1 Type Inference

(a) Recall the constraint-based typing judgment $\Gamma \vdash e : \tau \triangleright C$. Give inference rules for products and sums. That is, for the following expressions.

- $(e_1, e_2)$
- $\#1 \ e$
- $\#2 \ e$
- $\text{inl}_{\tau_1 + \tau_2} \ e$
- $\text{inr}_{\tau_1 + \tau_2} \ e$
- $\text{case } e_1 \text{ of } e_2 \mid e_3$

(b) Determine a set of constraints $C$ and type $\tau$ such that

$\vdash \lambda y : A. \lambda y : B. (\#1 \ y) + (x \ (\#2 \ y)) + (x \ 2) : \tau \triangleright C$

and give the derivation for it.

(c) Recall the unification algorithm from Lecture 14. What is the result of $\text{unify}(C)$ for the set of constraints $C$ from Question 1(b) above?

2 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.)

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

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

- $\forall X. X \to (X \to X)$
- $(\forall C. \forall D. C \to D) \to (\forall E. \text{int} \to E)$
- $\forall X. X \to (\forall Y. Y \to X)$
3 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 \text{inc} will increment the contents of the reference in the field \text{cell}.

(ii) Assuming that the variable \text{y} is bound to your expression, 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.)

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