Harvard School of Engineering and Applied Sciences — CS 152: Programming Languages

Parametric Polymorphism; Records and Subtyping; Curry-Howard Isomorphism; Existential Types
Section and Practice Problems

Mar 30 – Apr 3, 2020

---

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.)

- $\Lambda A.\, \lambda x\!:\!A \to \textbf{int}.\, 42$
- $\lambda y\!:\!\forall X.\, X \to X.\, (y\ [\textbf{int}])\ 17$
- $\Lambda Y.\, \Lambda Z.\, \lambda f\!:\!Y \to Z.\, \lambda a\!:\!Y.\, f\ a$
- $\Lambda A.\, \Lambda B.\, \Lambda C.\, \lambda f\!:\!A \to B \to C.\, \lambda b\!:\!B.\, \lambda a\!:\!A.\, f\ a\ b$

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

- $\forall X.\, X \to (X \to X)$
- $(\forall C.\, \forall D.\, C \to D) \to (\forall E.\, \textbf{int} \to E)$
- $\forall X.\, X \to (\forall Y.\, Y \to X)$

2 Records and Subtyping

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

(i) Write an expression with type

$$\{\ \mathit{cell} : \textbf{int ref}, \mathit{inc} : \textbf{unit} \to \textbf{int}\ \}$$

such that invoking the function in the field *inc* will increment the contents of the reference in the field *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 **int**). Show its typing derivation. (Note: you will need to use the subsumption rule.)

$$(\lambda x\!:\!\{\mathit{dogs} : \textbf{int}, \mathit{cats} : \textbf{int}\}.\, x.\mathit{dogs} + x.\mathit{cats})\ \{\mathit{dogs} = 2, \mathit{cats} = 7, \mathit{mice} = 19\}$$

(c) Suppose that Γ is a typing context such that

$$\Gamma(a) = \{\mathit{dogs}\!:\!\textbf{int}, \mathit{cats}\!:\!\textbf{int}, \mathit{mice}\!:\!\textbf{int}\}$$
$$\Gamma(f) = \{\mathit{dogs}\!:\!\textbf{int}, \mathit{cats}\!:\!\textbf{int}\} \to \{\mathit{apples}\!:\!\textbf{int}, \mathit{kiwis}\!:\!\textbf{int}\}$$

Write an expression e that uses variables a and f and has type $\{\mathit{apples} : \textbf{int}\}$ under context Γ, i.e., $\Gamma \vdash e\!:\!\{\mathit{apples}\!:\!\textbf{int}\}$. Write a typing derivation for it.

(d) Which of the following are subtypes of each other?

(a) $\{dogs : \textbf{int}, cats : \textbf{int}\} \to \{apples : \textbf{int}\}$
(b) $\{dogs : \textbf{int}\} \to \{apples : \textbf{int}\}$
(c) $\{dogs : \textbf{int}\} \to \{apples : \textbf{int}, kiwis : \textbf{int}\}$
(d) $\{dogs : \textbf{int}, cats : \textbf{int}, mice : \textbf{int}\} \to \{apples : \textbf{int}, kiwis : \textbf{int}\}$
(e) $(\{apples : \textbf{int}\})$ **ref**
(f) $(\{apples : \textbf{int}, kiwis : \textbf{int}\})$ **ref**
(g) $(\{kiwis : \textbf{int}, apples : \textbf{int}\})$ **ref**

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 $\forall\phi.\phi \implies \phi$, the corresponding type is $\forall X.\ X \to X$, and a term with that type is $\Lambda X.\ \lambda x\!:\!X.\, x$. Another example: for the logical formula $\tau_1 \wedge \tau_2 \implies \tau_1$, the corresponding type is $\tau_1 \times \tau_2 \to \tau_1$, 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) $\forall\phi.\ \forall\psi.\ \phi \wedge \psi \implies \psi \vee \phi$

(b) $\forall\phi.\ \forall\psi.\ \forall\chi.\ (\phi \wedge \psi \implies \chi) \implies (\phi \implies (\psi \implies \chi))$

(c) $\exists\phi.\ \forall\psi.\ \psi \implies \phi$

(d) $\forall\psi.\ \psi \implies (\forall\phi.\ \phi \implies \psi)$

(e) $\forall\psi.\ (\forall\phi.\ \phi \implies \psi) \implies \psi$

4 Existential types

(a) Write a term with type $\exists C.\ \{\ produce : \textbf{int} \to C, consume : C \to \textbf{bool}\ \}$. Moreover, ensure that calling the function *produce* will produce a value of type C such that passing the value as an argument to *consume* will return true if and only if the argument to *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 $\exists C.\ \{\ produce : \textbf{int} \to C, consume : C \to \textbf{bool}\ \}$, use v to "produce" and "consume" a value (i.e., make sure you know how to use the unpack $\{X, x\} = e_1$ in e_2 expression.