Assignment 11
Semantics, WS 2007/08

Prof. Dr. Gert Smolka, Dr. Jan Schwinghammer
www.ps.uni-sb.de/courses/sem-ws07/

Read Chapters 11.8, 11.10 and 15 of Pierce's TAPL.

Exercise 11.1 (Record and variant types) Show how to obtain product types as a special case of record types. Similarly, show how to obtain binary sums as a special case of variant types.

Exercise 11.2 (Failure of unique types and unique typing derivations) Consider the simply typed lambda calculus with record types, procedure types and subtyping.

a) Give Γ, t, T and T' where $T \neq T'$, such that $\Gamma \vdash t : T$ and $\Gamma \vdash t : T'$.

b) Give Γ, t and T such that there is more than one derivation of $\Gamma \vdash t : T$.

Exercise 11.3 (Partial orders) Give examples (for instance, by drawing Hasse diagrams) of partial orders as follows:

a) a finite order with a greatest and 2 minimal elements;

b) a finite order with 2 maximal and 3 minimal elements;

c) a finite order with two of its elements possessing an upper bound but no least upper bound;

d) an infinite order where every chain $a_1 \leq a_2 \leq \ldots$ is eventually stationary (i.e., there is no infinite ascending chain);

e) an order without maximal and minimal elements.

Exercise 11.4 (Partial (pre-) order of types) Consider the simply typed lambda calculus with record types, greatest type *Top*, and subtyping.

a) Show that the relation $<:$ is only a *partial* preorder but not a preorder: find types S and T such that neither $S <: T$ nor $T <: S$ holds. Will $<:$ still be partial in the calculus without records?

b) Show that the relation $<:$ is only a partial *pre*order but not a partial order: find types S and T such that $S <: T$ and $T <: S$ holds but $S \neq T$.

c) Find an infinite chain of procedure types $T_1, T_2, \ldots$ such that $T_1 <: T_2 <: \ldots$

d) Find an infinite chain of procedure types $S_1, S_2, \ldots$ such that $\ldots <: S_2 <: S_1$.

e) Is there a procedure type that is a supertype of every other procedure type? Does the addition of a least type *Bot* make a difference?

f) Is there a procedure type that is a subtype of every other procedure type? Does the addition of a least type *Bot* make a difference?

g) Give all supertypes of $\{l_1 : Top, l_2 : Top\}$.

h) Find an infinite chain of record types $T_1, T_2, \ldots$ where all fields have type *Top*, and such that $\ldots <: T_2 <: T_1$.

i) Find an infinite chain of record types $S_1, S_2, \ldots$ where only a single label is used, and such that $\ldots <: S_2 <: S_1$.

j) Is there a record type that is a supertype of every other record type?

k) Is there a record type that is a subtype of every other record type? Does it make a difference if the set $\mathcal{L}ab$ of labels is finite or infinite? Does the addition of a least type *Bot* make a difference?

Exercise 11.5 (Computing GLBs and LUBs) Consider the simply typed lambda calculus with record types, procedure types, *Top*, *Bot* and subtyping.

a) Give least upper and greatest lower bounds of $\{l_1{:}T_1, l_2{:}T_2\}$ and $\{l_2{:}T_2, l_3{:}T_3\}$.

b) Give least upper and greatest lower bounds of $Bot \to Bot$ and $Top \to Top$.

c) Write (mutually recursive) procedures $glb, lub : Ty \to Ty \to Ty$ that compute a greatest lower bound and least upper bound, resp., of two types.

Exercise 11.6 (Invariance of reference types) Consider the simply typed lambda calculus with references and subtyping. Show that invariance of reference types is necessary for type safety:

a) Give a term that is well-typed under the assumption $S <: T \Rightarrow Ref\ T <: Ref\ S$ but that will fail with a run-time type error (i.e., its evaluation will get stuck).

b) Give a term that is well-typed under the assumption $S <: T \Rightarrow Ref\ S <: Ref\ T$ but that will fail with a run-time type error.

Exercise 11.7 (Subtyping for trees, arrays and continuations) Propose inference rules to define the subtype relation with respect to the following extensions to the simply typed lambda calculus with subtyping:

a) a type *BTree T* of binary trees with nodes labelled by values of type T;

b) a type *Array T* of mutable arrays that store values of type T;

c) the type *Cont T* of T-accepting continuations.

In each case, give an (informal) argument why your proposed rule is sound.

Exercise 11.8 (Type isomorphisms) Show that 1 and $\{\}$ are equivalent, in the sense that there are mutually inverse maps, represented by lambda terms $t : 1 \to \{\}$ and $t' : \{\} \to 1$. Likewise, show that $\{l_1{:}T_1, l_2{:}T_2, l_3{:}T_3\}$ and $\{l_2{:}T_2, l_1{:}T_1, l_3{:}T_3\}$ are equivalent in this sense, and also $\{l_1{:}T_1, l_2{:}T_2, l_3{:}T_3\}$ and $(T_1 \times T_2) \times T_3$.