

## Домашнее задание №2: «формализация лямбда-исчисления»

1. Придумайте грамматику для лямбда-выражений, однозначно разбирающую любое выражение (в частности, учитывающую все сокращения скобок в записи).

Решение.

- $S \rightarrow \lambda T.S$
- $S \rightarrow S S$
- $S \rightarrow T$
- $S \rightarrow (S)$
- $T \rightarrow \Sigma$

□

2. Приведите пример лямбда-выражения, корректная бета-редукция которого невозможна без переименования связанных переменных. Возможно ли, чтобы в этом выражении все переменные в лямбда-абстракциях были различными?

Решение.  $(\lambda y. \lambda x. y x) (\lambda x. x)$

□

3. Два выражения  $A$  и  $B$  назовём родственными, если существует  $C$ , что  $A \rightarrow_{\beta} C$  и  $B \rightarrow_{\beta} C$ . Как соотносится родственность и бета-эквивалентность?

Решение.

Обозначение.  $A$  родственн.  $B \Leftrightarrow A \sim B$

$$A \sim B \Rightarrow \begin{cases} A \rightarrow_{\beta} C \\ B \rightarrow_{\beta} C \end{cases} \Rightarrow \begin{cases} A =_{\beta} C \\ B =_{\beta} C \end{cases} \Rightarrow A =_{\beta} B$$

В обратную сторону тоже верно по ромбовидному свойству.

□

4. Рассмотрим представление лямбда-выражений де Брауна (de Bruijn): вместо имени связанной переменной будем указывать число промежуточных лямбда-абстракций между связывающей абстракцией и переменной. Например,  $\lambda x. \lambda y. y x$  превратится в  $\lambda. \lambda. 0 \ 1$ .

Докажите, что  $A =_{\alpha} B$  тогда и только тогда, когда представления де Брауна для  $A$  и  $B$  совпадают. Сформулируйте правила (алгоритмы) для подстановки термов и бета-редукции для этого представления.

5. Как мы знаем,  $\Omega \rightarrow_{\beta} \Omega$ . А существуют ли такие лямбда-выражения  $A$  и  $B$  ( $A \neq_{\alpha} B$ ), что  $A \rightarrow_{\beta} B$  и  $B \rightarrow_{\beta} A$ ?

Решение.  $A = \omega (\lambda x. \omega x), B = (\lambda x. \omega x) (\lambda x. \omega x)$

□

6. Рассмотрим следующие лямбда-выражения для задания алгебраических типов:

| Обозначение | лямбда-терм                                  | название                 |
|-------------|--|--------------------------|
| $Case$      | $\lambda l. \lambda r. \lambda c. c \ l \ r$ | сопоставление с образцом |
| $InL$       | $\lambda l. (\lambda x. \lambda y. x \ l)$   | левая инъекция           |
| $InR$       | $\lambda r. (\lambda x. \lambda y. y \ r)$   | правая инъекция          |

Сопоставление с образцом — это функция от значения алгебраического типа и двух действий  $l$  и  $r$ , которая выполняет действие  $l$ , если значение создано «левым» конструктором, и  $r$  в случае «правого» конструктора. Иными словами,  $Case \ l \ r \ c$  — это аналог `case c { InL x -> l x; InR x -> r x }`.

Заметим, что список (например, целых чисел) — это алгебраический тип:

`List = Nil | Cons Integer List.`

Можно сконструировать значение данного типа: `Cons 3 (Cons 5 Nil)`. Можно, например, вычислить его длину:

`length Nil = 0`  
`length (Cons _ tail) = length tail + 1`

Определим  $Nil = InL \ 0$ , а  $Cons \ a \ b = InR \ (MkPair \ a \ b)$ . Заметим, что теперь списки могут быть напрямую перенесены в лямбда выражения.

Определите следующие функции в лямбда-исчислении для списков:

(a) вычисление длины списка;

$$Y \ \lambda f. \lambda l. Case \ (\lambda x. 0) \ (\lambda p. (+1) \ (f \ (PrR \ p))) \ l$$

(b) построение списка длины  $n$  из элементов  $0, 1, 2, \dots, n-1$ ;

$$\lambda n. (Y \ \lambda f. (\lambda n'. \lambda m. (Eq \ n' \ m) \ Nil \ (Cons \ m \ (f \ n' \ (inc \ m))))) \ n \ 0$$

(c) разворот списка: из списка  $a_1, a_2, \dots, a_n$  сделать список  $a_n, a_{n-1}, \dots, a_1$ ;

$$Add = \lambda e. Y \ \lambda f. Case \ (\lambda x. Cons \ e \ Nil) \ (\lambda p. Cons \ (PrL \ p) \ (f \ (PrR \ p)))$$

$$Reverse = Y \ \lambda f. Case \ (\lambda x. Nil) \ (\lambda p. Add \ (PrR \ p) \ (f \ (PrL \ p)))$$

(d) функцию высшего порядка  $map$ , которая по функции  $f$  и списку  $a_1, a_2, \dots, a_n$  строит список  $f(a_1), f(a_2), \dots, f(a_n)$ .

$$\lambda g. Y \ \lambda f. Case \ (\lambda x. Nil) \ (\lambda p. Cons \ (g \ (PrL \ p)) \ (f \ (PrR \ p)))$$

Решением задачи является полный текст соответствующего лямбда-выражения с объяснениями механизма его работы. Используйте интерпретатор лямбда-выражений *lci* или аналогичный для демонстрации результата.

7. Чёрчевские нумералы соответствуют натуральным числам в аксиоматике Пеано.

- (a) Предложите «двоичные нумералы» — способ кодирования чисел, аналогичный двоичной системе (такой, при котором длина записи числа соответствует логарифму числового значения).

$$\lambda n. \text{Reverse } ((Y \lambda f. \lambda r. (\text{IsZero } r) \text{ Nil } (\text{Cons } ((\text{IsEven } r) 0 1) (f (\text{DivBy2 } r)))) n)$$

- (b) Предложите реализацию функции (+1) в данном представлении.

```
BinaryPlus1 = \n.Reverse(
  (Y \f.\l.\r.Case
    (\x.
      (IsZero r)
      Nil
      (Cons 1 Nil))
    (\p.
      (IsZero r)
      (Cons (PrL p) (f (PrR p) 0))
      ((IsZero (PrL p))
        (Cons 1 (f (PrR p) 0))
        (Cons 0 (f (PrR p) 1))))
  1)
  (Reverse n)
  1);
```

- (c) Предложите реализацию лямбда-выражения преобразования числа из двоичного нумерала в чёрчевский.

```
ToChurch = \b.
(Y \f.\l.\s.Case
  (\x.s)
  (\p.f
    (PrR p)
    (Plus (Plus s s) (PrL p)))
  1)
b 0;
```

Аналогично прошлому заданию, решение должно содержать полный код лямбда-выражения вместе с объяснением механизма его работы.