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.

$$e ::= n \mid l \mid x \mid \text{fn} \ x : T \Rightarrow e \mid e \ e' \mid e := e' \mid !e$$

$$v ::= n \mid l \mid \text{fn} \ x : T \Rightarrow e$$

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 $\{R, W\}$, 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 $R$ and/or $W$ 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: $T ::= \text{int} \mid T \rightarrow_{\text{eff}} T' \mid T\text{ref}$

\[
\begin{array}{c}
\Gamma \vdash_{\text{eff}} e : T \\
\hline
\Gamma \vdash_{\{} n : \text{int} \quad l : T\text{ref} \in \Gamma \\
\hline
\Gamma \vdash_{\{R, W\}} l : T\text{ref} \\
\hline
\Gamma \vdash l \ x : T \Rightarrow e : T' \\
\hline
\Gamma \vdash_{\{} \text{fn} \ x : T \Rightarrow e : T' \\
\hline
\Gamma \vdash_{\text{eff}} e : T\text{ref} \\
\hline
\Gamma \vdash_{\text{eff}} e : T\text{ref} \\
\hline
\Gamma \vdash_{\text{eff} \cup \{W\}} e := e' : T \\
\hline
\Gamma \vdash_{\text{eff}} e : T \Rightarrow e' : T \\
\hline
\Gamma \vdash_{\text{eff} \cup \text{eff}' \cup \{W\}} e := e' : T \\
\hline
\Gamma \vdash_{\text{eff} \cup \text{eff}' \cup \{R\}} !e : T \\
\hline
\Gamma \vdash_{\text{eff} \cup \text{eff}' \cup \{W\}} e := e' : T \\
\end{array}
\]

(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).

(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.