Домашнее задание №3

Андрей Козлов

27 февраля 2015 г.

1. (a) $\alpha \to \alpha$

   (b) $\alpha \to (\alpha \to \beta) \to \beta$

   (c) $\alpha \to \alpha \to \alpha$

   (d) $(\alpha \to \beta) \to ((\beta \to \alpha) \to \alpha) \to \beta$

   (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.

3. (a)

```
isZero :: Nat -> Bool
isZero 0 = true
isZero (suc n) = false

ge :: Nat -> Nat -> Bool
ge n m = if (isZero (minus m n))
         then true
         else false
```

   (b)

```
fac :: Nat -> Nat
fac 0 = suc 0
fac (suc n) = mul (suc n) (fac n)
```

   (c)

```
f :: (Nat -> a) -> Nat -> a
f g 0 = g (suc 0)
```

```
f g (suc n) = g (f g n)

ack :: Nat -> Nat -> Nat
ack 0 = suc
ack (suc m) = f (ack m)
```

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

   (b)
   - $List_{\sigma}$
   -
     - $\Gamma \vdash nil_{\sigma} \colon List_{\sigma}$
     - $\Gamma \vdash cons_{\sigma} \colon \sigma \to List_{\sigma} \to List_{\sigma}$
   - $\Gamma \vdash rec_{List_{\sigma}} \colon \alpha \to (\sigma \to List_{\sigma} \to \alpha) \to List_{\sigma} \to \alpha$
   -
     - $\Gamma \vdash$ `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.
```
insert :: Int -> [Int] -> [Int]
insert x [] = [x]
insert x (y:ys) = if x <= y
         then (x:y:ys)
         else y:(insert x ys)

sort :: [Int] -> [Int]
sort [] = []
sort (x:xs) = insert x (sort xs)
```