONS}\ a\ b)) &= a \\ (\mathbf{CDR}\ (\mathbf{CONS}\ a\ b)) &= b \end{aligned}$$

$$\begin{aligned} \mathbf{CONS} &= \lambda.ab\,(\lambda.m\,(m\ a\ b)) \\ \mathbf{CAR} &= \lambda.p\,(p\ (\lambda.xy\,x)) \\ \mathbf{CDR} &= \lambda.p\,(p\ (\lambda.xy\,y)) \end{aligned}$$

$$\mathbf{CONS}\ A\ B = \lambda.m\,(m\ A\ B)$$

$$\begin{aligned} \mathbf{CAR}\ (\mathbf{CONS}\ A\ B) &= (\lambda.p\,(p\ (\lambda.xy\,x)))(\lambda.m\,(m\ A\ B) \\ &\to_\beta (\lambda.m\,(m\ A\ B))\ (\lambda.xy\,x) \\ &\to_\beta (\lambda.xy\,x)\ A\ B \\ &\to_\beta A \end{aligned}$$

$$\mathbf{CONS}\ A\ (\mathbf{CONS}\ B\ C) = \lambda.m\,(m\ A\ (\lambda.m'\,(m'\ B\ C)))$$

$$\begin{aligned} &\mathbf{CDR}\ (\mathbf{CONS}\ A\ (\mathbf{CONS}\ B\ C)) = \\ = \ &(\lambda.p\,(p\ (\lambda.xy\,y)))(\lambda.m\,(m\ A\ (\lambda.m'\,(m'\ B\ C)))) \\ \to_\beta\ &(\lambda.m\,(m\ A\ (\lambda.m'\,(m'\ B\ C))))\ (\lambda.xy\,y) \\ \to_\beta\ &(\lambda.xy\,y)\ A\ (\lambda.m'\,(m'\ B\ C)) \\ \to_\beta\ &\lambda.m'\,(m'\ B\ C) \end{aligned}$$

Точечная пара

Формальные уравнения

$$\mathbf{NULL}?\ (\mathbf{CONS}\ a\ b)) = \mathbf{F}$$
$$\mathbf{NULL}\ \mathbf{NULL} = \mathbf{T}$$

Точечная пара

Формальные уравнения

$$\mathbf{NULL?}\ (\mathbf{CONS}\ a\ b)) = \mathbf{F}$$
$$\mathbf{NULL}\ \mathbf{NULL} = \mathbf{T}$$

Реализация

$$\mathbf{NULL?} = \lambda.p\,(p\ (\lambda.xy\,\mathbf{F}))$$
$$\mathbf{NULL} = \lambda.p\,\mathbf{T}$$

Точечная пара

Формальные уравнения

$$\begin{aligned} \mathbf{NULL?}\ (\mathbf{CONS}\ a\ b)) &= \mathbf{F} \\ \mathbf{NULL}\ \mathbf{NULL} &= \mathbf{T} \end{aligned}$$

Реализация

$$\begin{aligned} \mathbf{NULL?} &= \lambda.p\,(p\ (\lambda.xy\,\mathbf{F})) \\ \mathbf{NULL} &= \lambda.p\,\mathbf{T} \end{aligned}$$

$$\begin{aligned} \mathbf{NULL?}\ (\mathbf{CONS}\ B\ C) \quad &= \quad (\lambda.p\,(p\ (\lambda.xy\,\mathbf{F})))\ (\lambda.m\,(m\ B\ C)) \\ &\to_\beta \quad (\lambda.m\,(m\ B\ C))\ (\lambda.xy\,\mathbf{F}) \\ &\to_\beta \quad (\lambda.xy\,\mathbf{F})\ B\ C \\ &\to_\beta \quad \mathbf{F} \end{aligned}$$

Точечная пара

Формальные уравнения

$$\begin{aligned} \mathbf{NULL?}\ (\mathbf{CONS}\ a\ b)) &= \mathbf{F} \\ \mathbf{NULL}\ \mathbf{NULL} &= \mathbf{T} \end{aligned}$$

Реализация

$$\begin{aligned} \mathbf{NULL?} &= \lambda.p\,(p\ (\lambda.xy\,\mathbf{F})) \\ \mathbf{NULL} &= \lambda.p\,\mathbf{T} \end{aligned}$$

$$\begin{aligned} \mathbf{NULL?}\ (\mathbf{CONS}\ B\ C) &= (\lambda.p\,(p\ (\lambda.xy\,\mathbf{F})))\ (\lambda.m\,(m\ B\ C)) \\ &\to_\beta (\lambda.m\,(m\ B\ C))\ (\lambda.xy\,\mathbf{F}) \\ &\to_\beta (\lambda.xy\,\mathbf{F})\ B\ C \\ &\to_\beta \mathbf{F} \end{aligned}$$

$$\begin{aligned} \mathbf{NULL?}\ \mathbf{NULL} &= (\lambda.p\,(p\ (\lambda.xy\,\mathbf{F})))\ (\lambda.p\,\mathbf{T}) \\ &\to_\beta (\lambda.p\,\mathbf{T})\ (\lambda.xy\,\mathbf{F}) \\ &\to_\beta \mathbf{T} \end{aligned}$$

Представление натуральных чисел

Натуральными числами что-нибудь считают.

Формальные уравнения

0:

1:

2:

3:

$\vdots$

N: $\underbrace{\ldots}_{N}$

0:

1: $f\ x$

2: $f\ (f\ x)$

3: $f\ (f\ (f\ x))$

$\vdots$

N: $\underbrace{f\ (f \ldots (f\ x))}_{N}$

Возможная реализация:

$$\mathcal{N}_{\mathbf{0}} = \lambda.f\,\lambda.x\,x$$
$$\mathcal{N}_{\mathbf{1}} = \lambda.f\,\lambda.x\,(f\,x)$$
$$\mathcal{N}_{\mathbf{2}} = \lambda.f\,\lambda.x\,(f\,(f\,x))$$
$$\mathcal{N}_{\mathbf{3}} = \lambda.f\,\lambda.x\,(f\,(f\,(f\,x)))$$

$\vdots$

Представление натуральных чисел

Натуральные числа образуют индуктивное множество.

Формальные уравнения

$$\mathcal{N} = 0 \mid \mathcal{N} + 1$$

Представление натуральных чисел

Натуральные числа образуют индуктивное множество.

Формальные уравнения

$$\mathcal{N} = 0 \mid \mathcal{N} + 1$$

$$\begin{aligned}
0 &= 0 \\
1 &= 0 + 1 \\
2 &= 1 + 1 \\
3 &= 2 + 1 \\
&\vdots \\
N &= (N-1) + 1
\end{aligned}$$

Представление натуральных чисел

Натуральные числа образуют индуктивное множество.

Формальные уравнения

$$\mathcal{N} = 0 \mid \mathcal{N} + 1$$

$$\begin{aligned}
0 &= 0 & \mathcal{N}_\mathbf{0} &= \lambda.f\,\lambda.x\,x \\
1 &= 0 + 1 & \mathcal{N}_\mathbf{1} &= \mathcal{S}\,\mathcal{N}_\mathbf{0} \\
2 &= 1 + 1 & \mathcal{N}_\mathbf{2} &= \mathcal{S}\,\mathcal{N}_\mathbf{1} \\
3 &= 2 + 1 & \mathcal{N}_\mathbf{3} &= \mathcal{S}\,\mathcal{N}_\mathbf{2} \\
&\vdots & &\vdots \\
N &= (N-1) + 1 & \mathcal{N}_\mathbf{n} &= \mathcal{S}\,\mathcal{N}_\mathbf{n-1}
\end{aligned}$$

Представление натуральных чисел

Натуральные числа образуют индуктивное множество.

Формальные уравнения

$$\mathcal{N} = 0 \mid \mathcal{N} + 1$$

$$\begin{aligned}
0 &= 0 & \mathcal{N}_{\mathbf{0}} &= \lambda.f\,\lambda.x\,x \\
1 &= 0+1 & \mathcal{N}_{\mathbf{1}} &= \mathcal{S}\;\mathcal{N}_{\mathbf{0}} \\
2 &= 1+1 & \mathcal{N}_{\mathbf{2}} &= \mathcal{S}\;\mathcal{N}_{\mathbf{1}} \\
3 &= 2+1 & \mathcal{N}_{\mathbf{3}} &= \mathcal{S}\;\mathcal{N}_{\mathbf{2}} \\
&\vdots & &\vdots \\
N &= (N-1)+1 & \mathcal{N}_{\mathbf{n}} &= \mathcal{S}\;\mathcal{N}_{\mathbf{n-1}}
\end{aligned}$$

Функция следования (инкремент)

$$\mathcal{S} = \lambda.n\,\lambda.f\,\lambda.x\,((n\;\;f)\;\;(f\;\;x))$$

Представление натуральных чисел

Натуральные числа образуют индуктивное множество.

Формальные уравнения

$$\mathcal{N} = 0 \mid \mathcal{N} + 1$$

$$\begin{aligned}
0 &= 0 & \mathcal{N}_\mathbf{0} &= \lambda.f\,\lambda.x\,x \\
1 &= 0 + 1 & \mathcal{N}_\mathbf{1} &= \mathcal{S}\ \mathcal{N}_\mathbf{0} \\
2 &= 1 + 1 & \mathcal{N}_\mathbf{2} &= \mathcal{S}\ \mathcal{N}_\mathbf{1} \\
3 &= 2 + 1 & \mathcal{N}_\mathbf{3} &= \mathcal{S}\ \mathcal{N}_\mathbf{2} \\
&\vdots & &\vdots \\
N &= (N-1) + 1 & \mathcal{N}_\mathbf{n} &= \mathcal{S}\ \mathcal{N}_\mathbf{n-1}
\end{aligned}$$

Функция следования (инкремент)

$$\mathcal{S} = \lambda.n\,\lambda.f\,\lambda.x\,((n\ \ f)\ \ (f\ \ x))$$

$$\mathcal{S}\ \mathcal{N}_\mathbf{n} = \lambda.f\,\lambda.x\,((\mathcal{N}_n\ \ f)\ \ (f\ \ x))$$

Представление натуральных чисел

Натуральные числа образуют индуктивное множество.

Формальные уравнения

$$\mathcal{N} = 0 \mid \mathcal{N} + 1$$

$$\begin{array}{ll}
0 = 0 & \mathcal{N}_\mathbf{0} = \lambda.f\,\lambda.x\,x \\
1 = 0 + 1 & \mathcal{N}_\mathbf{1} = \mathcal{S}\ \mathcal{N}_\mathbf{0} \\
2 = 1 + 1 & \mathcal{N}_\mathbf{2} = \mathcal{S}\ \mathcal{N}_\mathbf{1} \\
3 = 2 + 1 & \mathcal{N}_\mathbf{3} = \mathcal{S}\ \mathcal{N}_\mathbf{2} \\
\vdots & \vdots \\
N = (N-1) + 1 & \mathcal{N}_\mathbf{n} = \mathcal{S}\ \mathcal{N}_\mathbf{n-1}
\end{array}$$

Функция следования (инкремент)

$$\mathcal{S} = \lambda.n\,\lambda.f\,\lambda.x\,((n\ f)\ (f\ x))$$

$$\mathcal{S}\ \mathcal{N}_\mathbf{n} = \lambda.f\,\lambda.x\,((\mathcal{N}_n\ f)\ (f\ x))$$

$$\mathcal{N}_n\ f = \lambda.x'\,(\underbrace{f\ (f\ \ldots (f}_{n}\ x')))$$

Представление натуральных чисел

Натуральные числа образуют индуктивное множество.

Формальные уравнения

$$\mathcal{N} = 0 \mid \mathcal{N} + 1$$

$$\begin{aligned}
0 &= 0 & \mathcal{N}_\mathbf{0} &= \lambda.f\,\lambda.x\,x \\
1 &= 0 + 1 & \mathcal{N}_\mathbf{1} &= \mathcal{S}\ \mathcal{N}_\mathbf{0} \\
2 &= 1 + 1 & \mathcal{N}_\mathbf{2} &= \mathcal{S}\ \mathcal{N}_\mathbf{1} \\
3 &= 2 + 1 & \mathcal{N}_\mathbf{3} &= \mathcal{S}\ \mathcal{N}_\mathbf{2} \\
&\vdots & &\vdots \\
N &= (N-1) + 1 & \mathcal{N}_\mathbf{n} &= \mathcal{S}\ \mathcal{N}_\mathbf{n-1}
\end{aligned}$$

Функция следования (инкремент)

$$\mathcal{S} = \lambda.n\,\lambda.f\,\lambda.x\,((n\ f)\ (f\ x))$$

$$\mathcal{S}\ \mathcal{N}_\mathbf{n} = \lambda.f\,\lambda.x\,((\mathcal{N}_n\ f)\ (f\ x))$$

$$\mathcal{N}_n\ f = \lambda.x'\,(\underbrace{f\ (f\ \ldots (f}_{n}\ x')))$$

$$(\mathcal{N}_n\ f)\ (f\ x) = \underbrace{f\ (f\ \ldots (f}_{n}\ (f\ x))))$$

Представление натуральных чисел

Натуральные числа образуют индуктивное множество.

Формальные уравнения

$$\mathcal{N} = 0 \mid \mathcal{N} + 1$$

$$\begin{aligned}
0 &= 0 & \mathcal{N}_\mathbf{0} &= \lambda.f\,\lambda.x\,x \\
1 &= 0 + 1 & \mathcal{N}_\mathbf{1} &= \mathcal{S}\ \mathcal{N}_\mathbf{0} \\
2 &= 1 + 1 & \mathcal{N}_\mathbf{2} &= \mathcal{S}\ \mathcal{N}_\mathbf{1} \\
3 &= 2 + 1 & \mathcal{N}_\mathbf{3} &= \mathcal{S}\ \mathcal{N}_\mathbf{2} \\
&\vdots & &\vdots \\
N &= (N-1) + 1 & \mathcal{N}_\mathbf{n} &= \mathcal{S}\ \mathcal{N}_\mathbf{n-1}
\end{aligned}$$

Функция следования (инкремент)

$$\mathcal{S} = \lambda.n\,\lambda.f\,\lambda.x\,((n\ \ f)\ \ (f\ \ x))$$

$$\mathcal{S}\ \mathcal{N}_\mathbf{n} = \lambda.f\,\lambda.x\,((\mathcal{N}_n\ \ f)\ \ (f\ \ x))$$

$$\mathcal{N}_n\ f = \lambda.x'\,(\underbrace{f\ (f\ \ \ldots (f}_{n}\ \ x')))$$

$$(\mathcal{N}_n\ \ f)\ \ (f\ \ x) = \underbrace{f\ (f\ \ \ldots (f}_{n}\ \ (f\ \ x)))$$

$$\mathcal{S}\ \mathcal{N}_\mathbf{n} = \lambda.f\,\lambda.x\,(\underbrace{f\ (f\ \ \ldots (f}_{n}\ (f\ \ x)))) = \mathcal{N}_\mathbf{n+1}$$

Представление арифметики натуральных чисел

сложение

$$\mathbf{ADD} = \lambda.ab\,((a\ S)\ b)$$

Представление арифметики натуральных чисел

сложение

$$\mathbf{ADD} = \lambda.ab\,((a\ \mathcal{S})\ b)$$

$$\mathcal{N}_\mathbf{a}\ \mathcal{S} = \lambda.x\,(\underbrace{\mathcal{S}\ (\mathcal{S}\ldots(\mathcal{S}}_{a}\ x)))$$

$$(\mathcal{N}_\mathbf{a}\ \mathcal{S})\ \mathcal{N}_\mathbf{b} = \underbrace{\mathcal{S}\ (\mathcal{S}\ldots(\mathcal{S}}_{a}\ \mathcal{N}_\mathbf{b}))$$

Представление арифметики натуральных чисел

сложение

$$\mathbf{ADD} = \lambda.ab\,((a\ \mathcal{S})\ b)$$

$$\mathcal{N}_{\mathbf{a}}\ \mathcal{S} = \lambda.x\,(\underbrace{\mathcal{S}\ (\mathcal{S} \ldots (\mathcal{S}}_{a}\ x)))$$

$$(\mathcal{N}_{\mathbf{a}}\ \mathcal{S})\ \mathcal{N}_{\mathbf{b}} = \underbrace{\mathcal{S}\ (\mathcal{S} \ldots (\mathcal{S}}_{a}\ \mathcal{N}_{\mathbf{b}}))$$

вычитание

$$\mathbf{SUB} = \lambda.ab\,((b\ \mathcal{P})\ a)$$

Представление арифметики натуральных чисел

сложение

$\mathbf{ADD} \;= \lambda.ab\,((a\ \mathcal{S})\ b)$

$$\mathcal{N}_{\mathbf{a}}\ \mathcal{S} = \lambda.x\,(\underbrace{\mathcal{S}\ (\mathcal{S}\ldots(\mathcal{S}}_{a}\ x)))$$

$$(\mathcal{N}_{\mathbf{a}}\ \mathcal{S})\ \mathcal{N}_{\mathbf{b}} \;= \underbrace{\mathcal{S}\ (\mathcal{S}\ldots(\mathcal{S}}_{a}\ \mathcal{N}_{\mathbf{b}}))$$

умножение

$\mathbf{MUL} \;= \lambda.ab\,((a\ (b\ \mathcal{S}))\ \mathcal{N}_{\mathbf{0}}\,)$

вычитание

$\mathbf{SUB} = \lambda.ab\,((b\ \mathcal{P})\ a)$

$$(\mathcal{N}_{\mathbf{b}}\ \mathcal{P})\ \mathcal{N}_{\mathbf{a}} \;= \underbrace{\mathcal{P}\ (\mathcal{P}\ldots(\mathcal{P}}_{b}\ \mathcal{N}_{\mathbf{a}}))$$

Представление арифметики натуральных чисел

сложение

$\mathbf{ADD} = \lambda.ab\,((a\ \mathcal{S})\ b)$

$$\mathcal{N}_{\mathbf{a}}\ \mathcal{S} = \lambda.x\,(\underbrace{\mathcal{S}\ (\mathcal{S}\ldots(\mathcal{S}}_{a}\ x)))$$

$$(\mathcal{N}_{\mathbf{a}}\ \mathcal{S})\ \mathcal{N}_{\mathbf{b}} = \underbrace{\mathcal{S}\ (\mathcal{S}\ldots(\mathcal{S}}_{a}\ \mathcal{N}_{\mathbf{b}}))$$

умножение

$\mathbf{MUL} = \lambda.ab\,((a\ (b\ \mathcal{S}))\ \mathcal{N}_{\mathbf{0}})$

$$\mathcal{N}_{\mathbf{b}}\ \mathcal{S} = (\mathcal{N}_{\mathbf{b}}\ +)$$

$$(\mathcal{N}_{\mathbf{a}}\ (\mathcal{N}_{\mathbf{b}}\ \mathcal{S})) = \underbrace{(\mathcal{N}_{\mathbf{b}}\ +)\ ((\mathcal{N}_{\mathbf{b}}\ +)\ldots((\mathcal{N}_{\mathbf{b}}\ +)}_{a}\ \mathcal{N}_{\mathbf{0}}))$$

вычитание

$\mathbf{SUB} = \lambda.ab\,((b\ \mathcal{P})\ a)$

$$(\mathcal{N}_{\mathbf{b}}\ \mathcal{P})\ \mathcal{N}_{\mathbf{a}} = \underbrace{\mathcal{P}\ (\mathcal{P}\ldots(\mathcal{P}}_{b}\ \mathcal{N}_{\mathbf{a}}))$$

Представление арифметики натуральных чисел

сложение

$\mathbf{ADD} = \lambda.ab\,((a\ \mathcal{S})\ b)$

$$\mathcal{N}_\mathbf{a}\ \mathcal{S} = \lambda.x\,(\underbrace{\mathcal{S}\ (\mathcal{S} \ldots (\mathcal{S}}_{a}\ x)))$$

$$(\mathcal{N}_\mathbf{a}\ \mathcal{S})\ \mathcal{N}_\mathbf{b} = \underbrace{\mathcal{S}\ (\mathcal{S} \ldots (\mathcal{S}}_{a}\ \mathcal{N}_\mathbf{b}))$$

умножение

$\mathbf{MUL} = \lambda.ab\,((a\ (b\ \mathcal{S}))\ \mathcal{N}_\mathbf{0})$

$$\mathcal{N}_\mathbf{b}\ \mathcal{S} = (\mathcal{N}_\mathbf{b}\ +)$$

$$(\mathcal{N}_\mathbf{a}\ (\mathcal{N}_\mathbf{b}\ \mathcal{S})) = \underbrace{(\mathcal{N}_\mathbf{b}\ +)\ ((\mathcal{N}_\mathbf{b}\ +) \ldots ((\mathcal{N}_\mathbf{b}\ +)}_{a}\ \mathcal{N}_\mathbf{0}))$$

вычитание

$\mathbf{SUB} = \lambda.ab\,((b\ \mathcal{P})\ a)$

$$(\mathcal{N}_\mathbf{b}\ \mathcal{P})\ \mathcal{N}_\mathbf{a} = \underbrace{\mathcal{P}\ (\mathcal{P} \ldots (\mathcal{P}}_{b}\ \mathcal{N}_\mathbf{a}))$$

возведение в степень

$\mathbf{EXPT} = \lambda.ab\,(b\ a)$

Представление арифметики натуральных чисел

сложение

$$\mathbf{ADD} = \lambda.ab\,((a\ \mathcal{S})\ b)$$

$$\mathcal{N}_{\mathbf{a}}\ \mathcal{S} = \lambda.x\,(\underbrace{\mathcal{S}\ (\mathcal{S} \ldots (\mathcal{S}}_{a}\ x)))$$

$$(\mathcal{N}_{\mathbf{a}}\ \mathcal{S})\ \mathcal{N}_{\mathbf{b}} = \underbrace{\mathcal{S}\ (\mathcal{S} \ldots (\mathcal{S}}_{a}\ \mathcal{N}_{\mathbf{b}}))$$

умножение

$$\mathbf{MUL} = \lambda.ab\,((a\ (b\ \mathcal{S}))\ \mathcal{N}_{\mathbf{0}})$$

$$\mathcal{N}_{\mathbf{b}}\ \mathcal{S} = (\mathcal{N}_{\mathbf{b}}\ +)$$

$$(\mathcal{N}_{\mathbf{a}}\ (\mathcal{N}_{\mathbf{b}}\ \mathcal{S})) = \underbrace{(\mathcal{N}_{\mathbf{b}}\ +)\ ((\mathcal{N}_{\mathbf{b}}\ +) \ldots ((\mathcal{N}_{\mathbf{b}}\ +)}_{a}\ \mathcal{N}_{\mathbf{0}}))$$

вычитание

$$\mathbf{SUB} = \lambda.ab\,((b\ \mathcal{P})\ a)$$

$$(\mathcal{N}_{\mathbf{b}}\ \mathcal{P})\ \mathcal{N}_{\mathbf{a}} = \underbrace{\mathcal{P}\ (\mathcal{P} \ldots (\mathcal{P}}_{b}\ \mathcal{N}_{\mathbf{a}}))$$

возведение в степень

$$\mathbf{EXPT} = \lambda.ab\,(b\ a)$$

$$\mathcal{N}_{\mathbf{b}}\ \mathcal{N}_{\mathbf{a}} = \lambda.f\,\lambda.x\,(\underbrace{\mathcal{N}_{\mathbf{a}}\ (\mathcal{N}_{\mathbf{a}} \ldots (\mathcal{N}_{\mathbf{a}}}_{b}\ x)))$$

Представление арифметики натуральных чисел

Построим ряд:

$$(0,0),(1,0),(2,1),(3,2),...(n,n-1)$$

Правый элемент n-ной пары равен $n-1$.

Представление арифметики натуральных чисел

Построим ряд:

$$(0, 0), (1, 0), (2, 1), (3, 2), ...(n, n - 1)$$

Правый элемент n-ной пары равен $n - 1$.

Вычисление предшествующего числа:

$$\mathcal{P} = \lambda.n\,(\mathbf{CDR}\ ((n\ \mathbf{NEXT})\ (\mathbf{CONS}\ \mathcal{N}_\mathbf{0}\ \mathcal{N}_\mathbf{0})))$$

$$\mathbf{NEXT} = \lambda.p\,(\mathbf{CONS}\ (\mathcal{S}\ (\mathbf{CAR}\ p))\ (\mathbf{CAR}\ p))$$

Представление арифметики натуральных чисел

Построим ряд:

$$(0,0),(1,0),(2,1),(3,2),...(n,n-1)$$

Правый элемент n-ной пары равен $n-1$.

Вычисление предшествующего числа:

$$\mathcal{P} = \lambda.n\,(\mathbf{CDR}\ ((n\ \mathbf{NEXT})\ (\mathbf{CONS}\ \mathcal{N}_\mathbf{0}\ \mathcal{N}_\mathbf{0})))$$

$$\mathbf{NEXT} = \lambda.p\,(\mathbf{CONS}\ (\mathcal{S}\ (\mathbf{CAR}\ p))\ (\mathbf{CAR}\ p))$$

Отношения порядка для чисел:

$$\mathbf{LEQ}? \ = \lambda.ab\,(\mathbf{ZERO}?\ (\mathbf{SUB}\ a\ b))$$

$$\mathbf{EQL}? \ = \lambda.ab\,(\mathbf{AND}\ (\mathbf{LEQ}?\ a\ b)\ (\mathbf{LEQ}?\ b\ a))$$

Представление арифметики натуральных чисел

Построим ряд:

$$(0,0),(1,0),(2,1),(3,2),...(n,n-1)$$

Правый элемент n-ной пары равен $n-1$.

Вычисление предшествующего числа:

$$\mathcal{P} = \lambda.n\,(\mathbf{CDR}\;((n\;\mathbf{NEXT})\;(\mathbf{CONS}\;\mathcal{N}_\mathbf{0}\;\mathcal{N}_\mathbf{0})))$$

$$\mathbf{NEXT} = \lambda.p\,(\mathbf{CONS}\;(\mathcal{S}\;(\mathbf{CAR}\;p))\;(\mathbf{CAR}\;p))$$

Отношения порядка для чисел:

$$\mathbf{LEQ?}\;\; = \lambda.ab\,(\mathbf{ZERO?}\;(\mathbf{SUB}\;a\;b))$$

$$\mathbf{EQL?}\;\; = \lambda.ab\,(\mathbf{AND}\;(\mathbf{LEQ?}\;a\;b)\;(\mathbf{LEQ?}\;b\;a))$$

Формальные уравнения

$$\mathbf{ZERO?}\;\mathcal{N}_\mathbf{0}\;\; = \mathbf{T}$$
$$\mathbf{ZERO?}\;\mathcal{N}_\mathbf{a}\;\; = \mathbf{F}$$

Представление арифметики натуральных чисел

Построим ряд:

$$(0,0),(1,0),(2,1),(3,2),...(n,n-1)$$

Правый элемент n-ной пары равен $n-1$.

Вычисление предшествующего числа:

$$\mathcal{P} = \lambda.n\,(\mathbf{CDR}\ ((n\ \mathbf{NEXT})\ (\mathbf{CONS}\ \mathcal{N}_\mathbf{0}\ \mathcal{N}_\mathbf{0})))$$

$$\mathbf{NEXT} = \lambda.p\,(\mathbf{CONS}\ (\mathcal{S}\ (\mathbf{CAR}\ p))\ (\mathbf{CAR}\ p))$$

Отношения порядка для чисел:

$$\mathbf{LEQ?} \ = \lambda.ab\,(\mathbf{ZERO?}\ (\mathbf{SUB}\ a\ b))$$

$$\mathbf{EQL?} \ = \lambda.ab\,(\mathbf{AND}\ (\mathbf{LEQ?}\ a\ b)\ (\mathbf{LEQ?}\ b\ a))$$

Формальные уравнения

$$\mathbf{ZERO?}\ \mathcal{N}_\mathbf{0} \ = \mathbf{T}$$
$$\mathbf{ZERO?}\ \mathcal{N}_\mathbf{a} \ = \mathbf{F}$$

Предикат равенства нулю:

$$\mathbf{ZERO?} \ = \lambda.n\,((n\ (\lambda.x\,\mathbf{F}))\ \mathbf{T})$$

Представление арифметики натуральных чисел

Построим ряд:

$$(0,0),(1,0),(2,1),(3,2),...(n,n-1)$$

Правый элемент n-ной пары равен $n-1$.

Вычисление предшествующего числа:

$$\mathcal{P} = \lambda.n\,(\mathbf{CDR}\ ((n\ \mathbf{NEXT})\ (\mathbf{CONS}\ \mathcal{N}_\mathbf{0}\ \mathcal{N}_\mathbf{0})))$$

$$\mathbf{NEXT} = \lambda.p\,(\mathbf{CONS}\ (\mathcal{S}\ (\mathbf{CAR}\ p))\ (\mathbf{CAR}\ p))$$

Отношения порядка для чисел:

$$\mathbf{LEQ?} = \lambda.ab\,(\mathbf{ZERO?}\ (\mathbf{SUB}\ a\ b))$$

$$\mathbf{EQL?} = \lambda.ab\,(\mathbf{AND}\ (\mathbf{LEQ?}\ a\ b)\ (\mathbf{LEQ?}\ b\ a))$$

Формальные уравнения

$$\mathbf{ZERO?}\ \mathcal{N}_\mathbf{0} = \mathbf{T}$$
$$\mathbf{ZERO?}\ \mathcal{N}_\mathbf{a} = \mathbf{F}$$

Предикат равенства нулю:

$$\mathbf{ZERO?} = \lambda.n\,((n\ (\lambda.x\,\mathbf{F}))\ \mathbf{T})$$

$$\begin{aligned}\mathbf{ZERO?}\ \mathcal{N}_\mathbf{0} &= (((\lambda.f\,\lambda.x\,x)\ (\lambda.x\,\mathbf{F}))\ \mathbf{T})\\ &= ((\lambda.x\,x)\ \mathbf{T}) = \mathbf{T}\end{aligned}$$

Представление арифметики натуральных чисел

Построим ряд:

$$(0,0),(1,0),(2,1),(3,2),...(n,n-1)$$

Правый элемент n-ной пары равен $n-1$.

Вычисление предшествующего числа:

$$\mathcal{P} = \lambda.n\,(\mathbf{CDR}\ ((n\ \mathbf{NEXT})\ (\mathbf{CONS}\ \mathcal{N}_0\ \mathcal{N}_0)))$$

$$\mathbf{NEXT} = \lambda.p\,(\mathbf{CONS}\ (\mathcal{S}\ (\mathbf{CAR}\,p))\ (\mathbf{CAR}\,p))$$

Отношения порядка для чисел:

$$\mathbf{LEQ}? \ = \lambda.ab\,(\mathbf{ZERO}?\ (\mathbf{SUB}\ a\ b))$$

$$\mathbf{EQL}? \ = \lambda.ab\,(\mathbf{AND}\ (\mathbf{LEQ}?\ a\ b)\ (\mathbf{LEQ}?\ b\ a))$$

Формальные уравнения

$$\mathbf{ZERO}?\ \mathcal{N}_0 \ = \mathbf{T}$$
$$\mathbf{ZERO}?\ \mathcal{N}_a \ = \mathbf{F}$$

Предикат равенства нулю:

$$\mathbf{ZERO}? \ = \lambda.n\,((n\ (\lambda.x\,\mathbf{F}))\ \mathbf{T})$$

$$\begin{aligned}\mathbf{ZERO}?\ \mathcal{N}_0 &= (((\lambda.f\,\lambda.x\,x)\ (\lambda.x\,\mathbf{F}))\ \mathbf{T})\\ &= ((\lambda.x\,x)\ \mathbf{T}) = \mathbf{T}\end{aligned}$$

$$\begin{aligned}\mathbf{ZERO}?\ \mathcal{N}_2 &= (((\lambda.f\,\lambda.x'\,(f\ (f\ x')))\ (\lambda.x\,\mathbf{F}))\ \mathbf{T})\\ &= ((\lambda.x'\,((\lambda.x\,\mathbf{F})\ ((\lambda.x\,\mathbf{F})\ x')))\ \mathbf{T})\\ &= ((\lambda.x'\,\mathbf{F})\ \mathbf{T}) = \mathbf{F}\end{aligned}$$

Промежуточный итог

управление потоком вычислений и логика

$$\mathbf{T} = \lambda.qr\, q$$
$$\mathbf{F} = \lambda.qr\, r$$
$$\mathbf{NOT} = \lambda.q\,(q\ \mathbf{F}\ \mathbf{T})$$
$$\mathbf{IF} = \lambda.pqr\,(p\ q\ r)$$
$$\mathbf{AND} = \lambda.qr\,(q\ r\ q)$$
$$\mathbf{OR} = \lambda.qr\,(q\ q\ r)$$

средства абстракции данных

$$\mathbf{CONS} = \lambda.ab\,(\lambda.m\,(m\ a\ b))$$
$$\mathbf{CAR} = \lambda.p\,(p\ (\lambda.xy\, x))$$
$$\mathbf{CDR} = \lambda.p\,(p\ (\lambda.xy\, y))$$
$$\mathbf{NULL?} = \lambda.p\,(p\ (\lambda.xy\, \mathbf{F}))$$
$$\mathbf{NULL} = \lambda.p\, \mathbf{T}$$

элементарные числовые данные и арифметика

$$\mathcal{N}_\mathbf{0} = \lambda.f\, \lambda.x\, x \quad \mathcal{N}_\mathbf{n} = \mathcal{S}\, \mathcal{N}_\mathbf{n-1}$$
$$\mathcal{S} = \lambda.n\, \lambda.f\, \lambda.x\, ((n\ f)\ (f\ x))$$
$$\mathcal{P} = \lambda.n\,(\mathbf{CDR}\ ((n\ \mathbf{NEXT})\ (\mathbf{CONS}\ \mathcal{N}_\mathbf{0}\ \mathcal{N}_\mathbf{0})))$$
$$\mathbf{NEXT} = \lambda.p\,(\mathbf{CONS}\ (\mathcal{S}\ (\mathbf{CAR}\ p))\ (\mathbf{CAR}\ p))$$
$$\mathbf{ADD} = \lambda.ab\,((a\ \mathcal{S})\ b)$$
$$\mathbf{SUB} = \lambda.ab\,((b\ \mathcal{P})\ a)$$
$$\mathbf{MUL} = \lambda.ab\,((a\ (b\ \mathcal{S}))\ \mathcal{N}_\mathbf{0})$$
$$\mathbf{EXPT} = \lambda.ab\,(b\ a)$$
$$\mathbf{ZERO?} = \lambda.n\,((n\ (\lambda.x\, \mathbf{F}))\ \mathbf{T})$$
$$\mathbf{LEQ?} = \lambda.ab\,(\mathbf{ZERO?}\ (\mathbf{SUB}\ a\ b))$$
$$\mathbf{EQL?} = \lambda.ab\,(\mathbf{AND}\ (\mathbf{LEQ?}\ a\ b)\ (\mathbf{LEQ?}\ b\ a))$$

числа выполняют роль итератора (цикла `for`)

Для полноты по Тьюрингу не хватает цикла `while` (рекурсии).

Проблема рекурсии в λ-исчислении

До сих пор мы использовали связывание $(=)$ только для создания псевдонимов λ-функциям.

Проблема рекурсии в λ-исчислении

До сих пор мы использовали связывание $(=)$ только для создания псевдонимов λ-функциям.

Псевдоним всегда можно заменить функцией и получить конечное выражение, состоящее только из λ-функций.

Проблема рекурсии в λ-исчислении

До сих пор мы использовали связывание $(=)$ только для создания псевдонимов λ-функциям.

Псевдоним всегда можно заменить функцией и получить конечное выражение, состоящее только из λ-функций.

Попробуем написать рекурсивную функцию:

$$\begin{aligned}\mathbf{FACT} = \lambda n.\ (\mathbf{IF}\ (&\mathbf{ZERO?}\ n)\\ &\mathcal{N}_\mathbf{1}\\ &(\mathbf{MUL}\ n\ (\mathbf{FACT}\ (\mathcal{P}\ n))))\end{aligned}$$

Проблема рекурсии в λ-исчислении

До сих пор мы использовали связывание $(=)$ только для создания псевдонимов λ-функциям.

Псевдоним всегда можно заменить функцией и получить конечное выражение, состоящее только из λ-функций.

Попробуем написать рекурсивную функцию:

$$
\begin{aligned}
\mathbf{FACT} = \lambda n.\ (\mathbf{IF}\ &(\mathbf{ZERO?}\ n) \\
&\mathcal{N}_{\mathbf{1}} \\
&(\mathbf{MUL}\ n\ (\mathbf{FACT}\ (\mathcal{P}\ n))))
\end{aligned}
$$

Для вычисления на место $\mathbf{FACT}$ мы должны поставить её определение, в свою очередь содержащее $\mathbf{FACT}$. И так — бесконечное число раз!

Проблема рекурсии в λ-исчислении

До сих пор мы использовали связывание $(=)$ только для создания псевдонимов λ-функциям.

Псевдоним всегда можно заменить функцией и получить конечное выражение, состоящее только из λ-функций.

Попробуем написать рекурсивную функцию:

$$\begin{aligned}\mathbf{FACT} = \lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ & \\ & \mathcal{N}_{\mathbf{1}} \\ & (\mathbf{MUL}\ n\ (\mathbf{FACT}\ (\mathcal{P}\ n))))\end{aligned}$$

Для вычисления на место **FACT** мы должны поставить её определение, в свою очередь содержащее **FACT**. И так — бесконечное число раз!

Задержим вычисления с помощью замыкания:

$$\begin{aligned}\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ & \\ & \mathcal{N}_{\mathbf{1}} \\ & (\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))\end{aligned}$$

Проблема рекурсии в λ-исчислении

До сих пор мы использовали связывание $(=)$ только для создания псевдонимов λ-функциям.

Псевдоним всегда можно заменить функцией и получить конечное выражение, состоящее только из λ-функций.

Попробуем написать рекурсивную функцию:

$$\mathbf{FACT} = \lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_\mathbf{1}\\ (\mathbf{MUL}\ n\ (\mathbf{FACT}\ (\mathcal{P}\ n))))$$

Для вычисления на место $\mathbf{FACT}$ мы должны поставить её определение, в свою очередь содержащее $\mathbf{FACT}$. И так — бесконечное число раз!

Задержим вычисления с помощью замыкания:

$$\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_\mathbf{1}\\ (\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))$$

Чтобы запустить вычисления, нужно вызвать функцию Φ в такой форме:

$$(\Phi\ \Phi)\ \mathcal{N}_\mathbf{n}\ .$$

Проблема рекурсии в λ-исчислении

До сих пор мы использовали связывание $(=)$ только для создания псевдонимов λ-функциям.

Псевдоним всегда можно заменить функцией и получить конечное выражение, состоящее только из λ-функций.

Попробуем написать рекурсивную функцию:

$$\begin{aligned}\mathbf{FACT} = \lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_\mathbf{1}\\ (\mathbf{MUL}\ n\ (\mathbf{FACT}\ (\mathcal{P}\ n))))\end{aligned}$$

Для вычисления на место **FACT** мы должны поставить её определение, в свою очередь содержащее **FACT**. И так — бесконечное число раз!

Задержим вычисления с помощью замыкания:

$$\begin{aligned}\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_\mathbf{1}\\ (\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))\end{aligned}$$

Чтобы запустить вычисления, нужно вызвать функцию Φ в такой форме:

$$(\Phi\ \Phi)\ \mathcal{N}_\mathbf{n}\ .$$

$$\mathbf{FACT} = \lambda.x\,((\Phi\ \Phi)\ x) = U\ \Phi,$$

где

$$U = \lambda.f\,\lambda.x\,((f\ \ f)\ x)$$

Проблема рекурсии в λ-исчислении

До сих пор мы использовали связывание $(=)$ только для создания псевдонимов λ-функциям.

Псевдоним всегда можно заменить функцией и получить конечное выражение, состоящее только из λ-функций.

Попробуем написать рекурсивную функцию:

$$\begin{aligned}\mathbf{FACT} = \lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ & \\ & \mathcal{N}_{\mathbf{1}} \\ & (\mathbf{MUL}\ n\ (\mathbf{FACT}\ (\mathcal{P}\ n))))\end{aligned}$$

Для вычисления на место **FACT** мы должны поставить её определение, в свою очередь содержащее **FACT**. И так — бесконечное число раз!

Задержим вычисления с помощью замыкания:

$$\begin{aligned}\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ & \\ & \mathcal{N}_{\mathbf{1}} \\ & (\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))\end{aligned}$$

Чтобы запустить вычисления, нужно вызвать функцию Φ в такой форме:

$$(\Phi\ \Phi)\ \mathcal{N}_{\mathbf{n}}\ .$$

$$\mathbf{FACT} = \lambda.x\,((\Phi\ \Phi)\ x) = U\ \Phi,$$

где

$$U = \lambda.f\,\lambda.x\,((f\ \ f)\ x)$$

Функция U называется U-комбинатором.

Рекурсия, как поиск неподвижной точки

Существует иной подход.

$$\begin{aligned}\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (&\mathbf{ZERO}?\ n)\\ &\mathcal{N}_{\mathbf{1}}\\ &(\mathbf{MUL}\ n\ (f\ (\mathcal{P}\ n))))\end{aligned}$$

Рекурсия, как поиск неподвижной точки

Существует иной подход.

$$\begin{aligned}\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (&\mathbf{ZERO}?\ n)\\ &\mathcal{N}_\mathbf{1}\\ &(\mathbf{MUL}\ n\ (f\ (\mathcal{P}\ n))))\end{aligned}$$

Желая передать этой фукции саму себя, мы определяем следующее уравнение:

$$\mathbf{FACT} = \Phi\ \mathbf{FACT}.$$

Рекурсия, как поиск неподвижной точки

Существует иной подход.

$$
\begin{aligned}
\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO}?\ n)\\
\mathcal{N}_{\mathbf{1}}\\
(\mathbf{MUL}\ n\ (f\ (\mathcal{P}\ n))))
\end{aligned}
$$

Желая передать этой фукции саму себя, мы определяем следующее уравнение:

$$\mathbf{FACT} = \Phi\ \mathbf{FACT}.$$

То есть, функция $\mathbf{FACT}$ определяется, как неподвижная точка функции Φ.

Рекурсия, как поиск неподвижной точки

Существует иной подход.

$$\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_{\mathbf{1}}\ (\mathbf{MUL}\ n\ (f\ (\mathcal{P}\ n))))$$

Желая передать этой фукции саму себя, мы определяем следующее уравнение:

$$\mathbf{FACT} = \Phi\ \mathbf{FACT}.$$

То есть, функция $\mathbf{FACT}$ определяется, как неподвижная точка функции Φ.

Формально, мы ищем такую функцию Y, которая удовлетворяла бы уравнению:

$$Y\ f = f\ (Y\ f). \tag{1}$$

Рекурсия, как поиск неподвижной точки

Существует иной подход.

$$\begin{aligned}\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_1\\ (\mathbf{MUL}\ n\ (f\ (\mathcal{P}\ n))))\end{aligned}$$

Желая передать этой фукции саму себя, мы определяем следующее уравнение:

$$\mathbf{FACT} = \Phi\ \mathbf{FACT}.$$

То есть, функция $\mathbf{FACT}$ определяется, как неподвижная точка функции Φ.

Формально, мы ищем такую функцию Y, которая удовлетворяла бы уравнению:

$$Y\ f = f\ (Y\ f). \qquad (1)$$

Тогда мы могли бы записать:

$$\begin{aligned}\mathbf{FACT} = Y\ (\lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_1\\ (\mathbf{MUL}\ n\ (f\ (\mathcal{P}\ n)))))\end{aligned}$$

Рекурсия, как поиск неподвижной точки

Существует иной подход.

$$\Phi = \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_\mathbf{1}\\ (\mathbf{MUL}\ n\ (f\ (\mathcal{P}\ n))))$$

Желая передать этой фукции саму себя, мы определяем следующее уравнение:

$$\mathbf{FACT} = \Phi\ \mathbf{FACT}.$$

То есть, функция $\mathbf{FACT}$ определяется, как неподвижная точка функции Φ.

Формально, мы ищем такую функцию Y, которая удовлетворяла бы уравнению:

$$Y\ f = f\ (Y\ f). \tag{1}$$

Тогда мы могли бы записать:

$$\mathbf{FACT} = Y\ (\lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_\mathbf{1}\\ (\mathbf{MUL}\ n\ (f\ (\mathcal{P}\ n)))))$$

Функция, удовлетворяющая уравнению (1) называется комбинатором неподвижной точки или Y-комбинатором.

Вывод нормальной формы Y-комбинатора.

$$
\begin{aligned}
\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ & (\mathbf{ZERO?}\ n) \\
& \mathcal{N}_{\mathbf{1}} \\
& (\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))
\end{aligned}
$$

Вывод нормальной формы Y-комбинатора.

$$
\begin{aligned}
\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ &(\mathbf{ZERO}?\ n)\\
&\mathcal{N}_{\mathbf{1}}\\
&(\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))
\end{aligned}
$$

$$(f\ \ f) \leftarrow_{\eta} \lambda.x\,((f\ \ f)\ \ x) = U\ \ f$$

Вывод нормальной формы Y-комбинатора.

$$
\begin{aligned}
\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (&\mathbf{ZERO?}\ n)\\
&\mathcal{N}_{\mathbf{1}}\\
&(\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))
\end{aligned}
$$

$$(f\ \ f) \leftarrow_{\eta} \lambda.x\,((f\ \ f)\ x) = U\ \ f$$

$$
\begin{aligned}
\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (&\mathbf{ZERO?}\ n)\\
&\mathcal{N}_{\mathbf{1}}\\
&(\mathbf{MUL}\ n\ ((U\ \ f)\ (\mathcal{P}\ n))))
\end{aligned}
$$

Вывод нормальной формы Y-комбинатора.

$$\begin{aligned}\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_1\\ (\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))\end{aligned}$$

$$(f\ \ f) \leftarrow_\eta \lambda.x\,((f\ \ f)\ x) = U\ \ f$$

$$\begin{aligned}\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_1\\ (\mathbf{MUL}\ n\ ((U\ \ f)\ (\mathcal{P}\ n))))\end{aligned}$$

Вынесем функцию $(U\ \ f)$ из тела функции $\lambda.n$:

$$\begin{aligned}\mathbf{FACT} = U\ \lambda f.((\lambda f'.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\\ \mathcal{N}_1\\ (\mathbf{MUL}\ n\ (f'\ (\mathcal{P}\ n))))) (U\ \ f))\end{aligned}$$

Вывод нормальной формы Y-комбинатора.

$$\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_1\ (\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))$$

$$(f\ \ f) \leftarrow_\eta \lambda.x\,((f\ \ f)\ x) = U\ \ f$$

$$\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_1\ (\mathbf{MUL}\ n\ ((U\ \ f)\ (\mathcal{P}\ n))))$$

Вынесем функцию $(U\ \ f)$ из тела функции $\lambda.n$:

$$\mathbf{FACT} = U\ \lambda f.((\lambda f'.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_1\ (\mathbf{MUL}\ n\ (f'\ (\mathcal{P}\ n))))))\ (U\ \ f))$$

Функция $\lambda.f'$ ... α-эквивалентна функции Φ.

$$\mathbf{FACT} = U\ \lambda f.(\Phi\ (U\ \ f))$$

Вывод нормальной формы Y-комбинатора.

$$\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_1\ (\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))$$

$$(f\ \ f) \leftarrow_\eta \lambda.x\,((f\ \ f)\ x) = U\ \ f$$

$$\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_1\ (\mathbf{MUL}\ n\ ((U\ \ f)\ (\mathcal{P}\ n))))$$

Вынесем функцию $(U\ \ f)$ из тела функции $\lambda.n$:

$$\mathbf{FACT} = U\ \lambda f.((\lambda f'.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_1\ (\mathbf{MUL}\ n\ (f'\ (\mathcal{P}\ n))))) (U\ \ f))$$

Функция $\lambda.f'$ ... α-эквивалентна функции Φ.

$$\mathbf{FACT} = U\ \lambda f.(\Phi\ (U\ \ f))$$

Выделим Φ из тела функции:

$$\mathbf{FACT} = (\lambda.b\,(U\ \lambda.f\,(b\ (U\ \ f))))\ \Phi$$

Вывод нормальной формы Y-комбинатора.

$$\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_\mathbf{1}\ (\mathbf{MUL}\ n\ ((f\ \ f)\ (\mathcal{P}\ n))))$$

$$(f\ \ f) \leftarrow_\eta \lambda.x\,((f\ \ f)\ x) = U\ \ f$$

$$\mathbf{FACT} = U\ \lambda f.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_\mathbf{1}\ (\mathbf{MUL}\ n\ ((U\ \ f)\ (\mathcal{P}\ n))))$$

Вынесем функцию $(U\ \ f)$ из тела функции $\lambda.n$:

$$\mathbf{FACT} = U\ \lambda f.((\lambda f'.\lambda n.\ (\mathbf{IF}\ (\mathbf{ZERO?}\ n)\ \mathcal{N}_\mathbf{1}\ (\mathbf{MUL}\ n\ (f'\ (\mathcal{P}\ n))))) \ (U\ \ f))$$

Функция $\lambda.f'\ \dots\ \alpha$-эквивалентна функции Φ.

$$\mathbf{FACT} = U\ \lambda f.(\Phi\ (U\ \ f))$$

Выделим Φ из тела функции:

$$\mathbf{FACT} = (\lambda.b\,(U\ \lambda.f\,(b\ (U\ \ f))))\ \Phi$$

То, чем мы действуем на функцию Φ и есть искомый комбинатор:

$$\mathbf{FACT} = Y\ \Phi$$
$$Y = \lambda.b\,(U\ \lambda.f\,(b\ (U\ \ f)))$$

Свойства Y-комбинатора.

Выведенная нами форма работает как в строгих, так и в ленивых вычислениях.

$$\begin{aligned} Y &= \lambda.b\,(U\ \lambda.f\,(b\ (U\ f))) \\ &= \lambda.b\,(\lambda.f\,(b\ (\lambda.x\,((f\ f)\ x))) \\ &\qquad\quad \lambda.f\,(b\ (\lambda.x\,((f\ f)\ x)))) \end{aligned}$$

Свойства Y-комбинатора.

Выведенная нами форма работает как в строгих, так и в ленивых вычислениях.

$$\begin{aligned} Y &= \lambda.b\,(U\ \lambda.f\,(b\ (U\ \ f))) \\ &= \lambda.b\,(\lambda.f\,(b\ (\lambda.x\,((f\ \ f)\ x))) \\ &\qquad\qquad \lambda.f\,(b\ (\lambda.x\,((f\ \ f)\ x)))) \end{aligned}$$

Для ленивых вычислений можно провести η-редукцию: $\lambda.x\,((f\ \ f)\ x) \to_\eta (f\ \ f)$

$$Y = \lambda.b\,((\lambda.f\,(b\ (f\ \ f)))\ (\lambda.f\,(b\ (f\ \ f))))$$

Свойства Y-комбинатора.

Выведенная нами форма работает как в строгих, так и в ленивых вычислениях.

$$
\begin{aligned}
Y &= \lambda.b\,(U\ \lambda.f\ (b\ (U\ \ f))) \\
&= \lambda.b\,(\lambda.f\ (b\ (\lambda.x\,((f\ \ f)\ x))) \\
&\qquad\qquad \lambda.f\ (b\ (\lambda.x\,((f\ \ f)\ x))))
\end{aligned}
$$

Для ленивых вычислений можно провести η-редукцию: $\lambda.x\,((f\ \ f)\ x) \rightarrow_\eta (f\ \ f)$

$$Y = \lambda.b\,((\lambda.f\ (b\ (f\ \ f)))\ (\lambda.f\ (b\ (f\ \ f))))$$

Это – простейшая из возможных нормальных форм Y-комбинатора.

Свойства Y-комбинатора.

Выведенная нами форма работает как в строгих, так и в ленивых вычислениях.

$$\begin{aligned} Y &= \lambda.b\,(U\ \lambda.f\,(b\ (U\ \ f))) \\ &= \lambda.b\,(\lambda.f\,(b\ (\lambda.x\,((f\ \ f)\ x))) \\ &\qquad\qquad \lambda.f\,(b\ (\lambda.x\,((f\ \ f)\ x)))) \end{aligned}$$

Для ленивых вычислений можно провести η-редукцию: $\lambda.x\,((f\ \ f)\ x) \to_\eta (f\ \ f)$

$$Y = \lambda.b\,((\lambda.f\,(b\ (f\ \ f)))\ (\lambda.f\,(b\ (f\ \ f))))$$

Это – простейшая из возможных нормальных форм Y-комбинатора.

Важные теоремы:

Свойства Y-комбинатора.

Выведенная нами форма работает как в строгих, так и в ленивых вычислениях.

$$\begin{aligned} Y &= \lambda.b\,(U\ \lambda.f\,(b\ (U\ \ f))) \\ &= \lambda.b\,(\lambda.f\,(b\ (\lambda.x\,((f\ \ f)\ \ x))) \\ &\qquad\qquad \lambda.f\,(b\ (\lambda.x\,((f\ \ f)\ \ x)))) \end{aligned}$$

Для ленивых вычислений можно провести η-редукцию: $\lambda.x\,((f\ \ f)\ \ x) \to_\eta (f\ \ f)$

$$Y = \lambda.b\,((\lambda.f\,(b\ (f\ \ f)))\ (\lambda.f\,(b\ (f\ \ f))))$$

Это – простейшая из возможных нормальных форм Y-комбинатора.

Важные теоремы:

В рамках безтипового λ-исчисления каждая функция имеет по крайней мере одну неподвижную точку.

Свойства Y-комбинатора.

Выведенная нами форма работает как в строгих, так и в ленивых вычислениях.

$$
\begin{aligned}
Y &= \lambda.b\,(U\ \lambda.f\,(b\ (U\ \ f))) \\
&= \lambda.b\,(\lambda.f\,(b\ (\lambda.x\,((f\ \ f)\ \ x))) \\
&\qquad\qquad \lambda.f\,(b\ (\lambda.x\,((f\ \ f)\ \ x))))
\end{aligned}
$$

Для ленивых вычислений можно провести η-редукцию: $\lambda.x\,((f\ \ f)\ \ x) \to_\eta (f\ \ f)$

$$Y = \lambda.b\,((\lambda.f\,(b\ (f\ \ f)))\ (\lambda.f\,(b\ (f\ \ f))))$$

Это – простейшая из возможных нормальных форм Y-комбинатора.

Важные теоремы:

В рамках безтипового λ-исчисления каждая функция имеет по крайней мере одну неподвижную точку.

Теорема Майера Гольдберга

Существует бесконечное счётное множество нормальных форм Y-комбинатора.

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,A[x \to y]$,

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_\beta A[x \to B]$,

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\;x) \to_\eta A$.

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\, A \leftrightarrow_\alpha \lambda.y\, A[x \to y]$,
   b) β-редукция: $(\lambda.x\, A)\, B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\, (A\ x) \to_\eta A$.

Это позволяет строить:

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\, A \leftrightarrow_\alpha \lambda.y\, A[x \to y]$,
   b) β-редукция: $(\lambda.x\, A)\, B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\, (A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\, A \leftrightarrow_\alpha \lambda.y\, A[x \to y]$,
   b) β-редукция: $(\lambda.x\, A)\, B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\, A \leftrightarrow_\alpha \lambda.y\, A[x \to y]$,
   b) β-редукция: $(\lambda.x\, A)\, B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\, A \leftrightarrow_{\alpha} \lambda.y\, A[x \to y]$,
   b) β-редукция: $(\lambda.x\, A)\, B \to_{\beta} A[x \to B]$,
   c) η-редукция: $\lambda.x\, (A\ x) \to_{\eta} A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\, A \leftrightarrow_\alpha \lambda.y\, A[x \to y]$,
   b) β-редукция: $(\lambda.x\, A)\, B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\, (A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,
   b) индуктивное множество натуральных чисел и операции над ним,

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,
   b) индуктивное множество натуральных чисел и операции над ним,
   c) составные функциональные типы данных

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,
   b) индуктивное множество натуральных чисел и операции над ним,
   c) составные функциональные типы данных
      1) точечные пары и списки,

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,
   b) индуктивное множество натуральных чисел и операции над ним,
   c) составные функциональные типы данных
      1) точечные пары и списки,
      2) очереди, стеки, деревья и т.п.

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_{\alpha} \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_{\beta} A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\;x) \to_{\eta} A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,
   b) индуктивное множество натуральных чисел и операции над ним,
   c) составные функциональные типы данных
      1) точечные пары и списки,
      2) очереди, стеки, деревья и т.п.
3. абстракцию процедур:

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,
   b) индуктивное множество натуральных чисел и операции над ним,
   c) составные функциональные типы данных
      1) точечные пары и списки,
      2) очереди, стеки, деревья и т.п.
3. абстракцию процедур:
   a) функционалы и операторы

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_{\alpha} \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_{\beta} A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\;x) \to_{\eta} A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,
   b) индуктивное множество натуральных чисел и операции над ним,
   c) составные функциональные типы данных
      1) точечные пары и списки,
      2) очереди, стеки, деревья и т.п.
3. абстракцию процедур:
   a) функционалы и операторы
   b) структурную рекурсию (свёртку)

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\, A \leftrightarrow_\alpha \lambda.y\, A[x \to y]$,
   b) β-редукция: $(\lambda.x\, A)\, B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\, (A\ x) \to_\eta A$.

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,
   b) индуктивное множество натуральных чисел и операции над ним,
   c) составные функциональные типы данных
      1) точечные пары и списки,
      2) очереди, стеки, деревья и т.п.
3. абстракцию процедур:
   a) функционалы и операторы
   b) структурную рекурсию (свёртку)
   c) замыкания и объекты

Подведение итогов

Для построения чистого бестипового λ-исчисления нам потребовались:

1. базовые операции:
   a) абстракция,
   b) аппликация;
2. правила вывода:
   a) α-преобразование: $\lambda.x\,A \leftrightarrow_\alpha \lambda.y\,A[x \to y]$,
   b) β-редукция: $(\lambda.x\,A)\,B \to_\beta A[x \to B]$,
   c) η-редукция: $\lambda.x\,(A\ x) \to_\eta A$.

Существенное ограничение:

В бестиповом λ-исчислении невозможно установить отношение эквивалентности функциональных данных в общем виде

Это позволяет строить:

1. управляющие конструкции:
   a) ветвление,
   b) итерационные и условные циклы (рекурсию);
2. абстракцию данных:
   a) логические данные и операции над ними,
   b) индуктивное множество натуральных чисел и операции над ним,
   c) составные функциональные типы данных
      1) точечные пары и списки,
      2) очереди, стеки, деревья и т.п.
3. абстракцию процедур:
   a) функционалы и операторы
   b) структурную рекурсию (свёртку)
   c) замыкания и объекты