HW8 – Type Inference and Polymorphism

CS 476, Fall 2019
Due Nov. 25 at 2 PM

1 Instructions

This assignment is to be completed by hand (or in LaTeX if you know how to use it). Submit your answers as a PDF file via Gradescope. If you don’t have easy access to a scanner, you can use the one in SEO 1120, the main CS office – the staff will be happy to help you. As always, please don’t hesitate to ask for help on Piazza (https://piazza.com/class/jushbjmii13yb).

2 Type Inference Rules

The rules for type inference for a simple OCaml-like language are as follows.

\[
\begin{align*}
\vdash n : \text{int} & \quad \vdash b : \text{bool} \\
\vdash e_1 + e_2 : \text{int} & \quad \vdash e_1 \& e_2 : \text{bool} \\
\vdash e_1 = e_2 : \text{bool} & \\
\vdash x : \tau & \\
\vdash \text{fun} \ x : \tau \to e : \tau_1 \to \tau_2 &
\end{align*}
\]

When we add polymorphism, we add the following additional rules.

\[
\begin{align*}
\vdash e_1 : \tau_1 & \quad \text{fv}(\tau_1) - \text{fv}(\Gamma) = a_1, \ldots, a_n \\
\vdash x \mapsto \tau_1 & \quad \vdash e_2 : \tau \\
\vdash \text{let} \ x = e_1 \ \text{in} \ e_2 : \tau & \\
\vdash x : \forall a_1 \ldots a_n. \ \tau & \quad \text{fresh}
\end{align*}
\]

where \( [a_1 \mapsto b_1, \ldots, a_n \mapsto b_n] \tau \) means “replace the bound variables \( a_1 \) through \( a_n \) with the fresh type variables \( b_1 \) through \( b_n \).”
3 Problems

1. (3 points) What is the type of \( \text{fun} \ x \rightarrow \text{fun} \ y : \text{int} \rightarrow (x \ y) + 1 \)?

\[(\text{int} \rightarrow \text{int}) \rightarrow \text{int} \rightarrow \text{int} \]

2. (7 points) Write the proof tree for the judgment
\[
\{\} \vdash \text{fun} \ x \rightarrow \text{fun} \ y : \text{int} \rightarrow (x \ y) + 1) : \tau \ | \ C, \text{where } \tau \text{ and } C \text{ start as blank and are filled in as you complete the proof tree. You do not need to do unification.}
\]

\[
\begin{array}{l}
\{x : a, y : \text{int}\} \vdash x : a | \{\} \\
\{x : a, y : \text{int}\} \vdash y : \text{int} | \{\} \\
\{x : a, y : \text{int}\} \vdash (x \ y) + 1 : \text{int} | \{\tau = \text{int}, \text{int} = \text{int}, a = \text{int} \rightarrow \tau\} \\
\{x : a\} \vdash \text{fun} \ y : \text{int} \rightarrow (x \ y) + 1 : \text{int} \rightarrow \text{int} | \{\tau = \text{int}, \text{int} = \text{int}, a = \text{int} \rightarrow \tau\} \\
\{\} \vdash \text{fun} \ x \rightarrow \text{fun} \ y : \text{int} \rightarrow (x \ y) + 1 : a \rightarrow \text{int} \rightarrow \text{int} | \{\tau = \text{int}, \text{int} = \text{int}, a = \text{int} \rightarrow \tau\}
\end{array}
\]
3. Consider the program \( \text{let } g = (\text{fun } x -> (x = x)) \text{ in } g\ 5 \&\& g\ \text{true}. \)

(a) (5 points) Write a proof tree for the judgment \( \{\} \vdash (\text{fun } x -> (x = x)) : \tau \ | \ C. \) Again, \( \tau \) and \( C \) should start blank and be filled in as you complete the proof tree.

(b) (3 points) Given your answer to part a, what is the polymorphic type (polytype) of \( g \) in the program?

\[ \forall a.\ a \rightarrow \text{bool} \]
(c) (7 points) Write the proof tree for the judgment
\[ \{ \} \vdash \text{let } g = (\text{fun } x \rightarrow (x = x)) \text{ in } g \ 5 \ \&\& \ g \ \text{true} : \tau \ | \ C. \] Once again, \( \tau \) and \( C \) should start blank and be filled in as you complete the proof tree. You can use \( P_1 \) to stand for your proof tree from part a.

Hint 1: In the type context for the body of the let (i.e., \( g \ 5 \ \&\& \ g \ \text{true} \)), the type of \( g \) should be your answer to part b.

Hint 2: When you reach a variable, make sure to use the polymorphic variable rule from the end of section 2.

Let \( \Gamma \) be \( \{ g : \forall a \ a \rightarrow \text{bool} \}. \) Let \( C \) be \( \{ a = a, \tau_1 = \text{bool}, \tau_2 = \text{bool}, b \rightarrow \text{bool} = \text{int} \rightarrow \tau_1, c \rightarrow \text{bool} = \text{bool} \rightarrow \tau_2 \}. \)

\[
\begin{array}{c}
\Gamma \vdash g : b \rightarrow \text{bool} | \{ \} \\
\Gamma \vdash 5 : \text{int} | \{ \} \\
\Gamma \vdash g : c \rightarrow \text{bool} | \{ \} \\
\Gamma \vdash \text{true} : \text{bool} | \{ \} \\
\end{array}
\]

\[
\begin{array}{c}
\Gamma \vdash 5 : \tau_1 | \{ b \rightarrow \text{bool} = \text{int} \rightarrow \tau_1 \} \\
\Gamma \vdash g \ \text{true} : \tau_2 | \{ c \rightarrow \text{bool} = \text{bool} \rightarrow \tau_2 \} \\
\end{array}
\]

\[
\begin{array}{c}
\{ g : \forall a \ a \rightarrow \text{bool} \} \vdash g \ 5 \ \&\& \ g \ \text{true} : \text{bool} | \{ \tau_1 = \text{bool}, \tau_2 = \text{bool}, b \rightarrow \text{bool} = \text{int} \rightarrow \tau_1, c \rightarrow \text{bool} = \text{bool} \rightarrow \tau_2 \} \\
\{ \} \vdash \text{let } g = (\text{fun } x \rightarrow (x = x)) \text{ in } g \ 5 \ \&\& \ g \ \text{true} : \text{bool} | \ C \\
\end{array}
\]