98-317 Homework 8: Subtyping

Email solutions to cjwong@andrew.cmu.edu by the end of 2019-10-22.

Solve at least one of the following problems to the best of your ability. Incorrect or incomplete answers will be accepted if a reasonable attempt is present. *Please* do not spend more than 30 minutes on this homework if you don't want to. Post on Piazza if you need help or would like to discuss further.

1. When discussing the subtyping semantics of records, we settled on a *width subtyping* rule:

   $$\{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n, \ldots\} <: \{\ell_1 : \tau_1, \ldots, \ell_n : \tau_n\}$$

   i.e. $\rho_1 <: \rho_2$ when every label and type in ρ_2 is also in ρ_1.

   We might construct a similar rule for simple (unlabeled) products:

   $$\tau_1 \times \tau_2 \times \tau_3 <: \tau_1 \times \tau_2$$

   which is largely inoffensive, as we can treat an unlabeled record as a record labeled by indices in the resulting tuple (0, 1, 2, etc).

   As records with one field are isomorphic to the type of that field, we may be tempted to add two more rules as "reduced forms" of the above:

   $$\tau_1 \times \tau_2 <: \tau_1 \qquad\qquad \tau_1 \times \tau_2 <: \tau_2$$

   Is this a good idea? Why or why not?

2. Show that `'a ref` is not covariant. In other words, show that $\tau_1 <: \tau_2$ is not enough to conclude that τ_1 `ref` $<: \tau_2$ `ref`, where `'a ref` is the type of mutable reference cells from SML.

3. Let `t set` be the type representing sets whose elements are of type `t`, and suppose that values of this type are immutable. In particular, any encoding of the type `t set` must support `mem: t set -> t -> bool`, returning whether a given object is in the set.

   Intuitively, we might think that `set` should be covariant – if $\tau_1 <: \tau_2$, we can view all elements of a τ_1 `set` as values of type τ_2, giving us a τ_2 `set`.

   One reasonable representation of the datatype `'a set` is

   ```
   'a set = 'a -> bool
   ```

   encoding a given set as its membership function. However, this would make the type `'a set` *contravariant* in `'a`. Explain this seeming disconnect.