Technische Universität München Institut für Informatik

 ${\bf Lambda~Calculus~Winter~Term~2022/23}$

Prof. Tobias Nipkow, Ph.D.

Lukas Stevens

Exercise Sheet 10

Exercise 1 (Example of Type Inference for let)

Consider the typing problem

$$x: \alpha \vdash \text{let } y = \lambda z. \ z \ x \ \text{in } y \ (\lambda v. \ x) : ?\tau$$

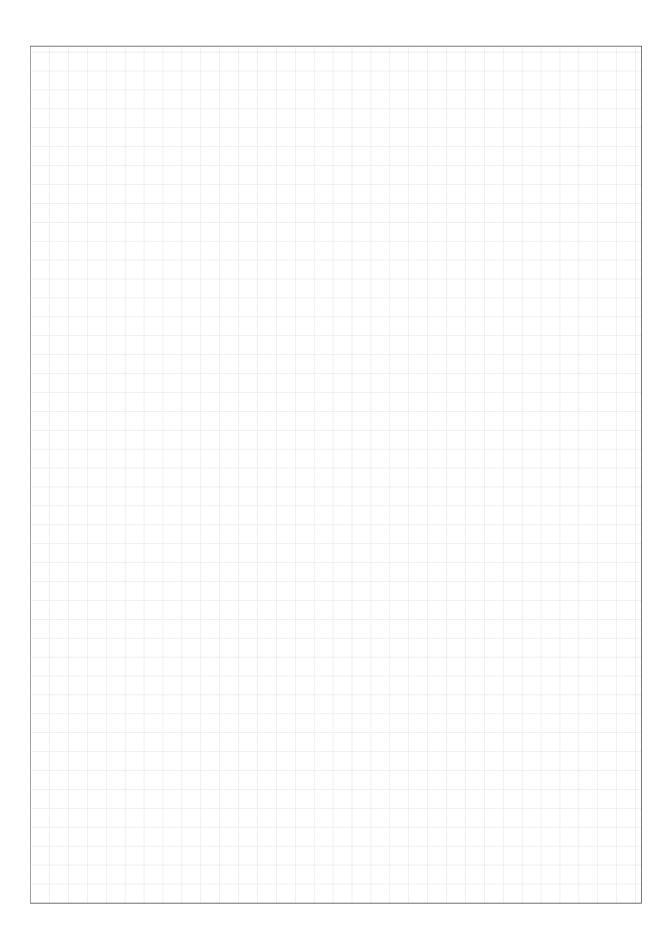
where α is a type variable.

- a) Find the most general type schema σ with $x: \alpha \vdash \lambda z. z x : \sigma$ and draw a type derivation tree.
- b) Draw the type derivation tree for

$$x: \alpha, y: \sigma \vdash y (\lambda v. x) : ?\tau$$

with the correct type for $?\tau$.



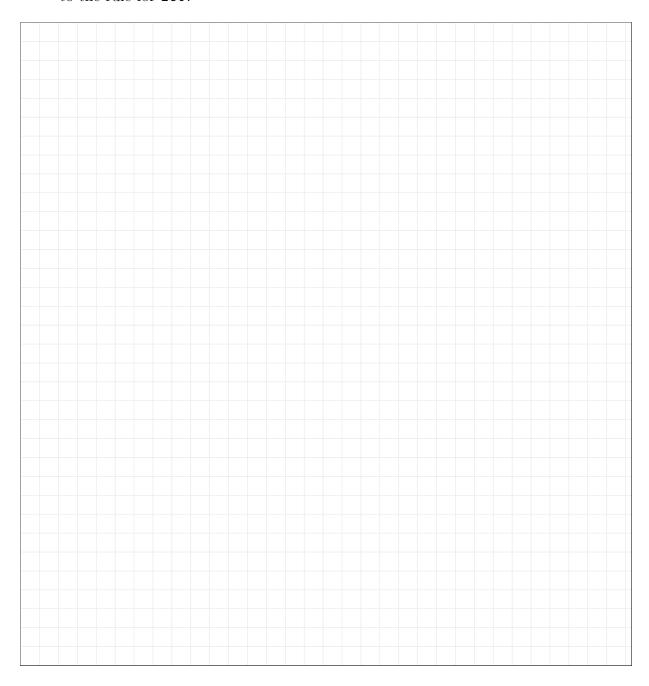


Exercise 2 (Recursive let)

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

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

- 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 3 (Type Inference in Haskell (2))

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

Homework 4 (Fixed-point combinator)

Let

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

and

Show that \in is a fixed-point combinator.

Homework 5 (let-Polymorphism)

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

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

Homework 6 (Towards Syntax-Directed let-Polymorphism)

In the lecture, it was claimed that the systems DM and DM', which, in contrast to DM, has explicit rules $\forall \text{Intro}$ and $\forall \text{Elim}$, are essentially equivalent. More specifically, it was claimed that

$$\Gamma \vdash_{DM} t : \sigma \Longrightarrow \exists \tau. \ \Gamma \vdash_{DM'} t : \tau \land \operatorname{gen}(\Gamma, \tau) \preceq \sigma.$$

As a step towards proving this result, we want to rearrange derivations in DM such that they resemble derivations in DM'. In particular, prove that

a) Any derivation $\Gamma \vdash_{DM} t$: σ can be transformed such that \forall Elim only occur in a chain below the Var rule, i.e.

$$\frac{\begin{array}{c} \Gamma \vdash x \colon \forall \alpha_1, \dots, \alpha_n. \ \tau \\ \vdots \\ \hline \Gamma \vdash x \colon \forall \alpha_n. \ \tau \\ \hline \underline{\Gamma \vdash x \colon \forall \alpha_n. \ \tau} \\ \hline \underline{\Gamma \vdash x \colon \tau} \\ \vdots \\ \hline \end{array}} \forall \text{Elim}$$

b) Any derivation $\Gamma \vdash_{DM} t$: σ can be transformed such that \forall Intro only occur in a chain that is terminated by an application of the Let rule or by the end of the proof, i.e.

$$\forall \text{Intro} \frac{\vdots}{\Gamma \vdash t_1 \colon \tau} \\ \forall \text{Intro} \frac{\Gamma \vdash t_1 \colon \forall \alpha_n. \ \tau}{\vdots} \\ \forall \text{Intro} \frac{\vdots}{\Gamma \vdash t_1 \colon \forall \alpha_1, \dots, \alpha_n. \ \tau} \quad \frac{\vdots}{\Gamma[x \colon \forall \alpha_1, \dots, \alpha_n. \ \tau] \vdash t_2 \colon \sigma} \\ \Gamma \vdash \text{let} \ x = t_1 \text{ in } t_2 \colon \sigma} \text{Let}$$

or

$$\frac{\vdots}{\begin{array}{c} \Gamma \vdash t_1 \colon \tau \\ \hline \Gamma \vdash t_1 \colon \forall \alpha_n. \ \tau \\ \hline \vdots \\ \hline \Gamma \vdash t_1 \colon \forall \alpha_1, \dots, \alpha_n. \ \tau \end{array}} \forall \text{Intro}$$