Домашнее задание №3

Андрей Козлов

8 марта 2015 г.

1. (a) $(\alpha \to \beta) \to \alpha \to \beta$
   (b) $\alpha \to (\alpha \to \beta) \to \beta$
   (c) $\alpha \to \beta \to \beta$
   (d) $(\alpha \to \beta) \to ((\alpha \to \beta) \to \alpha) \to \beta$
   (e) Терм не типизируется.
   Рассмотрим предтерм $(xy)x$, пусть он является термом. Тогда $\exists \Gamma, \sigma\colon \Gamma \vdash (x(yx))\colon \sigma$.
   Тогда по лемме об инверсии правый подтерм x имеет некий тип τ, а левый подтерм xy тип $\tau \to \sigma$, то есть $y\colon \alpha, x\colon \alpha \to \tau \to \sigma$. Таким образом, тип $\tau = \alpha \to \tau \to \sigma$ является подвыражением себя, что невозможно в силу конечности типа.

2. (a) $\lambda fg.f(\lambda a.gaa)$, $\lambda fg.f(g(f(\lambda a.gaa)))$, $\lambda fg.f(g(f(g(f(\lambda a.gaa)))))$
   (b) $\lambda fg.g(f(\lambda a.gaa))(f(\lambda a.gaa))$
   (c) $\lambda fg.g(\lambda a.fa)$
   (d) $\lambda f.f(\lambda g_1.g_1(\lambda a.f(\lambda g_2.a)))$

3. (a)
```
false :: Nat -> Bool -> Bool
false n _ = False

isZero :: Nat -> Bool -> Bool
isZero n _ = rec True false n

ge :: Nat -> Nat -> Bool
ge n m = rec True isZero (minus m n)
```
   (b)
```
multSucc :: Nat -> (Nat -> Nat)
multSucc n = mul (succ n)
```

```
fac :: Nat -> Nat
fac = rec 1 multSucc
```

(c)
```
ack :: Nat -> Nat-> Nat
ack = rec succ (\n r -> iter (\ a b -> r b))

iter :: (Nat -> Nat -> Nat) -> Nat -> Nat
iter s n = rec (s 0 1) s n
```

4. (a)
   - $Pair_{\sigma,\tau}$
   - $\Gamma \vdash pair_{\sigma,\tau}: \sigma \to \tau \to Pair_{\sigma,\tau}$
   - $\Gamma \vdash fst_{\sigma}: Pair_{\sigma,\tau} \to \sigma$
     $\Gamma \vdash snd_{\tau}: Pair_{\sigma,\tau} \to \tau$
   - $\Gamma \vdash$ `fst (pair x y)` $\to$ `x`
     $\Gamma \vdash$ `snd (pair x y)` $\to$ `y`

   (b)
   - $List_{\sigma}$
   - $\Gamma \vdash nil_{\sigma}: List_{\sigma}$
     $\Gamma \vdash cons_{\sigma}: \sigma \to List_{\sigma} \to List_{\sigma}$
   - $\Gamma \vdash rec_{List_{\sigma}}: \alpha \to (\sigma \to List_{\sigma} \to \alpha \to \alpha) \to List_{\sigma} \to \alpha$
   - $\Gamma \vdash$ `rec` $_{List_{\sigma}}$ `n c nil` $\to$ `n`
     $\Gamma \vdash$ `rec` $_{List_{\sigma}}$ `n c (cons x xs)` $\to$ `c x xs (rec n c xs)`

5.
```
recList :: a -> (b -> [b] -> a -> a) -> [b] -> a
recList n c [] = n
recList n c (x:xs) = c x xs $ recList n c xs

insert :: Ord a => a -> [a] -> [a] -> [a]
insert x _ [] = [x]
insert x _ (y:ys) | x <= y = (x:y:ys)
                  | otherwise = y:(insert x [] ys)

sort :: Ord a => [a] -> [a]
sort = recList [] insert
```