CST 2015-16 Part II Types – Exercise Sheet

ML Polymorphism

Exercise 1. Prove the following typings hold for the Mini-ML type system:

(i) $\{\,\} \vdash \lambda x_1\,(\lambda x_2\,(x_1)) : \forall \alpha_1, \alpha_2\,(\alpha_1 \to (\alpha_2 \to \alpha_1))$

(ii) $\{\,\} \vdash \lambda x\,(x :: \texttt{nil}) : \forall \alpha\,(\alpha \to \alpha\ \mathit{list})$

(iii) $\{\,\} \vdash \lambda x\,(\texttt{case}\ x\ \texttt{of}\ \texttt{nil} \Rightarrow \texttt{true}\ \texttt{|}\ x_1 :: x_2 \Rightarrow \texttt{false}) : \forall \alpha\,(\alpha\ \mathit{list} \to \mathit{bool})$

(iv) $\{\,\} \vdash \texttt{let}\ f = \lambda x_1\,(\lambda x_2\,(x_1))\ \texttt{in}\ f\ f : \forall \alpha_1, \alpha_2, \alpha_3\,(\alpha_1 \to (\alpha_2 \to (\alpha_3 \to \alpha_2)))$

Exercise 2. Which of the following are valid instances of the specialisation relation between ML type schemes and types?

(i) $\forall \alpha_1, \alpha_2 (\alpha_1 \to \alpha_2) \succ (\alpha_1 \to \alpha_1) \to \alpha_1$

(ii) $\forall \alpha_1 (\alpha_1 \to \alpha_2) \succ (\alpha_1 \to \alpha_1) \to \alpha_1$

(iii) $\forall \alpha_1 (\alpha_1 \to \alpha_2) \succ (\alpha_2 \to \alpha_2) \to \alpha_2$

(iv) $\forall \alpha_1 (\alpha_1 \to \alpha_1) \succ (\alpha_1 \to \alpha_1) \to \alpha_2$

Exercise 3. Show that if $\{\,\} \vdash M : \tau$ is provable from the Mini-ML typing rules, then M must be *closed*, i.e. have no free variables. [Hint: use rule induction for the rules on Slides 16–18 to show that the provable typing judgements, $\Gamma \vdash M : \tau$, all have the property that $\mathit{fv}(M) \subseteq \mathit{dom}(\Gamma)$.]

Exercise 4. Consider the following Mini-ML typing problems (Slide 26).

(i) $x : \forall\{\beta\}(\beta \to \alpha) \vdash x\,(x\,\texttt{nil}) : ?$

(ii) $x : \forall\{\alpha\}(\beta \to \alpha) \vdash x\,(x\,\texttt{nil}) : ?$

(iii) $x : \forall\{\beta\}(\beta \to \alpha\texttt{list}) \vdash x :: (x\,\texttt{nil}) : ?$

(iv) $x : \forall\{\alpha\}(\beta \to \alpha\texttt{list}) \vdash x :: (x\,\texttt{nil}) : ?$

For each typing problem, either give a solution together with a proof of typing, or show that no solution exists.

Exercise 5. Complete the definition of *pt* on Slide 31 and in Figure 3 with clauses for `nil`, cons, and `case`-expressions.

Polymorphic Reference Types

Exercise 6. Show that if

$$M \triangleq \texttt{let}\ f = (\lambda x\,(x))\,\lambda y\,(y)\ \texttt{in}\ (f\,\texttt{true}) :: (f\,\texttt{nil})$$

then in Mini-ML $\{\,\} \vdash M : \tau$ is provable for some τ, but that in Midi-ML with the value-restricted rule (letv), it is not provable for any τ.

Exercise 7. Which of the following typing judgements are provable in the Midi-ML type system with the value-restricted rule (letv)?

(i) $\{\,\} \vdash \texttt{let}\ r = \texttt{ref}\ \lambda x\,(x)\ \texttt{in}\ (!r)(r := \lambda y\,(\texttt{true})) : \mathit{unit}$

(ii) $\{\,\} \vdash \texttt{let}\ r = \texttt{ref}\ \lambda x\,(x)\ \texttt{in}\ (!r)(r := \lambda y\,(())) : \mathit{unit}$

(iii) $\{\,\} \vdash \texttt{let}\ f = \lambda x\,(\texttt{ref}\ x)\ \texttt{in}\ f\ f : \sigma$ (for some type scheme σ)

Polymorphic Lambda Calculus

Exercise 8. Consider the ML type system modified as in Example 7, that is, with polymorphic types and with (var $\succ$) replaced by the rules on Slide 43. Show that

$$\{\,\} \vdash \lambda f\,((f\,\texttt{true}) :: (f\,\texttt{nil})) : \pi_i$$

holds in this type system, where

$$\pi_1 \triangleq (\forall \alpha\,(\alpha \to \alpha)) \to \mathit{bool\,list}$$
$$\pi_2 \triangleq (\forall \alpha_1\,(\alpha_1 \to \forall \alpha_2\,(\alpha_2))) \to \mathit{bool\,list}$$

Exercise 9. For each of the following PLC typing judgements, are there PLC types $\tau_1, \ldots, \tau_5$ that make the judgements provable?

(i) $\{\,\} \vdash \lambda x : \forall \alpha\,(\alpha)\,(\Lambda \beta\,(x\,\beta)) : \tau_1$

(ii) $\{\,\} \vdash \Lambda \alpha\,(\lambda x : \alpha\,(\Lambda \beta\,(x\,\beta))) : \tau_2$

(iii) $\{\,\} \vdash \lambda x : \tau_3\,(\Lambda \alpha\,(x\,(\alpha \to \alpha)\,(x\,\alpha))) : \tau_3 \to \forall \beta\,(\beta)$

(iv) $\{\,\} \vdash \lambda x : \tau_4\,(\Lambda \alpha\,(x\,(\alpha \to \alpha)\,(x\,\alpha))) : \tau_4 \to \forall \alpha\,(\alpha \to \alpha)$

(v) $\{\,\} \vdash \Lambda \alpha\,(\lambda x : \tau_5\,(x\,(\alpha \to \alpha)\,(x\,\alpha))) : \forall \alpha\,(\alpha \to \alpha)$

Exercise 10. In PLC, defining the expression $\texttt{let}\,x = M_1 : \tau\,\texttt{in}\,M_2$ to be an abbreviation for $(\lambda x : \tau\,(M_2))\,M_1$, show that the typing rule

$$\frac{\Gamma \vdash M_1 : \tau_1 \qquad \Gamma, x : \tau_1 \vdash M_2 : \tau_2}{\Gamma \vdash (\texttt{let}\,x = M_1 : \tau_1\,\texttt{in}\,M_2) : \tau_2} \quad \text{if } x \notin \mathit{dom}(\Gamma)$$

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

Exercise 11. The *erasure*, $\mathit{erase}(M)$, of a PLC expression M is the expression of the untyped lambda calculus obtained by deleting all type information from M:

$$\begin{aligned}
\mathit{erase}(x) &\triangleq x\\
\mathit{erase}(\lambda x : \tau\,(M)) &\triangleq \lambda x\,(\mathit{erase}(M))\\
\mathit{erase}(M_1\,M_2) &\triangleq \mathit{erase}(M_1)\,\mathit{erase}(M_2)\\
\mathit{erase}(\Lambda \alpha\,(M)) &\triangleq \mathit{erase}(M)\\
\mathit{erase}(M\,\tau) &\triangleq \mathit{erase}(M).
\end{aligned}$$

(i) Find PLC expressions M_1 and M_2 satisfying $\mathit{erase}(M_1) = \lambda x\,(x) = \mathit{erase}(M_2)$ 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 12 that there is a closed PLC expression M of type $\forall \alpha\,(\alpha) \to \forall \alpha\,(\alpha)$ satisfying $\mathit{erase}(M) = \lambda f\,(f\,f)$. Find some other closed, typeable PLC expressions with this property.

(iii) [For this part you will need to recall from the CST Part IB *Computation Theory* 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 $\mathit{erase}(M)$ is strongly normalisable in the untyped lambda calculus, i.e. there are no infinite chains of beta-reductions starting from $\mathit{erase}(M)$. Assuming this result, exhibit an expression of the untyped lambda calculus which is not equal to $\mathit{erase}(M)$ for any closed, typeable PLC expression M.

Exercise 12. Define $\alpha_1 * \alpha_2 \triangleq \forall\alpha\,((\alpha_1 \to \alpha_2 \to \alpha) \to \alpha)$. Show that there are PLC expressions *Pair*, *fst*, and *snd* satisfying:

$$\begin{aligned}
&\{\,\} \vdash \mathit{Pair} : \forall\alpha_1, \alpha_2\,(\alpha_1 \to \alpha_2 \to (\alpha_1 * \alpha_2)) && (16)\\
&\{\,\} \vdash \mathit{fst} : \forall\alpha_1, \alpha_2\,((\alpha_1 * \alpha_2) \to \alpha_1) && (17)\\
&\{\,\} \vdash \mathit{snd} : \forall\alpha_1, \alpha_2\,((\alpha_1 * \alpha_2) \to \alpha_2) && (18)\\
&\mathit{fst}\,\alpha_1\,\alpha_2(\mathit{Pair}\,\alpha_1\,\alpha_2\,x_1\,x_2) =_\beta x_1 && (19)\\
&\mathit{snd}\,\alpha_1\,\alpha_2(\mathit{Pair}\,\alpha_1\,\alpha_2\,x_1\,x_2) =_\beta x_2. && (20)
\end{aligned}$$

Exercise 13. Suppose that τ is a PLC type with a single free type variable, α. Suppose also that T is a closed PLC expression satisfying a (weak) 'functoriality' property:

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

Define ι to be the closed PLC type

$$\iota \triangleq \forall\alpha\,((\tau \to \alpha) \to \alpha).$$

Show how to define PLC expressions R and I satisfying

$$\begin{aligned}
&\{\,\} \vdash R : \forall\alpha\,((\tau \to \alpha) \to \iota \to \alpha) && (21)\\
&\{\,\} \vdash I : \tau[\iota/\alpha] \to \iota && (22)\\
&(R\,\beta\,f)(I\,x) \to^* f\,(T\,\iota\,\beta\,(R\,\beta\,f)\,x). && (23)
\end{aligned}$$

(Category-theoretic background: altogether these properties say that (ι, I) is a weak initial τ-algebra: given any τ-algebra $(\beta, f : \tau[\beta/\alpha] \to \beta)$, we get $R\,\beta\,f : \iota \to \beta$ making the square of functions

$$\begin{tikzcd}
\tau[\iota/\alpha] \arrow[r, "I"] \arrow[d, "T\,\iota\,\beta\,(R\,\beta\,f)"'] & \iota \arrow[d, "R\,\beta\,f"] \\
\tau[\beta/\alpha] \arrow[r, "f"'] & \beta
\end{tikzcd}$$

commute up to beta reduction.) [Hint: you will need to use R in the definition of I.]

Dependent Types

Exercise 14. The translation from PLC to the PTS $\lambda 2$ given on Slide 77 sends the PLC type $\forall\alpha\,(\alpha \to \alpha)$ to the pseudo-term $\Pi\alpha : *\,(\Pi x : \alpha\,(\alpha))$ and the PLC term $\Lambda\alpha\,(\lambda x : \alpha\,(x))$ to the pseudo-term $\lambda\alpha : *\,(\lambda x : \alpha\,(x))$. Verify that the judgement

$$\diamond \vdash \lambda\alpha : *\,(\lambda x : \alpha\,(x)) : \Pi\alpha : *\,(\Pi x : \alpha\,(\alpha))$$

is provable in $\lambda 2$. Indicate clearly where the proof uses the axiom $(*, \square)$ and the rules $(*, *, *)$ and $(\square, *, *)$. (If you do not use them all, you are doing something wrong.)

Exercise 15. Give an example of pseudo-terms M and A in the PTS $\lambda\omega$ satisfying $\diamond \vdash M : A$ for which the proof of typing makes use of the PTS rule (conv) (Slide 73).

Exercise 16. By analogy with the encoding of existential types in PLC (Slide 63), define pseudo-terms *exists*, *pack* and *unpack* in the PTS $\lambda\omega$ satisfying

$$\begin{aligned}
&\diamond \vdash \mathit{exists} : (* \to *) \to * && (24)\\
&\diamond \vdash \mathit{pack} : \Pi T : * \to *\,(\Pi\alpha : *\,(T\,\alpha \to \mathit{exists}\,T)) && (25)\\
&\diamond \vdash \mathit{unpack} : \Pi T : * \to *\,(\mathit{exists}\,T \to \Pi\beta : *\,((\Pi\alpha : *\,(T\,\alpha \to \beta)) \to \beta)) && (26)\\
&\mathit{unpack}\,T\,(\mathit{pack}\,T\,\alpha\,x)\,\beta\,f \to^* f\,\alpha\,x && (27)
\end{aligned}$$

Propositions as Types

Exercise 17. Conjunction (*conj*) and bi-implication (*bimp*) can be defined in the Calculus of Constructions λC in the same way as they are in 2IPC (Slide 91):

$$conj \triangleq \lambda p, q : \texttt{Prop}\,(\Pi r : \texttt{Prop}\,((p \to q \to r) \to r)) \tag{28}$$
$$\diamond \vdash conj : \texttt{Prop} \to \texttt{Prop} \to \texttt{Prop}$$
$$bimp \triangleq \lambda p, q : \texttt{Prop}\,(conj\,(p \to q)\,(q \to p)) \tag{29}$$
$$\diamond \vdash bimp : \texttt{Prop} \to \texttt{Prop} \to \texttt{Prop}$$

Bi-implication is used in the definition of Leibniz equality on Slide 96, where $P\,x \leftrightarrow P\,y$ stands for $bimp\,(P\,x)\,(P\,y)$:

$$Eq_A \triangleq \lambda x, y : A\,(\Pi P : A \to \texttt{Prop}\,(P\,x \leftrightarrow P\,y)) \tag{30}$$
$$\Gamma \vdash Eq_A : A \to A \to \texttt{Prop} \quad \text{if} \quad \Gamma \vdash A : \texttt{Set}$$

Show that the simpler definition

$$Eq'_A \triangleq \lambda x, y : A\,(\Pi P : A \to \texttt{Prop}\,(P\,x \to P\,y))$$

gives a logically equivalent notion of equality by constructing pseudo-terms F, G satisfying

$$\Gamma \vdash F : \Pi x, y : A\,(Eq_A\,x\,y \to Eq'_A\,x\,y)$$
$$\Gamma \vdash G : \Pi x, y : A\,(Eq'_A\,x\,y \to Eq_A\,x\,y)$$

(assuming $\Gamma \vdash A : \texttt{Set}$). [Hint for G: given $x, y : A$, $f : \Pi P : A \to \texttt{Prop}\,(P\,x \to P\,y)$ and $P : A \to \texttt{Prop}$, we can get a function $P\,y \to P\,x$ by applying f to $\lambda z : A\,(P\,z \to P\,x)$ and $\lambda p : P\,x\,(p)$.]

Exercise 18. In λC extended with inductively defined identity propositions (Slide 102) construct proofs that equality is symmetric and transitive

$$\Gamma \vdash symm_A : \Pi x, y : A\,(\texttt{Id}_{A,x}y \to \texttt{Id}_{A,y}x) \tag{31}$$
$$\Gamma \vdash trans_A : \Pi x, y : A\,(\texttt{Id}_{A,x}y \to \Pi z : A\,(\texttt{Id}_{A,y}z \to \texttt{Id}_{A,x}z)) \tag{32}$$

(where $\Gamma \vdash A : s$).

Exercise 19. In λC extended with an inductive type of natural numbers (Slide 100) and inductive identity propositions (Slide 102), give a pseudo-term P satisfying

$$\diamond \vdash P : \Pi x : \texttt{Nat}\,(\texttt{Id}_{\texttt{Nat},x}(add\,\texttt{zero}\,x)) \tag{33}$$

where

$$add \triangleq \lambda x : \texttt{Nat}\,(\texttt{elimNat}(y.\,\texttt{Nat})\,x\,(\lambda y : \texttt{Nat}\,(\texttt{succ}))) \tag{34}$$

[Hint: try to convert the Agda proof on Slide 103, which uses Agda's pattern-matching facilities, into a proof using the eliminators `elimNat` and `J`.]