Homework #4

CS231

Due by the end of the day on November 13. You should submit one file: `hw4.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. Recall the typing rules for `left t`, `right t`, and `match`, which are the terms related to tagged unions — see the cheat sheet.

   Write the corresponding *type inference* rules, of the form $\Gamma \vdash \texttt{t} \ : \ \texttt{T} \mid C$, for these three terms, where C is a set of type equality constraints of the form $\texttt{T}_1 = \texttt{T}_2$. You may assume that type variables ranged over by metavariable `X` are part of the syntax of types.

   Hint: Make sure your rules are such that each rule is applicable to every term of the corresponding form (i.e. a derivation is always possible); it simply may be that the resulting constraint set is unsatisfiable.

   As an example, here are the type inference rules for the lambda calculus with integers and addition:

$$\frac{\texttt{X fresh}}{\Gamma \vdash \texttt{n} \ : \ \texttt{X} \mid \{\texttt{X=Int}\}}$$

$$\frac{\texttt{X fresh} \qquad \Gamma \vdash \texttt{t}_1 \ : \ \texttt{T}_1 \mid C_1 \qquad \Gamma \vdash \texttt{t}_2 \ : \ \texttt{T}_2 \mid C_2}{\Gamma \vdash \texttt{t}_1 \ \texttt{+} \ \texttt{t}_2 \ : \ \texttt{X} \mid \{\texttt{X=Int}, \ \texttt{T}_1\texttt{=Int}, \ \texttt{T}_2\texttt{=Int}\} \cup C_1 \cup C_2}$$

$$\frac{\texttt{X fresh} \qquad \Gamma(\texttt{x}) = \texttt{T}}{\Gamma \vdash \texttt{x} \ : \ \texttt{X} \mid \{\texttt{X=T}\}}$$

$$\frac{\texttt{X fresh} \qquad \texttt{X}_1 \text{ fresh} \qquad \Gamma, \texttt{x:X}_1 \vdash \texttt{t} \ : \ \texttt{T}_2 \mid C}{\Gamma \vdash \texttt{function x -> t} \ : \ \texttt{X} \mid \{\texttt{X=X}_1\texttt{->T}_2\} \cup C}$$

$$\frac{\texttt{X fresh} \qquad \Gamma \vdash \texttt{t}_1 \ : \ \texttt{T}_1 \mid C_1 \qquad \Gamma \vdash \texttt{t}_2 \ : \ \texttt{T}_2 \mid C_2}{\Gamma \vdash \texttt{t}_1 \ \texttt{t}_2 \ : \ \texttt{X} \mid \{\texttt{T}_1\texttt{=T}_2\texttt{->X}\} \cup C_1 \cup C_2}$$

2. Consider the simply-typed lambda calculus augmented with explicit parametric polymorphism (see Sections 2 and 4 of the cheat sheet) and assume we also have the unit value `()` and the unit type `Unit` as well as the value `true` and the type `Bool`. For each type environment Γ and term `t` below, replace all occurrences of `?T?` with types such that there exists a type `T` whereby $\Gamma \vdash \texttt{t} \ : \ \texttt{T}$. Show both the rewritten term and the type `T`. If there is no such rewritten term and type, just say so.

   (a) $\Gamma = \emptyset$

   `t =` `(function X -> function x:X -> x) ?T? true`

(b) $\Gamma = \emptyset$
```
t =
function x:?T? -> ((function y:Unit -> (x Bool true)) (x Unit ()))
```

(c) $\Gamma = \emptyset$
```
t =
function x:?T? -> ((function y:Bool -> (x Bool true)) (x Unit ()))
```

(d) Γ = {app:∀X.∀X′.(X → X′) → X → X′}
```
t = app ?T? ?T? (function x:Unit -> true)
```

(e) Γ = {app:∀X.∀X′.(X → X′) → X → X′}
```
t = app ?T? ?T? app
```

(f) Γ = {app:∀X.∀X′.(X → X′) → X → X′}
```
t = app ?T? ?T? (app ?T? ?T?)
```

3. In each of the following programs (which are written in a simply-typed lambda calculus augmented with mutable references, `let`, the unit value `()` and sequencing, and integers with addition and multiplication), replace the occurrence of `...` with a term `t` so that the program evaluates to 42? *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 ())
```

4. 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 $\mu\backslash\{\mathtt{l}\}$ denotes the store identical to μ but with the location `l` removed from the domain):

$$\frac{\mathtt{t} \mid \mu \longrightarrow \mathtt{t}' \mid \mu'}{\mathtt{free\ t} \mid \mu \longrightarrow \mathtt{free\ t}' \mid \mu'} \qquad \text{(E-FREE)}$$

$$\frac{\mathtt{l} \in \mathit{dom}(\mu)}{\mathtt{free\ l} \mid \mu \longrightarrow \mathtt{unit} \mid \mu\backslash\{\mathtt{l}\}} \qquad \text{(E-FREELOC)}$$

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

(a) Show a term `t` in this language such that $\emptyset;\emptyset \vdash \mathtt{t} : \mathtt{T}$ but the evaluation of `t` is eventually stuck. You may use the $\mathtt{t}_1;\mathtt{t}_2$ and `let x =` $\mathtt{t}_1$ `in` $\mathtt{t}_2$ syntactic sugars if you want, to make your term more readable.

(b) Consider the Progress Theorem:

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

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

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

(c) Consider a modified form of the Preservation Theorem, which removes the requirement that $\Sigma' \supseteq \Sigma$:

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

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