CST Part II Types: Exercise Sheet

ML Polymorphism

Exercise 1. Here are some type checking problems, in the sense of Slide 7. Prove the following typings hold for the Mini-ML type system:

\[ \vdash \lambda x :: \text{nil} : \forall \alpha (\alpha \rightarrow \alpha \text{ list}) \]
\[ \vdash \lambda x (\text{case } x \text{ of } \text{nil } => \text{true } | x_1 :: x_2 => \text{false}) : \forall \alpha (\alpha \text{ list } \rightarrow \text{ bool}) \]
\[ \vdash \text{let } f = \lambda x_1 (\lambda x_2 (x_1)) \text{ in } f f : \forall \alpha_1, \alpha_2, \alpha_3 (\alpha_1 \rightarrow (\alpha_2 \rightarrow (\alpha_3 \rightarrow \alpha_2)))) \]

Exercise 2. Show that if \{ \} \vdash M : \sigma is provable, then M must be closed, i.e. have no free variables. [Hint: use rule induction for the rules on Slides 16–19 to show that the provable typing judgements, \( \Gamma \vdash M : \tau \), all have the property that \( \text{fv}(M) \subseteq \text{dom}(\Gamma) \).]

Exercise 3. Let \( \sigma \) and \( \sigma' \) be Mini-ML type schemes. Show that the relation \( \sigma \succ \sigma' \) defined on Slide 27 holds if and only if

\[ \forall \tau (\sigma' \succ \tau \Rightarrow \sigma \succ \tau) \]

[Hint: use the following property of simultaneous substitution:

\[ (\tau[\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n])[\bar{\tau}'/\bar{\alpha}'] = \tau[\tau_1[\bar{\tau}'/\bar{\alpha}']/\alpha_1, \ldots, \tau_n[\bar{\tau}'/\bar{\alpha}']/\alpha_n] \]

which holds provided the type variables \( \bar{\alpha} \)' do not occur in \( \tau \).]

Exercise 4. Try to augment the definition of \( pt \) on Slide 30 and in Figure 3 with clauses for \text{nil}, \text{cons}, and \text{case}-expressions.

Exercise 5. Suppose \( M \) is a closed expression and that \( (S, \sigma) \) is a principal solution for the typing problem \{ \} \vdash M : ? in the sense of Slide 27. Show that \( \sigma \) must be a principal type scheme for \( M \) in the sense of Slide 23.

Exercise 6. Show that if \( \Gamma \vdash M : \sigma \) is provable and \( S \in \text{Sub} \) is a type substitution, then \( S \Gamma \vdash M : S \sigma \) is also provable.

Polymorphic Reference Types

Exercise 7. Letting \( M \) denote the expression on Slide 33 and \{ \} the empty state, show that \( \langle M, \{ \} \rangle \rightarrow^* \text{FAIL} \) is provable in the transition system defined in Figure 4.

Exercise 8. Give an example of a Mini-ML \text{let}-expression which is typeable in the type system of Section 2.1, but not in the type system of Section 3.2 for Midi-ML with the value-restricted rule (letv).
Polymorphic Lambda Calculus

**Exercise 9.** Give a proof inference tree for (8) in Example 4.1.1. Show that

$$\forall \alpha_1 (\alpha_1 \to \forall \alpha_2 (\alpha_2)) \to \text{bool list}$$

is another possible polymorphic type for $\lambda f ((f \text{true}) :: (f \text{nil}))$.

**Exercise 10.** Show that if $\Gamma \vdash M : \tau$ and $\Gamma \vdash M : \tau'$ are both provable in the PLC type system, then $\tau = \tau'$ (equality up to $\alpha$-conversion). [Hint: show that $H \equiv \{ (\Gamma, M, \tau) \mid \Gamma \vdash M : \tau \land \forall \tau' (\Gamma \vdash M : \tau' \Rightarrow \tau = \tau') \}$ is closed under the axioms and rules on Slide 45.]

**Exercise 11.** In PLC, defining the expression \texttt{let x = M1 in M2} to be an abbreviation for $(\lambda x : \tau (M2)) M1$, show that the typing rule

$$
\frac{\Gamma \vdash M_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash M_2 : \tau_2}{\Gamma \vdash (\text{let} \ x = M_1 \ \text{in} \ M_2) : \tau_2}
$$

is admissible—in the sense that the conclusion is provable if the hypotheses are.

**Exercise 12.** The \textit{erasure}, $\text{erase}(M)$, of a PLC expression $M$ is the expression of the untyped lambda calculus obtained by deleting all type information from $M$:

$$
\text{erase}(x) \equiv x
$$

$$
\text{erase}(\lambda x : \tau (M)) \equiv \lambda x (\text{erase}(M))
$$

$$
\text{erase}(M_1 M_2) \equiv \text{erase}(M_1) \text{erase}(M_2)
$$

$$
\text{erase}(\Lambda \alpha (M)) \equiv \text{erase}(M)
$$

$$
\text{erase}(M \tau) \equiv \text{erase}(M).
$$

(i) Find PLC expressions $M_1$ and $M_2$ satisfying $\text{erase}(M_1) = \lambda x (x) = \text{erase}(M_2)$ such that $\vdash M_1 : \forall \alpha (\alpha \to \alpha)$ and $\vdash M_2 : \forall \alpha_1 (\alpha_1 \to \forall \alpha_2 (\alpha_1))$ are provable PLC typings.

(ii) We saw in Example 4.2.6 that there is a closed PLC expression $M$ of type $\forall \alpha (\alpha) \to \forall \alpha (\alpha)$ satisfying $\text{erase}(M) = \lambda f (f f)$. Find some other closed, typeable PLC expressions with this property.

(iii) [For this part you will need to recall, from the CST Part IB Foundations of Functional Programming course, some properties of beta reduction of expressions in the untyped lambda calculus.] A theorem of Girard says that if $\vdash M : \tau$ is provable in the PLC type system, then $\text{erase}(M)$ is strongly normalisable in the untyped lambda calculus, i.e. there are no infinite chains of beta-reductions starting from $\text{erase}(M)$. Assuming this result, exhibit an expression of the untyped lambda calculus which is not equal to $\text{erase}(M)$ for any closed, typeable PLC expression $M$.

**Exercise 13.** Prove the various typings and beta-reductions asserted in Example 4.4.4.
Exercise 14. Prove the various typings asserted in Example 4.4.5 and the beta-conversions on Slide 56.

Exercise 15. For the polymorphic product type $\alpha_1 \ast \alpha_2$ defined in the right-hand column of Figure 5, show that there are PLC expressions $Pair$, $fst$, and $snd$ satisfying:

\[
\begin{align*}
\{ \} & \vdash Pair : \forall \alpha_1, \alpha_2 (\alpha_1 \rightarrow \alpha_2 \rightarrow (\alpha_1 \ast \alpha_2)) \\
\{ \} & \vdash fst : \forall \alpha_1, \alpha_2 ((\alpha_1 \ast \alpha_2) \rightarrow \alpha_1) \\
\{ \} & \vdash snd : \forall \alpha_1, \alpha_2 ((\alpha_1 \ast \alpha_2) \rightarrow \alpha_2) \\
fst \alpha_1 \alpha_2(Pair \alpha_1 \alpha_2 x_1 x_2) & =_\beta x_1 \\
snd \alpha_1 \alpha_2(Pair \alpha_1 \alpha_2 x_1 x_2) & =_\beta x_2.
\end{align*}
\]

Exercise 16. [hard] Suppose that $\tau$ is a PLC type with a single free type variable, $\alpha$. Suppose also that $T$ is a closed PLC expression satisfying

\[
\{ \} \vdash T : \forall \alpha_1, \alpha_2 ((\alpha_1 \rightarrow \alpha_2) \rightarrow (\tau[\alpha_1/\alpha] \rightarrow \tau[\alpha_2/\alpha])).
\]

Define $\iota$ to be the closed PLC type

\[
\iota \overset{\text{def}}{=} \forall \alpha ((\tau \rightarrow \alpha) \rightarrow \alpha).
\]

Show how to define PLC expressions $R$ and $I$ satisfying

\[
\begin{align*}
\{ \} & \vdash R : \forall \alpha ((\tau \rightarrow \alpha) \rightarrow \iota \rightarrow \alpha) \\
\{ \} & \vdash I : \tau[\iota/\alpha] \rightarrow \iota \\
(R \alpha f)(I x) & \rightarrow^* f (T \iota \alpha (R \alpha f) x).
\end{align*}
\]