1 Parametric polymorphism

(a) For each of the following System F expressions, is the expression well-typed, and if so, what type does it have? (If you are unsure, try to construct a typing derivation. Make sure you understand the typing rules.)

   • \( \Lambda A. \lambda x : A \rightarrow \texttt{int}. 42 \)
   • \( \lambda y : \forall X. X \rightarrow X. (y \texttt{[int]}) 17 \)
   • \( \Lambda Y. \Lambda Z. \lambda f : Y \rightarrow Z. \lambda a : Y. f a \)
   • \( \Lambda A. \Lambda B. \Lambda C. \lambda f : A \rightarrow B \rightarrow C. \lambda b : B. \lambda a : A. f a b \)

(b) For each of the following types, write an expression with that type.

   • \( \forall X. X \rightarrow (X \rightarrow X) \)
   • \( (\forall C. \forall D. C \rightarrow D) \rightarrow (\forall E. \texttt{int} \rightarrow E) \)
   • \( \forall X. X \rightarrow (\forall Y. Y \rightarrow X) \)

2 Records and Subtyping

(a) Assume that we have a language with references and records.

   (i) Write an expression with type

       \( \{ \texttt{cell : int ref, inc : unit} \rightarrow \texttt{int} \} \)

       such that invoking the function in the field \( \texttt{inc} \) will increment the contents of the reference in the field \( \texttt{cell} \).

   (ii) Assuming that the variable \( y \) is bound to the expression you wrote for part (i) above, write an expression that increments the contents of the cell twice.

(b) The following expression is well-typed (with type \( \texttt{int} \)). Show its typing derivation. (Note: you will need to use the subsumption rule.)

   \( (\lambda x : \{ \texttt{dogs : int, cats : int} \}. x.\text{dogs} + x.\text{cats}) \{ \texttt{dogs = 2, cats = 7, mice = 19} \} \)

(c) Suppose that \( \Gamma \) is a typing context such that

   \( \Gamma(a) = \{ \texttt{dogs : int, cats : int, mice : int} \} \)
   \( \Gamma(f) = \{ \texttt{dogs : int, cats : int} \} \rightarrow \{ \texttt{apples : int, kiwis : int} \} \)

   Write an expression \( e \) that uses variables \( a \) and \( f \) and has type \( \{ \texttt{apples : int} \} \) under context \( \Gamma \), i.e., \( \Gamma \vdash e : \{ \texttt{apples : int} \} \). Write a typing derivation for it.

(d) Which of the following are subtypes of each other?
For each such pair, make sure you have an understanding of why one is a subtype of the other (and for pairs that aren’t subtypes, also make sure you understand).

3 Curry-Howard isomorphism

The following logical formulas are tautologies, i.e., they are true. For each tautology, state the corresponding type, and come up with a term that has the corresponding type.

For example, for the logical formula $\forall \phi. \forall \psi. \phi \land \psi = \Rightarrow \phi$, the corresponding type is $\forall X. X \rightarrow X$, and a term with that type is $\lambda x:X. x$. Another example: for the logical formula $\tau_1 \land \tau_2 = \Rightarrow \tau_1$, the corresponding type is $\tau_1 \times \tau_2 \rightarrow \tau_1$, and a term with that type is $\lambda x: \tau_1 \times \tau_2 . \#1 x$.

You may assume that the lambda calculus you are using for terms includes integers, functions, products, sums, universal types and existential types.

(a) $\forall \phi. \forall \psi. \phi \land \psi = \Rightarrow \phi \lor \phi$

(b) $\forall \phi. \forall \psi. \forall \chi. (\phi \land \psi = \Rightarrow \chi) = \Rightarrow (\phi = \Rightarrow (\psi = \Rightarrow \chi))$

(c) $\exists \phi. \forall \psi. \psi = \Rightarrow \phi$

(d) $\forall \psi. \psi = \Rightarrow (\forall \phi. \phi = \Rightarrow \psi)$

(e) $\forall \psi. (\forall \phi. \phi = \Rightarrow \psi) = \Rightarrow \psi$

4 Existential types

(a) Write a term with type $\exists C. \{ \text{produce} : \text{int} \rightarrow C, \text{consume} : C \rightarrow \text{bool} \}$. Moreover, ensure that calling the function $\text{produce}$ will produce a value of type $C$ such that passing the value as an argument to $\text{consume}$ will return true if and only if the argument to $\text{produce}$ was 42. (Assume that you have an integer comparison operator in the language.)

(b) Do the same as in part (a) above, but now use a different witness type.

(c) Assuming you have a value $v$ of type $\exists C. \{ \text{produce} : \text{int} \rightarrow C, \text{consume} : C \rightarrow \text{bool} \}$, use $v$ to “produce” and “consume” a value (i.e., make sure you know how to use the $\text{unpack} \{X,x\} = e_1$ in $e_2$ expression.