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.

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

**Answer:**

Note that in all of the rules below except for the rule for pairs $(e_1, e_2)$, the types in the premise and conclusion are connected only through constraints. The reason for this is the same as in the typing rule for function application, and for addition: we may not be able to derive that the premise has the appropriate type, e.g., for a projection $\#1 e$, we may not be able to derive that $\Gamma \vdash e : \tau_1 \times \tau_2 \triangleright C$. We instead use constraints to ensure that the derived type is appropriate.

\[
\begin{align*}
\Gamma \vdash e_1 : \tau_1 \triangleright C_1 & & \Gamma \vdash e_2 : \tau_2 \triangleright C_2 \\
\hline
\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2 \triangleright C_1 \cup C_2
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash e : \tau \triangleright C & & X, Y \text{ are fresh} \\
\hline
\Gamma \vdash \#1 e : X \triangleright C \cup \{\tau \equiv X \times Y\}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash e : \tau \triangleright C & & X, Y \text{ are fresh} \\
\hline
\Gamma \vdash \#2 e : Y \triangleright C \cup \{\tau \equiv X \times Y\}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash e : \tau \triangleright C & & \text{X, Y are fresh} \\
\hline
\Gamma \vdash \text{inl}_{\tau_1 + \tau_2} e : \tau_1 + \tau_2 \triangleright C \cup \{\tau \equiv \tau_1\}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash e : \tau \triangleright C & & \text{X, Y are fresh} \\
\hline
\Gamma \vdash \text{inr}_{\tau_1 + \tau_2} e : \tau_1 + \tau_2 \triangleright C \cup \{\tau \equiv \tau_2\}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash e_1 : \tau_1 \triangleright C_1 & & \Gamma \vdash e_2 : \tau_2 \triangleright C_2 & & \Gamma \vdash e_3 : \tau_3 \triangleright C_3 \\
\hline
\Gamma \vdash \text{case } e_1 \text{ of } e_2 \mid e_3 : Z \triangleright C_1 \cup C_2 \cup C_3 \cup \{\tau_1 \equiv X + Y, \tau_2 \equiv X \rightarrow Z, \tau_3 \equiv Y \rightarrow Z\}
\end{align*}
\]

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

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

and give the derivation for it.
Answer:

\[ C = \{ B \equiv X \times Y , X \equiv \text{int}, B \equiv Z \times W , A \equiv W \rightarrow U , U \equiv \text{int}, A \equiv \text{int} \rightarrow V , V \equiv \text{int} \} \]

\[ \tau \equiv A \rightarrow B \rightarrow \text{int} \]

To see how we got these constraints, we will consider the subexpressions in turn (rather than trying to typeset a really really big derivation).

The expression \(#1 y\) requires us to add a constraint that the type of \(y\) (i.e., \(B\)) is equal to a product type for some fresh variables \(X\) and \(Y\), thus constraint \(B \equiv X \times Y\). (And expression \(#1 y\) has type \(X\).)

The expression \((#2 y)\) similarly requires us to add a constraint that the type of \(y\) (i.e., \(B\)) is equal to a product type for some fresh variables \(Z\) and \(W\), thus constraint \(B \equiv Z \times W\). (And expression \(#2 y\) has type \(W\).)

The expression \(x (#2 y)\) requires us to add a constraint that unifies the type of \(x\) (i.e., \(A\)) with a function type \(W \rightarrow U\) (where \(W\) is the type of \(#2 y\) and \(U\) is a fresh type variable).

The expression \(x 2\) requires us to add a constraint that unifies the type of \(x\) (i.e., \(A\)) with a function type \(\text{int} \rightarrow V\) (where \(\text{int}\) is the type of expression \(2\) and \(V\) is a fresh type).

The addition operations leads us to add constraints \(X \equiv \text{int}, U \equiv \text{int}\), and \(V \equiv \text{int}\) (i.e., the types of expressions \(#1 y\), \((x (#2 y))\) and \((x 2)\) must all unify with \(\text{int}\)).

(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?

Answer: The result is a substitution equivalent to

\[ [A \mapsto \text{int} \rightarrow \text{int}, B \mapsto \text{int} \times \text{int}, X \mapsto \text{int}, Y \mapsto \text{int}, Z \mapsto \text{int}, W \mapsto \text{int}, U \mapsto \text{int}, V \mapsto \text{int}] \]