Функциональное программирование. HW#2

Тураев Тимур, 504 (SE)

1. *Приведите пример замкнутого чистого λ-терма, находящегося*
– в WHNF, но не в HNF
Не-редекс на верхнем уровне, но внутри чистый редекс

$$\lambda\mathtt{x}.(\lambda\mathtt{y}.\mathtt{y})\mathtt{x}$$

– в HNF, но не в NF
Не-редекс на верхнем уровне, но внутри не чистый редекс, но можно провести β-редукцию

$$\lambda\mathtt{x}.\mathtt{x}((\lambda\mathtt{y}.\mathtt{y})\mathtt{x})$$

2. *Написать* `minus`*,* `equals`*,* `lt`*,* `gt`*,* `le`*,* `ge`

$$\mathtt{minus} \equiv \lambda\mathtt{mn}.\mathtt{n}\,\mathtt{pred}\,\mathtt{m}$$

$$\mathtt{equals} \equiv \lambda\mathtt{mn}.\mathtt{AND}(\mathtt{iszro}(\mathtt{minus}\,\mathtt{m}\,\mathtt{n}))(\mathtt{iszro}(\mathtt{minus}\,\mathtt{n}\,\mathtt{m}))$$

$$\mathtt{le} \equiv \lambda\mathtt{mn}.\mathtt{iszro}(\mathtt{minus}\,\mathtt{m}\,\mathtt{n})$$

$$\mathtt{gt} \equiv \lambda\mathtt{mn}.\mathtt{NOT}(\mathtt{le}\,\mathtt{m}\,\mathtt{n})$$

$$\mathtt{ge} \equiv \lambda\mathtt{mn}.\mathtt{AND}(\mathtt{gt}\,\mathtt{m}\,\mathtt{n})(\mathtt{equals}\,\mathtt{m}\,\mathtt{n})$$

$$\mathtt{lt} \equiv \lambda\mathtt{mn}.\mathtt{NOT}(\mathtt{ge}\,\mathtt{m}\,\mathtt{n})$$

3. *Построить функцию* `sum`*, суммирующую элементы списка и функцию* `length`*, вычисляющую длину списка.*

$$\mathtt{sum} \equiv \lambda\mathtt{l}.\mathtt{l}(\lambda\mathtt{ht}.\mathtt{plus}\,\mathtt{h}\,\mathtt{t})\mathtt{0}$$

$$\mathtt{length} \equiv \lambda\mathtt{l}.\mathtt{l}(\lambda\mathtt{ht}.\mathtt{plus}\,\mathtt{1}\,\mathtt{t})\mathtt{0}$$

Пример `sum`:

$$\begin{aligned}
sum\ [2,3] &= \\
(\lambda l.l(\lambda ht.plus\ h\ t)0)\ [2,3] &= \\
(\lambda l.l(\lambda ht.plus\ h\ t)0)\ (\lambda cn.c\ 2\ (c\ 3\ n)) &= \\
(\lambda cn.c\ 2\ (c\ 3\ n))(\lambda ht.plus\ h\ t)0 &= \\
(\lambda ht.plus\ h\ t)\ 2((\lambda ht.plus\ h\ t)\ 3\ 0) &= \\
(\lambda ht.plus\ h\ t)\ 2(plus\ 3\ 0) &= \\
(\lambda ht.plus\ h\ t)\ 2\ 3 &= \\
plus\ 2\ 3 &= 5
\end{aligned}$$

Пример `length`. Видно, что это ни что иное, как сумма, которая игнорирует список и складывает единицы:

$$
\begin{aligned}
length;[2,3] &= \\
(\lambda l.l(\lambda ht.plus\ 1\ t)0)\ [2,3] &= \\
(\lambda l.l(\lambda ht.plus\ 1\ t)0)\ (\lambda cn.c\ 2\ (c\ 3\ n)) &= \\
(\lambda cn.c\ 2\ (c\ 3\ n))(\lambda ht.plus\ 1\ t)0 &= \\
(\lambda ht.plus\ 1\ t)\ 2((\lambda ht.plus\ 1\ t)\ 3\ 0) &= \\
(\lambda ht.plus\ 1\ t)\ 2(plus\ 1\ 0) &= \\
(\lambda ht.plus\ 1\ t)\ 2\ 1 &= \\
plus\ 1\ 1 &= 2
\end{aligned}
$$

4. *Построить функцию* `tail`*, возвращающую хвост списка.*

Используем ту же идею, что и с функцией `pred`.

Определим начальное значение (в `pred`-е это была пара (0, 0)) – в нашем случае это будет пара из двух пустых списоков:

$$\mathtt{xz} \equiv \mathtt{pair\ nil\ nil}$$

Далее, определим функцию, обрабатывающую наш список на каждой итерации. Смысл этой функции вот в чем: мы будем пересобирать список с конца, сохраняя его во втором элементе паре, в первом же элементе будем держать тот же список, но на предыдущей итерации. Таким образом, в конце получим, что в первом элементе паре будет лежать недособранный на один элемент список – то есть хвост.
Этой фунции на вход мы будем подавать очередной элемент списка и пару-состояние:

$$\mathtt{fs} \equiv \lambda \mathtt{ep.pair}\ (\mathtt{snd\ p})\ (\mathtt{cons\ e\ snd\ p})$$

Ну и, наконец, функция `tail`. Ее поведение уже описано выше.

$$\mathtt{tail} \equiv \lambda \mathtt{l.fst}(\mathtt{l\ fs\ xz})$$

Пример.

$$
\begin{aligned}
tail\ [5,3,2] &= \\
(\lambda l.fst(l\ fs\ xz))\ [5,3,2] &= \\
(\lambda l.fst(l\ fs\ xz))\ (\lambda cn.c\ 5\ (c\ 3\ (c\ 2\ n))) &= \\
fst((\lambda cn.c\ 5\ (c\ 3\ (c\ 2\ n)))\ fs\ xz) &= \\
fst(fs\ 5\ (fs\ 3\ (fs\ 2\ xz))) &= \\
fst(fs\ 5\ (fs\ 3\ (fs\ 2\ [\ nil\ nil]))) &= \\
fst(fs\ 5\ (fs\ 3\ ((\lambda ep.pair\ (snd\ p)\ (cons\ e\ snd\ p))\ 2\ [nil, nil]))) &= \\
fst(fs\ 5\ (fs\ 3\ [nil,[2]])) &= \\
fst(fs\ 5\ [[2],[3,2]]) &= \\
fst[[3,2],[5,3,2]] &= [3,2]
\end{aligned}
$$

5.1 *Построить терм-пожиратель, то есть такой терм* F*, что для любого терма* M *верно следующее:* $\mathtt{FM} = \mathtt{F}$

Пусть `F` выглядит как-то так: $\mathtt{F} = \mathtt{YX}$, где `X` пока неизвестный терм. Из определения fixpoint combinator известно, что $\mathtt{YX} = \mathtt{X}(\mathtt{YX})$.

Посмотрим что такое `FM`:

$$\mathtt{FM} = (\mathtt{YX})\mathtt{M} = (\mathtt{X}(\mathtt{YX}))\mathtt{M} = \mathtt{X}(\mathtt{YX})\mathtt{M} = \mathtt{XFM} = \mathtt{F}$$

Тогда что такое неизвестный терм `X`? Это такая функция, принимающая 2 параметра и возвращающая первый. Да это же `K`!

Answer: $\mathtt{F} = \mathtt{YK}$

5.2 *Построить такой терм* F*, что для любого терма* M *верно следующее:* $\mathtt{FM} = \mathtt{MF}$

(позволю себе скопировать :))

Пусть $\mathtt{F}$ выглядит как-то так: $\mathtt{F} = \mathtt{YX}$, где $\mathtt{X}$ пока неизвестный терм. Из определения fixpoint combinator известно, что $\mathtt{YX} = \mathtt{X(YX)}$.

Посмотрим что такое $\mathtt{FM}$:

$$\mathtt{FM} = \mathtt{(YX)M} = \mathtt{(X(YX))M} = \mathtt{X(YX)M} = \mathtt{XFM} = \mathtt{MF}$$

Тогда что такое неизвестный терм $\mathtt{X}$? Это такая функция, принимающая 2 параметра и возвращающая применение второго к первую. Ну запишем это: $\mathtt{X} = \lambda\mathtt{xy.yx}$

Answer: $\mathtt{F} = \mathtt{Y}(\lambda\mathtt{xy.yx})$

5.3 *Построить такой терм* F*, что для любых термов* M *и* N *верно следующее:* $\mathtt{FMN} = \mathtt{NF(NMF)}$

(позволю себе опять скопировать :))

Пусть $\mathtt{F}$ выглядит как-то так: $\mathtt{F} = \mathtt{YX}$, где $\mathtt{X}$ пока неизвестный терм. Из определения fixpoint combinator известно, что $\mathtt{YX} = \mathtt{X(YX)}$.

Посмотрим что такое $\mathtt{FMN}$:

$$\mathtt{FMN} = \mathtt{(YX)MN} = \mathtt{(X(YX))MN} = \mathtt{X(YX)MN} = \mathtt{XFMN} = \mathtt{NF(NMF)}$$

Решаем простейшее уравнение на термы и находим вид неизвестного терма $\mathtt{X}$. Решили: $\mathtt{X} = \lambda\mathtt{fmn.nf(nmf)}$

Answer: $\mathtt{F} = \mathtt{Y}(\lambda\mathtt{fmn.nf(nmf)})$

6. *Пусть имеется взаимно-рекурсивное определение функций* f *и* g*:* $\mathsf{f} = \mathsf{Ffg}$, $\mathsf{g} = \mathsf{Gfg}$*. Используя* Y*-комбинатор, найдите нерекурсивные определения этих функций*

Абстрагируемся:

$$\mathtt{f} = \mathtt{Ffg} = (\lambda\mathrm{n.F\ n\ g})\mathtt{f} = \mathtt{f}'\ \mathtt{f}$$

$$\mathtt{g} = \mathtt{Gfg} = (\lambda\mathrm{n.G\ f\ n})\mathtt{g} = \mathtt{g}'\ \mathtt{g}$$

Видно, что и f и g есть неподвижные точки для лямбд. Поэтому:

$$\mathtt{f} = \mathtt{Yf}'$$

$$\mathtt{g} = \mathtt{Yg}'$$

Функции стали не столько рекурсивные, сколько взаимно-определенными. Продолжаем абстрагироваться:

$$\mathtt{f} = \mathtt{Yf}' = \mathtt{Y}(\lambda\mathrm{n.F\ n\ g}) = \mathtt{Y}(\lambda\mathrm{n.F\ n}\ (\mathtt{Yg}')) = \mathtt{Y}(\lambda\mathrm{n.F\ n}\ (\mathtt{Y}(\lambda\mathrm{n.G\ f\ n}))) = \mathtt{Y}((\lambda\mathrm{mn.F\ n}\ (\mathtt{Y}(\lambda\mathrm{n.G\ m\ n})))\mathtt{f})$$

Обозначим, для краткости $\lambda\mathsf{mn.F\ n}\ (\mathsf{Y}(\lambda\mathsf{n.G\ m\ n})) = \mathsf{P}$. Продолжаем абстрагироваться:

$$\mathtt{f} = \mathtt{Y(Pf)} = (\lambda\mathrm{s}.\mathtt{Y(Ps)})\mathtt{f} = \mathtt{Y}(\lambda\mathrm{s}.\mathtt{Y(Ps)}) = \mathtt{Y}(\lambda\mathrm{s}.\mathtt{Y}((\lambda\mathrm{mn.F\ n}\ (\mathtt{Y}(\lambda\mathrm{n.G\ m\ n})))\mathtt{s}))$$

Все, записываем ответ

$$\mathtt{f} = \mathtt{Y}(\lambda\mathtt{s.Y}((\lambda\mathtt{mn.F\ n}\ (\mathtt{Y}(\lambda\mathtt{t.G\ m\ t})))\mathtt{s}))$$

Для функции g все аналогично:

$$\mathtt{g} = \mathtt{Y}(\lambda\mathtt{s.Y}((\lambda\mathtt{mn.G}\ (\mathtt{Y}(\lambda\mathtt{t.F\ t\ n}))\ \mathtt{m})\mathtt{s}))$$