1 Products and Sums

For these questions, use the lambda calculus with products and sums (Lecture 13§1.1).

(a) Write a program that constructs two values of type int + (int → int), one using left injection, and one using right injection.

(b) Write a function that takes a value of type int + (int → int) and if the value is an integer, it adds 7 to it, and if the value is a function it applies the function to 42.

(c) Give a typing derivation for the following program.
\[ \lambda p: (\text{unit} \to \text{int}) \times (\text{int} \to \text{int}).\ \lambda x: \text{unit} + \text{int}.\ \text{case}\ x\ of\ \#1\ p\ |\ \#2\ p \]

(d) Write a program that uses the term in part (c) above to produce the value 42.

2 Recursion

(a) Use the \( \mu x. e \) expression to write a function that takes a natural number \( n \) and returns the sum of all even natural numbers less than or equal to \( n \). (You can assume you have appropriate integer comparison operators, and also a modulus operator.)

(b) Try executing your program by applying it to the number 5.

(c) Give a typing derivation for the following program. What happens if you execute the program?
\[ \mu p: (\text{int} \to \text{int}) \times (\text{int} \to \text{int}). (\lambda n: \text{int}.\ n + 1, \#1\ p) \]

3 References

(a) Give a typing derivation for the following program.
\[
\begin{align*}
\text{let } a: \text{int ref } &= \text{ref 4 in} \\
\text{let } b: (\text{int} \to \text{int}) \text{ ref } &= \text{ref } \lambda x: \text{int}.\ x + 38 \text{ in} \\
!b \!a
\end{align*}
\]

(b) Execute the program above for 4 small steps, to get configuration \( (e, \sigma) \). What is an appropriate \( \Sigma \) such that \( \emptyset, \Sigma \vdash e: \tau \) and \( \Sigma \vdash \sigma \)?

(c) Consider a store \( \sigma = [\ell_1 \mapsto 42, \ell_2 \mapsto \lambda n: \text{int}.\ n + 1] \). What is the domain of \( \sigma \)?
Now consider a store type \( \Sigma = [\ell_1 \mapsto \text{int}, \ell_2 \mapsto \text{int} \to \text{int}] \). Note that \( \text{dom}(\sigma) = \text{dom}(\Sigma) \).
Show that \( \emptyset, \Sigma \vdash \sigma \).
4 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.)

- \( \Delta A. \lambda x : A \rightarrow \text{int} \)
- \( \lambda y : \forall X. X \rightarrow X. (y [\text{int}]) \)
- \( \Delta Y. \Delta Z. \lambda f : Y \rightarrow Z. \lambda a : Y. f a \)
- \( \Delta A. \Delta B. \Delta C. \lambda f : A \rightarrow B \rightarrow C. \lambda b : B. \lambda a : A. f a b \)

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

- \( \forall X. X \rightarrow (X \rightarrow X) \)
- \( (\forall C. \forall D. C \rightarrow D) \rightarrow (\forall E. \text{int} \rightarrow E) \)
- \( \forall X. X \rightarrow (\forall Y. Y \rightarrow X) \)

5 Records and Subtyping

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

(i) Write an expression with type

\[ \{ \text{cell} : \text{int ref}, \text{inc} : \text{unit} \rightarrow \text{int} \} \]

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 \( \text{int} \)). Show its typing derivation. (Note: you will need to use the subsumption rule.)

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

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

\[ \Gamma(a) = \{ \text{dogs} : \text{int}, \text{cats} : \text{int}, \text{mice} : \text{int} \} \]

\[ \Gamma(f) = \{ \text{dogs} : \text{int}, \text{cats} : \text{int} \} \rightarrow \{ \text{apples} : \text{int}, \text{kiwis} : \text{int} \} \]

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

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

(a) \( \{ \text{dogs} : \text{int}, \text{cats} : \text{int} \} \rightarrow \{ \text{apples} : \text{int} \} \)
(b) \( \{ \text{dogs} : \text{int} \} \rightarrow \{ \text{apples} : \text{int} \} \)
(c) \( \{ \text{dogs} : \text{int} \} \rightarrow \{ \text{apples} : \text{int}, \text{kiwis} : \text{int} \} \)
(d) \( \{ \text{dogs} : \text{int}, \text{cats} : \text{int}, \text{mice} : \text{int} \} \rightarrow \{ \text{apples} : \text{int}, \text{kiwis} : \text{int} \} \)
(e) \( (\{ \text{apples} : \text{int} \}) \) ref
(f) \( (\{ \text{apples} : \text{int}, \text{kiwis} : \text{int} \}) \) ref
(g) \( (\{ \text{kiwis} : \text{int}, \text{apples} : \text{int} \}) \) 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).