9 Semantics of Programming Languages (PMS)

Consider the following language with higher-order functions and mutable global references. Here \( l \) ranges over mutable location names, that can hold arbitrary values \( v \), \( n \) ranges over natural numbers, and \( x \) ranges over immutable variable names.

\[
\begin{align*}
e &::= n \mid l \mid x \mid \mathbf{fn} \ x : T \Rightarrow e \mid e \ e' \mid e := e' \mid !e \\
v &::= n \mid l \mid \mathbf{fn} \ x : T \Rightarrow e
\end{align*}
\]

Suppose it has a standard left-to-right call-by-value operational semantics. You need not state this semantics.

The definition of \( \Gamma \vdash \text{eff} \ e : T \) below, where an effect \( \text{eff} \) is a subset of \( \{ \text{R}, \text{W} \} \), is a flawed attempt to statically compute a sound approximation of the possible dynamic side-effects of expressions. Such an analysis is sound if \( \text{eff} \) contains \( \text{R} \) and/or \( \text{W} \) whenever there is any execution of \( \langle e, s \rangle \), for any store \( s \) that is well-typed with respect to \( \Gamma \), that involves (respectively) reading and/or writing the store.

Function types are annotated with the latent effects that may occur when the function is applied:

\[
T ::= \text{int} | T \rightarrow \text{eff} \ T' | T_{\text{ref}}
\]

\[
\Gamma \vdash \text{eff} \ e : T
\]

\[
\begin{array}{llllll}
\Gamma \vdash \{ \} n : \text{int} & \quad \text{NUM} & l : \text{ref} \in \Gamma & \quad \text{LOC} & x : T \in \Gamma & \quad \text{VAR} \\
\Gamma \vdash \{ \text{R}, \text{W} \} l : \text{ref} & \text{FN} & \Gamma \vdash \text{eff} \ e : T' & \text{APP} \\
\Gamma, x : T \vdash \text{eff} \ e : T' & \text{ASSIGN} & \Gamma \vdash \text{eff} \ e : \text{ref} & \Gamma \vdash \text{eff} \cup \text{eff'} \cup \{ \text{W} \} e := e' : T & \text{DEREF}
\end{array}
\]

\((a)\) There are three flaws in the above rules, which make them either not sound or an unnecessarily coarse approximation. Explain each flaw, giving a corrected rule for each and an example that shows the problem (assuming the other flaws are fixed). [15 marks]

\((b)\) In the system above, functions have to be applied to arguments of exactly the expected type. Define a subtype relation \( T <: T' \) and subsumption rule that would let function arguments be used even if they have fewer (latent) effects than those anticipated by the function. [5 marks]