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

Exercise 1 (Example of Type Inference for let)

Consider the typing problem

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

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

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

with the correct type for $?\tau$.



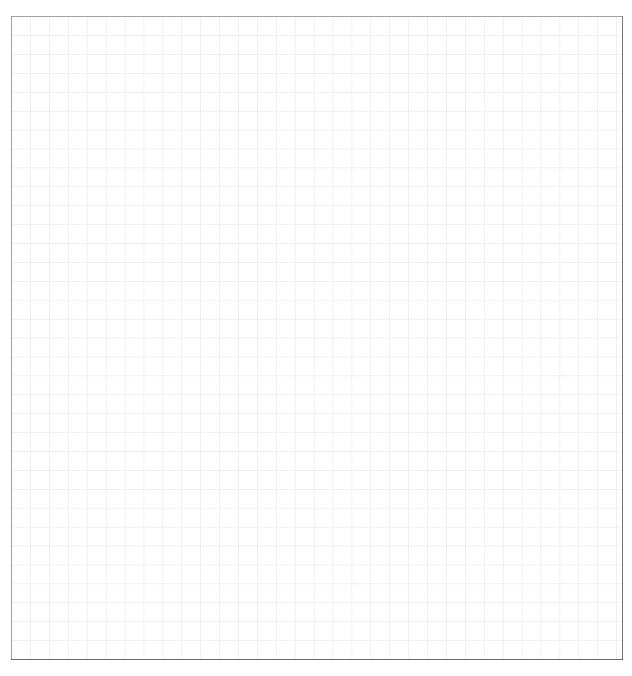
		_					 								
															+
															-+
															-+
															-+
															-+

Exercise 2 (Recursive let)

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

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

- 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 3 (Type Inference in Haskell (2))

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

You can find a template here.

Homework 4 (Fixed-point combinator)

Let

 $= \lambda abcdefghijklmnopqstuvwxyzr. r(this is a fixed point combinator)$

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 $x = \lambda y \ z. \ z \ y \ y \ in \ x \ (x \ z) : \tau$

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 Intro and \forall 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.

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

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.

$$\forall \text{Intro} \frac{ \overbrace{\Gamma \vdash t_1 : \tau}}{ \begin{array}{c} \hline{\Gamma \vdash t_1 : \tau} \\ \forall \text{Intro} \end{array}} \\ \forall \text{Intro} \frac{ \overbrace{\Gamma \vdash t_1 : \forall \alpha_n. \tau}}{ \overbrace{\Gamma \vdash t_1 : \forall \alpha_1, \dots, \alpha_n. \tau}} & \overbrace{\Gamma[x : \forall \alpha_1, \dots, \alpha_n. \tau] \vdash t_2 : \sigma} \\ \hline \\ \Gamma \vdash \text{let } x = t_1 \text{ in } t_2 : \sigma \end{array}$$
 Let

or

:	
$\Gamma \vdash t_1 \colon \tau$	\/Tratmo
$\Gamma \vdash t_1 \colon \forall \alpha_n.$	$\frac{\tau}{\tau} \forall Intro \\ \forall Intro$
:	VIIIIIO
$\Gamma \vdash t_1 \colon \forall \alpha_1, \ldots,$	$\overline{\alpha_n. \tau}$ \forall Intro