Types

Explain the term minimal type and discuss its importance in typechecking algorithms for type systems with subtyping. What is the difference between a minimal type and a principal typing? [6 marks]

Write subtyping and typing algorithms (either as syntax-directed systems of inference rules or as pseudo-code) for the following “core” of the simply typed lambda-calculus with subtyping.

\[ e ::= x \]
\[ \text{fun}(x \in T)e \]
\[ e_1 e_2 \]

\[ T ::= T_1 \rightarrow T_2 \]
\[ \text{Top} \]

Your algorithms need not handle records or booleans. [6 marks]

Suppose that we add to this calculus a type \( \text{Box}(T) \) for each type \( T \), and the expression constructors

\[ e ::= \ldots \]
\[ \text{box } e \]
\[ \text{contents } e \]
\[ e_1 \leftarrow e_2 \]

with the following evaluation rules:

\[ \frac{e \Downarrow r}{\text{box } e \Downarrow \text{box } r} \]

\[ \frac{e \Downarrow \text{box } r}{\text{contents } e \Downarrow r} \]

\[ \frac{e_1 \Downarrow \text{box } r_1 \quad e_2 \Downarrow r_2}{e_1 \leftarrow e_2 \Downarrow \text{box } r_2} \]

Write sound typing and subtyping rules for these constructs. [5 marks]

Now suppose that we add to this calculus the type variables and bounded universal quantification of System \( F_{\leq} \). Indicate how your typing and/or subtyping rules must change (while remaining sound!) so that the expression

\[ \text{fun}(X \leq \text{Box}(\text{Top} \rightarrow \text{Top})) \text{fun}(x \in X) \; x \leftarrow (\text{fun}(y \in \text{Top}) \; y) \]

has the type

\[ \text{All}(X \leq \text{Box}(\text{Top} \rightarrow \text{Top})) \; X \rightarrow X \] [3 marks]