Homework #6

CS231

Due by the end of the day on March 10. You should submit one file: `hw6.pdf`.

Remember that you are encouraged to work in pairs on homework assignments. See the course syllabus for details. Also remember the course's academic integrity policy. In particular, you must credit other people and other resources that you consulted. Again, see the syllabus for details.

1. Each of the following programs is written in the simply-typed lambda calculus augmented with mutable references, `let`, the unit value `()`, and integers with addition and multiplication. We also use the syntax $\texttt{t}_1\texttt{;}\,\texttt{t}_2$ as syntactic sugar for $\texttt{let x = t}_1\texttt{ in t}_2$, where `x` is not free in $\texttt{t}_2$.

   Each of these programs contains a *hole*, denoted `...`. Your job is to fill in each hole with a term `t` so that the program evaluates to 42. See the cheat sheet for the semantics of all of these language features. *Tip: Playing with these programs in OCaml should be helpful.*

   (a)
   ```
   let r = ref 41 in
   let x = (...)  in
     !r
   ```

   (b)
   ```
   let r = ref 41 in
   let x = ((function r:Ref Int -> (r := 41; 500)) (...))  in
     !r
   ```

   (c)
   ```
   let f = (...)  in
     (f ()) * (f ())
   ```

2. Suppose we extend the simply-typed lambda calculus with mutable references (see the cheat sheet) to include a term of the form `free t`, whose evaluation and typing rules are defined as follows (where $\mu\backslash\{\texttt{l}\}$ denotes the store identical to μ but with the location `l` removed from the domain):

$$\frac{\texttt{<t}, \mu\texttt{>} \longrightarrow \texttt{<t}', \mu'\texttt{>}}{\texttt{<free t}, \mu\texttt{>} \longrightarrow \texttt{<free t}', \mu'\texttt{>}} \qquad \text{(E-FREE)}$$

$$\frac{\texttt{l} \in \text{domain}(\mu)}{\texttt{<free l}, \mu\texttt{>} \longrightarrow \texttt{<()}, \mu\backslash\{\texttt{l}\}\texttt{>}} \qquad \text{(E-FREERED)}$$

$$\frac{\Gamma;\Sigma \vdash \texttt{t} : \texttt{Ref T}}{\Gamma;\Sigma \vdash \texttt{free t} : \texttt{True}} \qquad \text{(T-FREE)}$$

(a) Show a term $\texttt{t}$ in the language of the cheat sheet such that $\emptyset;\emptyset \vdash \texttt{t} : \texttt{T}$ but the evaluation of $\texttt{t}$ is eventually stuck. You may use the $\texttt{t}_1\texttt{;}\,\texttt{t}_2$ syntactic sugar if you want, to make your term more readable.

(b) Consider the Progress Theorem:

Theorem: If $\emptyset;\Sigma \vdash \texttt{t} : \texttt{T}$ and $\Sigma \vdash \mu$, then either $\texttt{t}$ is a value or there exists a term $\texttt{t}'$ and store μ' such that $<\texttt{t}, \mu> \longrightarrow <\texttt{t}', \mu'>$.

(Recall the definition of $\Sigma \vdash \mu$ from class: dom(Σ) = dom(μ) and for all locations $l \in$dom(Σ) we have $\emptyset;\Sigma \vdash \mu(\texttt{l}) : \Sigma(\texttt{l})$.)

Does Progress still hold? If so, just say so. If not, give a Σ, $\texttt{t}$, $\texttt{T}$, and μ that constitute a counterexample.

(c) Consider the Preservation Theorem:

Theorem: If $\emptyset;\Sigma \vdash \texttt{t} : \texttt{T}$ and $\Sigma \vdash \mu$ and $<\texttt{t}, \mu> \longrightarrow <\texttt{t}', \mu'>$, then there exists a Σ' such that $\emptyset;\Sigma' \vdash \texttt{t}' : \texttt{T}$ and $\Sigma' \vdash \mu'$.

Does Preservation still hold? If so, just say so. If not, give a Σ, $\texttt{t}$, $\texttt{T}$, μ, $\texttt{t}'$, and μ' that constitute a counterexample.

3. For each of the following two types, indicate whether the first can safely be allowed to be a subtype of the second. If yes, just say so. If no, provide a term in the language of the cheat sheet that would typecheck if we were to allow the two types to be in the subtype relation but would eventually get stuck at run time. You can assume that `Top` is a supertype of all types and that the type system has the following "subsumption" rule, which allows subtyping to be used anywhere:

$$\frac{\Gamma;\Sigma \vdash \texttt{t} \ \texttt{:}\ \texttt{T}' \qquad \texttt{T}' \ \texttt{<:}\ \ \texttt{T}}{\Gamma;\Sigma \vdash \texttt{t} \ \texttt{:}\ \texttt{T}} \qquad \text{(T-SUB)}$$

(a) `(Top → Ref Top) ∧ Bool` and `(Top → Top) ∧ Top`

(b) `(Ref Top) → Top` and `Top → Top`

(c) `Ref Top` and `Ref (Top ∧ Top)`

(d) `Ref (Top ∧ Top)` and `Ref Top`