CS 565 Spring 2022 Homework 6
(Type Inference + Subtyping)

Your name: _____

April 10, 2022

Problem 1 (1 point). Construct a constraint typing derivation whose conclusion is

$$\vdash \lambda \mathsf{x} : \mathsf{X}.\ \lambda \mathsf{y} : \mathsf{Y}.\ \lambda \mathsf{z} : \mathsf{Z}.\ (\mathsf{x}\ \mathsf{z})\ (\mathsf{y}\ \mathsf{z}) : \mathsf{S} \ \mid \ \mathcal{C}$$

for some S, $\mathcal{C}$.

Problem 2 (2 points). Write down principal unifers (when they exist) for the following sets of constraints:

- $\{\}$ (The empty set of constraints)

- $\{\mathsf{Y} = \mathsf{V} \to \mathsf{U}, \mathsf{Y} = \mathsf{X} \to \mathsf{V}\}$

- $\{\mathsf{X} = \mathsf{Bool}, \mathsf{Y} = \mathsf{X} \to \mathsf{X}\}$

- $\{\mathsf{Bool} \to \mathsf{Bool} = \mathsf{X} \to \mathsf{Y}\}$

- $\{(\mathsf{Bool} \to \mathsf{Y}) \to \mathsf{Bool} = \mathsf{Bool} \to \mathsf{U}\}$

Problem 3 (2 points). Suppose we have types S, T, U, and V with S <: T and U <: V. Which of the following subtyping assertions are then true? Write true or false after each one.

- T→S <: T→S

- T→T→U <: S→S→V

- (T→T)→U <: (S→S)→V

- ((T→S)→T)→U <: ((S→T)→S)→V

Problem 4 (1 point). How many supertypes does the type

$$\{\{\mathsf{x}\text{: }\{\mathsf{z}\text{:Bool, }\mathsf{q}\text{: Nat}\}, \mathsf{y}\text{: Bool} \to \mathsf{Bool}\}\}$$

have? That is, how many different types T are there such that

$$\{\mathsf{x}\text{: }\{\mathsf{z}\text{:Bool, }\mathsf{q}\text{: Nat}\}, \mathsf{y}\text{: Bool} \to \mathsf{Bool}\} <: \mathsf{T}$$

(We consider two types to be different if they are written differently, even if each is a subtype of the other. For example, $\{\mathsf{x}\text{:A},\mathsf{y}\text{:B}\}$ and $\{\mathsf{y}\text{:B},\mathsf{x}\text{:A}\}$ are different.)

Problem 5 (2 points). The subtyping rule for product types:

$$\frac{\mathsf{S}_1 <: \mathsf{T}_1 \qquad \mathsf{S}_2 <: \mathsf{T}_2}{\mathsf{S}_1 * \mathsf{S}_2 <: \mathsf{T}_1 * \mathsf{T}_2}$$

intuitively corresponds to the "depth" subtyping rule for records. Extending the analogy, a language designer might consider including a "permutation" rule as well

$$\frac{}{\mathsf{T}_1 * \mathsf{T}_2 <: \mathsf{T}_2 * \mathsf{T}_1}$$

for products. Explain in a couple of sentences why such a subtyping rule is or is not sound?