

Haskell. Домашнее задание №1

Горбунов Егор Алексеевич

4 октября 2015 г.

1 Долг с пары

Задача №1.1 (функция *or*)

$$or\ x\ y = x\ true\ y$$

Если $x = true = \lambda ab.a$, то $or\ x\ y$ вернёт $true$ (т.е. x), иначе будет возвращён y , который и будет верным ответом, т.к. теперь всё от него и зависит (если $y = true$, то $or\ x\ y = true$ и с $false$ аналогично).

2 Примитивная рекурсия

$$prim_rec\ b\ z\ n = snd\ (n\ (\lambda p.pair\ (succ\ (fst\ p))(p\ b))\ (pair\ \bar{0}\ z))$$

Суть: применяем $b\ n$ раз к z (каждое следующее применение производится к результату предыдущего). Исходя из этого будет в следующих заданиях строить требуемые функции.

Задача №2.1 (факториал)

$$fact\ n = prim_rec\ (\lambda xy.mult\ (succ\ x)\ y)\ \bar{1}\ n$$

Для $n = \bar{0}$ ($\bar{0} = zero$ — число Чёрча): $fact\ 0 = snd\ (pair\ \bar{0}\ \bar{1}) = \bar{1}$ (уд. определению факториала).

Для $n = \bar{1}$:

$$\begin{aligned} fact\ 1 &= snd\ (pair\ (succ\ (fst\ (pair\ \bar{0}\ \bar{1})))\ ((pair\ \bar{0}\ \bar{1})\ (\lambda xy.mult\ (succ\ x)\ y))) \\ &= snd\ (pair\ (succ\ \bar{0})\ (mult\ (succ\ \bar{0})\ \bar{1})) \\ &= \bar{1} \end{aligned}$$

Аналогично для $n > \bar{1}$. ■

Задача №2.2 (предыдущее число)

Идея: вернуть значение счётчика в примитивной рекурсии, только на единицу меньше, т.е. увеличивать результат столько же раз, сколько увеличивается счётчик, за исключением одного раза.

При применении абстракции b (параметр $prim_rec$) счётчик передаётся первым параметром.

Итого:

$$pred\ n = prim_rec\ (\lambda xy.(ifZero\ x)\ x\ (succ\ y))\ \bar{0}$$

$$n = \bar{0}: pred\ \bar{0} = \bar{0} = snd\ (pair\ \bar{0}\ \bar{0})$$

$$n = \bar{1}:$$

$$\begin{aligned} pred\ \bar{1} &= snd\ ((\lambda p.pair\ (succ\ (fst\ p)))(p\ (\lambda xy.(ifZero\ x)\ x\ (succ\ y))))\ (pair\ \bar{0}\ \bar{0}) \\ &= snd\ (pair\ (succ\ (fst\ (pair\ \bar{0}\ \bar{0})))((pair\ \bar{0}\ \bar{0})\ (\lambda xy.(ifZero\ x)\ x\ (succ\ y)))) \\ &= snd\ (pair\ (\bar{1})\ ((ifZero\ \bar{0})\ \bar{0}\ (succ\ \bar{0}))) \\ &= snd\ (pair\ \bar{1}\ \bar{0}) \\ &= \bar{0} \end{aligned}$$

) Для $n = \bar{1}$ верно. Легко проверить для $n = 2$: нужно применить абстракцию $(\lambda p.pair\ (succ\ (fst\ p)))(p\ (\lambda xy.(ifZero\ x)\ x\ (succ\ y)))$ к паре $(pair\ \bar{1}\ \bar{0})$ (такая пара получена после первого применения). Ясно что тогда, т.к. $\bar{1} \neq zero$, то в итоге получится пара $(pair\ \bar{2}\ \bar{1})$ и будет возвращён в качестве ответа её второй элемент: $\bar{1}$. Верно! Таким образом в конце k -ой итерации примитивной рекурсии имеем пару $pair\ \bar{k}\ \bar{k-1}$.

■

Задача №2.3 (меньше или равно)

Пользуясь тем, что $pred\ \bar{0} = \bar{0}$ и определением чисел Чёрча получаем:

$$leq\ a\ b = (ifZero\ (b\ pred\ a))\ true\ false$$

Т.е. b раз пытаемся вычесть 1 из a . Если число стало в итоге $\bar{0}$, то $a - b \leq 0 \Rightarrow a \leq b$ ■

Задача №2.4 (модуль разности)

Имея возможность узнать, какое из чисел больше, можем найти модуль разности вычитанием большего из меньшего:

$$dist\ a\ b = (leq\ a\ b)\ (a\ pred\ b)\ (b\ pred\ a)$$

■

3 Списки

$$nil = \lambda cx.x$$

$$cons\ a\ as = \lambda cx.c\ a\ (as\ c\ x)$$

Задача №3.1 (*isEmpty*)

$$isEmpty\ l = l\ (\lambda ht.false)\ true$$

Проверим для пустого списка *nil*: $isEmpty\ nil = nil\ (\lambda ht.false)\ true = \lambda cx.x\ (\lambda ht.false)\ true =_{\beta} true$. Ок. Любой непустой список $l = cons\ a_1(cons\ a_2(\dots(cons\ nil)\dots))$ представляется в виде:

$$l = \lambda cx.c\ a_1\ (T\ c\ x) = \tag{1}$$

$$= \lambda cx.c\ a_1\ (c\ a_2\ (T'\ c\ x)) = \tag{2}$$

$$= \lambda cx.c\ a_1\ (c\ a_2\ (\dots(c\ a_n\ (nil\ c\ x))\dots)) = \tag{3}$$

$$= \lambda cx.c\ a_1\ (c\ a_2(\dots(c\ a_n\ x)\dots)) \tag{4}$$

Ясно, что если вместо *c* подставить любую абстракцию с 2 связанными переменными, всегда возвращающую *false*, то $isEmpty\ l$ будет возвращать *false*. ■

Задача №3.2 (*head*)

$$head\ l = l\ true\ nil$$

Для *nil*: $head\ nil = nil\ true\ nil = (\lambda cx.x)\ true\ nil =_{\beta} nil$. Разумно.

Для непустого $l = cons\ a_1\ T$:

$$\begin{aligned} head\ l &= (\lambda cx.c\ a_1\ (c\ a_2(\dots(c\ a_n\ x)\dots))\ true\ nil =_{\beta} \\ &=_{\beta} true\ a_1\ (\dots) =_{\beta} \\ &=_{\beta} a_1 \end{aligned}$$

■

Задача №3.3 (*tail*)

Соберём хвост списка $l = [a_1, \dots, a_{n-1}, a_n]$ начиная с конца. Пускай:

$$c\ e\ p = pair\ (snd\ p)\ (cons\ e\ (snd\ p)))$$

Тогда, например $c\ a_n\ (pair\ nil\ nil) = pair\ nil\ l'$, где l' — лист из элемента a_n , т.е. в первом элементе пары хранится хвост списка l' из одного элемента. Таким образом последовательные применения *c* при построении списка (см. формулу 4) приведут к тому, что после последнего применения *c* получится пара $pair\ tail\ l$. Таким образом ответ на задачу:

$$tail\ l = fst\ (l\ (\lambda ep.pair\ (snd\ p)\ (cons\ e\ (snd\ p))))\ (pair\ nil\ nil))$$

■

Задача №3.4 (*append*)

Аналогично операции *tail* с одним изменением:

$$\text{append } l \ a = \text{snd } (l \ (\lambda ep.\text{pair } (\text{snd } p) \ (\text{cons } e \ (\text{snd } p)))) \ (\text{pair } \text{nil } a))$$

Таким образом после первого применения *c* (см. формулу 4): $c \ a_n \ (\text{pair } \text{nil } a) = \text{pair } a \ (\text{cons } a_n \ a)$. Итого во втором элементе пары у нас будет записан исходный список *l*, с добавленным в конец элементом *a*. ■

4 Y-комбинатор

Задача №4.1 (про $F: \forall M (FM = F)$)

Равенство $F = FM$ β -эквивалентно следующему: $F = (\lambda fm.f)F$. Видим, что *F* — неподвижная точка терма $\lambda fm.f$. *Y*-комбинатор по определению таков, что: $\forall G: (YG = G(YG))$, но тогда $F = YG$, где $G = \lambda fm.f$, итого:

$$F = Y \ (\lambda fm.f)$$

■

Задача №4.2 (*fact* через *Y*)

Определим факториал рекурсивно:

$$\text{fact} = \lambda n. (\text{isZero } n) \ 1 \ (\text{mult } n \ (\text{fact } (\text{pred } n)))$$

Теперь абстрагируемся:

$$\text{fact} = \overbrace{(\lambda fn. (\text{isZero } n) \ 1 \ (\text{mult } n \ (f \ (\text{pred } n))))}^{\text{fact}'} \ \text{fact}$$

fact' — неподвижная точка терма *fact*, таким образом:

$$\text{fact} = Y \ \text{fact}'$$

■

Задача №4.3 (функция Аккермана)

Функция Аккермана:

$$A(n, m) = \begin{cases} n + 1 & m = 0 \\ A(m - 1, 1) & m > 0 \wedge n = 0 \\ A(m - 1, A(m, (n - 1))) & m > 0 \wedge n > 0 \end{cases}$$

Зададим в терминах λ -исчисления (абстрагируемся сразу, как в предыдущем задании):

$$A = \overbrace{(\lambda f n m. (isZero\ m) (succ\ n) ((isZero\ n) (f\ (pred\ m)\ \bar{1}) (f\ (pred\ m)\ (f\ m\ (pred\ n))))})^{A'} A$$

Тут $\bar{1}$ — число Чёрча, соответствующее 1.

Аналогично предыдущей задаче: $A = Y A'$. ■