Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Lukas Stevens

Lambda Calculus
Winter Term 2021/22
Exercise Sheet 09

Exercise 1 (Recursive `let`)

Recursive `let` expressions are one way (besides Y-combinators) to add recursion to $\lambda^{\to}$.

$$t ::= x \mid (t_1\ t_2) \mid (\lambda x.\ \ t) \mid \texttt{letrec}\ x = t_1\ \texttt{in}\ t_2$$

a) Modify the standard typing rule for `let` to create a suitable rule for `letrec`.

b) Considering *type inference*, what is the problematic property of this rule compared to the rule for `let`?

Exercise 2 (Type Inference in Haskell (2))

Extend the implementation of the type inference algorithm from the last exercise with `let` and `letrec` constructs.

Exercise 3 (Peirce's Law in Intuitionistic Logic)

Prove the following variant of Peirce's Law in inuitionistic logic:

$$((((P \to Q) \to P) \to P) \to Q) \to Q$$

Homework 4 (Fixed-point combinator)

Let

$$\$ = \lambda abcdefghijklmnopqstuvwxyzr.\ r(thisisafixedpointcombinator)$$

and

$$€ = \$\ .$$

Show that € is a fixed-point combinator.

Homework 5 (`let`-Polymorphism)

Give a derivation tree for the following statement, and so determine the type τ:

$$[z\colon \tau_0] \vdash \texttt{ let } x = \lambda y\ z.\ z\ y\ y \texttt{ in } x\ (x\ z)\ :\tau$$

Homework 6 (Constructive Logic)

a) Prove the following statement using the calculus for intuitionistic propositional logic:

$$((c \to b) \to b) \to (c \to a) \to ((a \to b) \to b)$$

Hint: To make your proof tree more compact, you may remove unneeded assumptions to the left of the $\vdash$ during the proof as you see fit. For example, the following step is valid:

$$\frac{p \vdash p}{p, q \vdash p}$$

b) Give a well-typed expression in $\lambda^{\to}$ with the type

$$((\gamma \to \beta) \to \beta) \to (\gamma \to \alpha) \to ((\alpha \to \beta) \to \beta)$$

(You don't need to give the derivation tree.)