

Języki i paradygmaty programowania (Info, III rok) 16/17

Kokpit ► Moje kursy ► JiPP.INFO.III.16/17 ► 24-30.4 ► Lab Typy

Lab Typy

Napisz kontrolę typów dla rachunku lambda z prostymi typami. Możliwe rozszerzenia: let, unifikacja, rekonstrukcja typów,

Jako składni abstrakcyjnej można użyć na przykład

```

module IntLambda where

infixr 5 :->
data Type = TInt | Type :-> Type
    deriving Eq

type Name = String
data Exp = EInt Int | EVar Name
    | ELam Name Type Exp | EApp Exp Exp

-- Przykładowe lambda-termny
type Exp1 = Type -> Exp
type Exp2 = Type -> Exp1
type Exp3 = Type -> Exp2

int :: Type
int = TInt

mkI :: Exp1
mkI a = ELam "x" a $ EVar "x"

mkK :: Exp2
mkK a b = ELam "x" a $ ELam "y" b $ EVar "x"

intK = mkK int int

mkS :: Exp3
mkS a b c = ELam "x" a $ ELam "y" b $ ELam "z" c
    $ EApp
    (EApp (EVar "x") (EVar "z"))
    (EApp (EVar "y") (EVar "z"))

intS = mkS (int:->int:->int) (int:->int) int

-- kombinator omega nie typuje sie w prostym rachunku lambda
mkOmega :: Exp1
mkOmega t = ELam "x" t $ EApp (EVar "x") (EVar "x")

intOmega = mkOmega TInt

-----
-- Koniec IntLambda
-----

```

W pierwszej wersji można użyć po prostu `error` do raportowania błędów, potem jednak lepiej zrobić lepsze raportowanie, np z `ErrorT`

Przykładowa sesja:

```
*Main> typeCheck intK
int -> int -> int

*Main> typeCheck intS
(int -> int -> int) -> (int -> int) -> int -> int

*Main> typeCheck intOmega
Error:
Type error in
\ (x:int).x x
In expression
x x
The type of x: int is not a function type

*Main> typeCheck (EApp intS intK)
(int -> int) -> int -> int

*Main> typeCheck $ EApp (EApp intS intK) intK
Error:
Type error in
(\ (x:int -> int -> int). \ (y:int -> int). \ (z:int).x z (y z)) (\ (x:int). \ (y:int).x) (\ (x:int). \ (y:int).x)
For expression:
\ (x:int). \ (y:int).x
Cannot match expected type
int -> int
against inferred
int -> int -> int
```

Ostatnia modyfikacja: środa, 9 kwiecień 2014, 18:48

NAWIGACJA



Kokpit

■ Strona główna

Strony

Moje kursy


JNPI.INFO.II.16/17

JiPP.INFO.III.16/17

Uczestnicy

 Odznaki

 Kompetencje

 Oceny

Główne składowe

T1 27.2-5.3 Haskell 1

T2 6-12.3 Haskell 2

T3 13-19.3 Monady 1

20-26.3 Monady 2

27.3-2.4 Składnia 1