3 Semantics of Programming Languages (pes20)

Many languages (C, C++, Rust, LLVM IR, etc.) deem some programs to have undefined behaviour (UB). This is done to permit compiler optimisations that are sound only under the assumption that the source program has no UB.

(*a*) Give a semantics for the following C-ish language in which it is UB to use a pointer outside its original allocation. Track the original allocations of pointers in the semantics with pointer values $p ::= (id, n)$ that pair an allocation ID id : ID with an address $n : \mathbb{N}$. The semantics should define a small-step transition relation $A, M, e \longrightarrow A', M', e'$ and an UB relation $A, M, e \longrightarrow \mathbf{UB}$, where A : ID $\rightharpoonup \mathbb{B} \times \mathbb{N} \times \mathbb{N}$ is an *allocation environment* that for each allocation created so far gives its metadata (*live*, *addr*, *size*), and memory $M : \mathbb{N} \to \mathbb{N}$ holds a natural number at each natural-number address. Use evaluation contexts. A source program is *closed* if it does not contain any free variables or literal pointer values. The semantics should either reduce or flag UB for any non-value closed e.

$$
\begin{array}{lll}
e ::= & v \mid e + n \mid *e \mid *e = e' \mid \mathbf{malloc}\,(e) \mid \mathbf{free}\,(e) \mid x \mid \mathbf{let}\, x = e\, \mathbf{in}\, e' \\
v ::= & n \mid p
\end{array}
$$

[14 marks]

(*b*) Explain briefly how your semantics handles use-after-free and use-after-reallocation, with examples of closed programs that exhibit UB because of each.

[3 marks]

(*c*) The top-level semantics of a closed program e are defined to be UB if any execution flags UB, and otherwise the set of integers it can evaluate to:

$$
\begin{array}{ll}
\mathbf{UB} & \text{if } A_0, M_0, e \longrightarrow^* \longrightarrow \mathbf{UB} \\
\{n' \mid \exists A', M'. A_0, M_0, e \longrightarrow^* A', M', n'\} & \text{otherwise}
\end{array}
$$

where $A_0 = \{\}$ and M_0 maps all addresses to 0.

Discuss when it is sound to hoist a load, for example optimising

$$\mathbf{let}\, x = e\, \mathbf{in}\, \mathbf{let}\, y = *z\, \mathbf{in}\, e'$$

to

$$\mathbf{let}\, y = *z\, \mathbf{in}\, \mathbf{let}\, x = e\, \mathbf{in}\, e'$$

[3 marks]