Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Johannes Hölzl and Lars Hupel

Equational Logic
Summer Term 2015
Solutions to Exercise Sheet 12

Exercise 60 (Type Inference)

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

$$[] \vdash \lambda xyz.\ x\ y\ (y\ z)\ : \tau$$

Solution

Abbreviations:

$$\rho = (\alpha \to \beta) \to (\beta \to \gamma)$$
$$\sigma = (\alpha \to \beta) \to (\alpha \to \gamma)$$
$$\Gamma = [x : \rho, y : \alpha \to \beta, z : \alpha]$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash y : \alpha \to \beta \qquad \Gamma \vdash x : \rho}{\Gamma \vdash xy : \beta \to \gamma} \qquad \dfrac{\Gamma \vdash y : \alpha \to \beta \qquad \Gamma \vdash z : \alpha}{\Gamma \vdash yz : \beta}}{[x : \rho, y : \alpha \to \beta, z : \alpha] \vdash xy(yz) : \gamma}}{[x : \rho, y : \alpha \to \beta] \vdash \lambda z.\, xy(yz) : \alpha \to \gamma}}{[x : \rho] \vdash \lambda yz.\, xy(yz) : \sigma}}{[] \vdash \lambda xyz.\, xy(yz) : \rho \to \sigma}$$

Exercise 61 (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) Give a derivation tree for the following statement, and so determine the type τ:

$$[] \vdash\ \texttt{letrec}\ x = \lambda y.\ x\ (x\ y)\ \texttt{in}\ x\ x : \tau$$

Solution

a) The rule for `letrec` is like the rule for `let`, but we also add x to Γ when checking t_1.

$$\dfrac{\Gamma[x : \sigma_1] \vdash t_1 : \sigma_1 \qquad \Gamma[x : \sigma_1] \vdash t_2 : \sigma_2}{\Gamma \vdash (\texttt{letrec}\ x = t_1\ \texttt{in}\ t_2) : \sigma_2}\ \textsc{LetRec}$$

Alternatively, we can combine this rule with the ∀-intro typing rule:

$$\frac{\{\alpha_1 \dots \alpha_n\} = FV(\tau) \setminus FV(\Gamma) \qquad \Gamma[x : \forall\alpha_1 \dots \alpha_n.\tau] \vdash t_1 : \tau \qquad \Gamma[x : \forall\alpha_1 \dots \alpha_n.\tau] \vdash t_2 : \tau_2}{\Gamma \vdash \texttt{letrec}\ x = t_1\ \texttt{in}\ t_2 : \tau_2}\ \textsc{LetRec'}$$

b) Abbreviations: $\Gamma_1 = [x : \forall\alpha.\ \alpha \to \alpha]$ and $\Gamma_2 = [x : \forall\alpha.\ \alpha \to \alpha, y : \alpha]$.

$$\dfrac{\dfrac{\dfrac{}{\Gamma_2 \vdash x : \alpha \to \alpha}\ \textsc{Var'} \qquad \dfrac{\dfrac{}{\Gamma_2 \vdash x : \alpha \to \alpha}\ \textsc{Var'} \quad \dfrac{}{\Gamma_2 \vdash y : \alpha}\ \textsc{Var'}}{\Gamma_2 \vdash x\ y : \alpha}\ \textsc{App}}{\Gamma_2 \vdash x\ (x\ y) : \alpha}\ \textsc{App}}{\Gamma_1 \vdash \lambda y.\ x\ (x\ y) : \alpha \to \alpha}\ \textsc{Abs}$$

$$\dfrac{\dfrac{\text{see above}}{\Gamma_1 \vdash \lambda y.\ x\ (x\ y) : \alpha \to \alpha} \qquad \dfrac{\dfrac{}{\Gamma_1 \vdash x : (\beta \to \beta) \to \beta \to \beta}\ \textsc{Var'} \quad \dfrac{}{\Gamma_1 \vdash x : \beta \to \beta}\ \textsc{Var'}}{\Gamma_1 \vdash x\ x : \beta \to \beta}\ \textsc{App}}{[] \vdash\ \texttt{letrec}\ x = \lambda y.\ x\ (x\ y)\ \texttt{in}\ x\ x : \beta \to \beta}\ \textsc{LetRec'}$$

Homework 62 (`let`-Polymorphism)

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

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

Homework 63 (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.)