

Домашнее задание №7: «типовая система Хиндли-Милнера»

1. *О выразительной силе НМ.* Заметим, что список — это «параметризованные» числа в аксиоматике Пеано. Число — это длина списка, а к каждому штриху мы присоединяем какое-то значение. Операции добавления и удаления элемента из списка — это операции прибавления и вычитания единицы к числу.

Рассмотрим тип «бинарного списка» (расширение Окамля):

```
type 'a bin_list = Nil | Zero of (('a*'a) bin_list) |
  One of 'a * (('a*'a) bin_list);;
```

и операцию добавления элемента к списку:

```
let rec add elem lst = match lst with
  Nil -> One (elem, Nil)
| Zero tl -> One (elem, tl)
| One (hd, tl) -> Zero (add (elem, hd) tl)
```

- (а) Какой тип имеет add (обратите внимание на ключевое слово rec: для точного указания соответствующего лямбда-выражения и вывода типа необходимо использовать Y-комбинатор)? Считайте, что семейство типов bin_list 'a предопределено и обозначается как τ_α . Выразим ли этот тип в системе Хиндли-Милнера?
 - (б) Реализуйте предложенный тип и функцию add на Хаскеле (используйте опцию RankNTypes). Также реализуйте функцию для удаления элемента списка (головы).
 - (с) Предложите функцию для эффективного соединения двух списков (источник для вдохновения — сложение двух чисел в столбик).
 - (д) Предложите функцию для эффективного выделения n -го элемента из списка.
2. На занятии мы рассмотрели функцию strange_pair $x = (x\ 1, x\ "a")$. Покажите, что данную функцию невозможно типизировать в типовой системе Хиндли-Милнера. Указания: (а) ограничение мономорфизма отношения к делу не имеет; (б) ограничение на правило введения квантора всеобщности может оказаться существенным.
 3. Покажем, что алгоритм W действительно находит корректный тип для лямбда-выражения (доказательство, что он находит наиболее общий тип, мы оставим в стороне). Для этого докажем по индукции, что $W(\Gamma, X)$ действительно находит такие тип τ и подстановку S , что $S\Gamma \vdash X : \tau$:

- (а) покажите базу индукции: $W(\Gamma, x)$;

$$\frac{\Gamma, x : \forall \alpha_1 \dots \alpha_n. \tau : x \vdash \forall \alpha_1 \dots \alpha_n. \tau}{\Gamma \vdash x : \forall \beta_1 \dots \beta_n. \tau}$$

(b) покажите случай аппликации: $W(\Gamma, P Q)$;

$$\begin{array}{c}
 \frac{}{W(\Gamma, P) = (\tau_p, S_1)} \\
 \frac{}{S_1 \Gamma \vdash P : \tau_p} \\
 \frac{}{S_{21} \Gamma \vdash P : S_2 \tau_p} \\
 \frac{}{S_{321} \Gamma \vdash P : S_3(S_2 \tau_p)} \quad \frac{}{W(S_1 \Gamma, Q) = (\tau_q, S_2)} \\
 \frac{}{S_{321} \Gamma \vdash P : S_3(\tau_q \rightarrow \gamma)} \quad \frac{}{S_{21} \Gamma \vdash Q : \tau_q} \\
 \frac{}{S_{321} \Gamma \vdash P : (S_3 \tau_q) \rightarrow S_3 \gamma} \quad \frac{}{S_{321} \Gamma \vdash Q : S_3 \tau_q} \\
 \hline
 (S_3 \circ S_2 \circ S_1) \Gamma \vdash P Q : S_3 \gamma
 \end{array}$$

(c) покажите случай лямбда-абстракции: $W(\Gamma, \lambda x. P)$;

$$\begin{array}{c}
 \frac{}{W(\Gamma \cup \{x : \tau_x\}, P) = (\tau_p, S_1)} \\
 \frac{}{S_1(\Gamma \cup \{x : \tau_x\}) \vdash P : S_1 \tau_p} \\
 \frac{}{S_1 \Gamma, x : S_1 \tau_x \vdash P : S_1 \tau_p} \\
 \frac{}{S_1 \Gamma \vdash \lambda x. P : S_1 \tau_x \rightarrow S_1 \tau_p} \\
 \frac{}{S_1 \Gamma \vdash \lambda x. P : S_1(\tau_x \rightarrow \tau_p)}
 \end{array}$$

(d) покажите случай let-выражения: $W(\Gamma, \text{let } x = P \text{ in } Q)$.

$$\begin{array}{c}
 \frac{}{W(\Gamma, P) = (\tau_p, S_1)} \quad \frac{}{W(S_1 \Gamma \cup \{x : \forall \{\alpha_i\}. \tau_p\}, Q) = (\tau_q, S_2), \{\alpha_i\} \in FV(\tau_p)} \\
 \frac{}{S_1 \Gamma \vdash P : S_1 \tau_p} \quad \frac{}{S_2(S_1 \Gamma \cup \{x : \forall \{\alpha_i\}. \tau_p\}) \vdash Q : \tau_q, \{\alpha_i\} \in FV(\tau_p)} \\
 \frac{}{S_{21} \Gamma \vdash P : S_{21} \tau_p} \quad \frac{}{S_2(S_1 \Gamma \cup \{x : S_{21} \tau_p\}) \vdash Q : \tau_q} \\
 \frac{}{S_{21} \Gamma, x : S_{21} \tau_p \vdash Q : \tau_q} \\
 \hline
 S_{21} \Gamma \vdash \text{let } x = P \text{ in } Q : \tau_q
 \end{array}$$

4. Покажите, что в Хаскеле выражается $Y : \forall \alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha$ и правило исключённого третьего $E : \alpha \vee \neg \alpha$.

`magic :: a`
`magic = magic`

`Y :: (a -> a) -> a`
`Y = magic`

`E :: Either a (a -> Void)`
`E = magic`

5. Возможно ли в C++ построить выражения с типами ранга два и выше (включая конструкции с темплейтами)? Приведите пример, если да.