1 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 $\forall \phi. \phi \implies \psi \lor \phi$, the corresponding type is $\forall X. X \to X$, and a term with that type is $\Lambda X. \lambda x : X. x$.

You may assume that the lambda calculus you are using for terms includes integers, functions, products, sums, universal types and existential types.

(a) $\forall \phi. \forall \psi. \phi \land \psi \implies \psi \lor \phi$

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

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

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

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

2 Existential types

(a) Write a term with type $\exists C. \{\text{produce} : \text{int} \to C, \text{consume} : C \to \text{bool}\}$. 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 $\exists C. \{\text{produce} : \text{int} \to C, \text{consume} : C \to \text{bool}\}$, use $v$ to “produce” and “consume” a value (i.e., make sure you know how to use the unpack expression).

3 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

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

and give the derivation for it.

(c) Recall the unification algorithm from Lecture 16. What is the result of $\text{unify}(C)$ for the set of constraints $C$ from Question 3(b) above?