2003 Paper 5 Question 11

Semantics of Programming Languages

The Global Computer Corporation has just started working on a new language, tentatively named HMM, with syntax:

$$\begin{aligned} T &::= \mathsf{int} \mid \mathsf{unit} \mid T \to T \\ e &::= n \mid \textbf{skip} \mid \ell := e \mid !\ell \mid \textbf{fn}\ x : T \Rightarrow e \mid e\ e \mid x \end{aligned}$$

Their initial implementation, written in ML, has a one-step reduction function as follows.

```
fun reduce (Integer n,s) = NONE
  | reduce (Skip,s) = NONE
  | reduce (Deref l,s) = (case lookup  (s,l) of
        SOME n => SOME(Integer n,s)
      | NONE => SOME(Integer 0, (l,0)::s ))
  | reduce (Assign (l,e),s) = (case e of
       Integer n => (case update (s,(l,n)) of
                         SOME s' => SOME(Skip, s')
                       | NONE => SOME(Skip, (l,n)::s))
     | _ => (case reduce (e,s) of
                 SOME (e',s') => SOME(Assign (l,e'), s')
               | NONE => NONE  ) )
  | reduce (Var n,s) = raise (Reduce "bogus unbound Var")
  | reduce (Fn (t,e),s) = NONE
  | reduce (App (e1,e2),s) = (case e1 of
       Fn (t,e) =>  SOME (subst e2 0 e,s)
     | _ => (case reduce (e1,s) of
                 SOME (e1',s') => SOME(App (e1',e2), s')
               | NONE => NONE ))
```

This uses the standard auxilary functions below for manipulating association lists and for substituting an expression for a De Bruijn index in another expression. Following their tradition, they sometimes execute badly-typed programs.

```
update:(string*int) list * (string*int) -> (string*int) list option
lookup:(string*int) list * string -> int option
subst:expr -> int -> expr -> expr
```

(*a*) Give an inductive definition of a reduction relation $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ which corresponds exactly to their implementation. Say carefully what e and s range over in your semantics. [11 marks]

(*b*) Explain how the HMM semantics differs from the "standard" L2 semantics. For each difference, give the standard semantic rule(s), an expression that would behave differently using them, and any reason(s) why one or the other would be a better language design. [9 marks]