Solutions to CS 520 Problem Set 7

Problems 1-4 by Yarom, Problems 5,6 Ashwin

Problem 1, Exercise 16.1.2 [Pierce]

Proof for part 1: by induction on the structure of type S.

- $S = Top$: We can derive $Top <: Top$ using S-Top.
- $S = S_1 \to S_2$: According to the induction hypothesis, we can derive both $S_1 <: S_1$ and $S_2 <: S_2$ without using S-Refl. $S_1 <: S_1$ and $S_2 <: S_2$ implies $S_1 \to S_2 <: S_1 \to S_2$ using S-Arrow.
- $S = \{l_i : S_i{}^{i \in 1..n}\}$: By the induction hypothesis we can derive $S_i <: S_i$ without using S-Refl for $i \in 1..n$. Now using S-RCD, we can make the following derivation:

$$\frac{\{l_i{}^{i \in 1..n}\} \subseteq \{l_i{}^{i \in 1..n}\} \quad l_i = l_i \textit{ implies } S_i <: S_i}{\{l_i : S_i{}^{i \in 1..n}\} <: \{l_i : S_i{}^{i \in 1..n}\}} \ (S - RCD)$$

Problem 2, Exercise 16.1.3 [Pierce]

With the introduction of these new base types the lemma does not hold. We have to add new rules that express the connection between the different basic types.

- $Bool <: Bool$
- $Char <: Char$
- $Nat <: Nat$
- $Bool <: Nat$
- $Char <: Nat$

Problem 3, Exercise 16.2.3 [Pierce]

Need to show terms s with algorithmic type S and t with algorithmic type T s.t. $s \rightarrow^* t$ and $T <: S$, but $S \not<: T$.

$s = (\lambda x : \{\}.x)\ \{x = 0\}$
$t = \{x = 0\}$
$S = \{\}$
$T = \{x : Nat\}$

Problem 4

1. Types T and U s.t. $T \neq U$ and each is a subtype of the other.
   $T = \{l_1 : Top\ ,\ l_2 : Top \rightarrow Top\}$
   $U = \{l_2 : Top \rightarrow Top\ ,\ l_1 : Top\}$

2. Rules for equivalence relation:

   $Top \equiv Top\ (EQ - Top)$

   $$\frac{S_1 \equiv T_1 \quad S_2 \equiv T_2}{S_1 \rightarrow S_2 \equiv T_1 \rightarrow T_2}\ (EQ - Arrow)$$

   $$\frac{\{{l_i}^{i \in 1..n}\}\ permutation\ of\ \{{k_j}^{j \in 1..n}\} \quad l_i = k_j\ implies\ T_i \equiv S_j}{\{l_i : {T_i}^{i \in 1..n}\} \equiv \{k_j : {S_j}^{j \in 1..n}\}}\ (EQ - RCD)$$

   We can also add the following rules to express Reflexivity, Symmetry, and Transitivity, although these rules can be inferred from the above rules.

   $T \equiv T\ (EQ - Refl)$

   $$\frac{T_1 \equiv T_2}{T_2 \equiv T_1}\ (EQ - Sym)$$

   $$\frac{T \equiv U \quad U \equiv S}{T \equiv S}\ (EQ - Trans)$$

3. $T \equiv U$ iff $T <: U$ and $U <: T$

   Proof of $\Longleftarrow$ by induction of the structure of type T.

   - $T = Top$: then $Top <: U$. U here must be Top too (Top is the only supertype of Top). By EQ-Top we can derive $T \equiv U$.

- $T = T_1 \to T_2$: by the assumption, $T <: U$ and $U <: T$, U has the form $U = U_1 \to U_2$. From $T <: U$, we know $U_1 <: T_1$ and $T_2 <: U_2$ (using S-Arrow). From $U <: T$, we know $T_1 <: U_1$ and $U_2 <: T_2$ (again S-Arrow).
  By induction hypothesis $T_1 <: U_1$ and $U_1 <: T_1$ implies $T_1 \equiv U_1$, and in the same way $U_2 \equiv T_2$. Now applying EQ-Arrow we can derive $T \equiv U$.
- $T = \{l_i : T_i{}^{i \in 1..n}\}$: by the assumption, $T <: U$ and $U <: T$, U has the form $U = \{k_j : U_j{}^{j \in 1..n}\}$ where $\{l_i{}^{i \in 1..n}\}$ is a permutation of $\{k_j{}^{j \in 1..n}\}$. From $U <: T$ and S-RCD we know, $k_j = l_i$ implies $U_j <: T_i$. From $T <: U$ and S-RCD we know, $l_i = k_j$ implies $T_i <: U_j$. Then, $l_i = k_j$ implies both $T_i <: U_j$ and $U_j <: T_i$. By applying the induction hypothesis on each one of the subderivations we can say $l_i = k_j$ implies $T_i \equiv U_j$ and by applying EQ-RCD we conclude $T \equiv U$

Proof of $\Longrightarrow$ by induction of the structure of type T.

- $T = Top$: by looking at the equivalence rules (EQ-Top, EQ-Arrow, EQ-RCD), we see that in order to derive $TOP \equiv U$, we have to use EQ-Top. Hence, $U = Top$ too. The result is then immediately obtained.
- $T = T_1 \to T_2$: by the equivalence rules, and EQ-Arrow in particular, we see U has the form $U = U_1 \to U_2$ in order to derive $U \equiv T$. According to EQ-Arrow, we can also infer $U_1 \equiv T_1$ and $U_2 \equiv T_2$. By applying the induction hypothesis twice we get,
  $U_1 <: T_1$ and $T_1 <: U_1$
  $U_2 <: T_2$ and $T_2 <: U_2$
  Now, by applying S-Arrow twice we conclude $U_1 \to U_2 <: T_1 \to T_2$ and $T_1 \to T_2 <: U_1 \to U_2$
- $T = \{l_i : T_i{}^{i \in 1..n}\}$: by the equivalence rules, and EQ-RCD in particular, we see U has the form $U = \{k_j : U_j{}^{j \in 1..n}\}$ where $\{l_i{}^{i \in 1..n}\}$ is a permutation of $\{k_j{}^{j \in 1..n}\}$. According to EQ-RCD, we can also infer $l_i = k_j$ implies $T_i \equiv U_j$. By applying the induction hypothesis on each subderivation we get $T_i \equiv U_j$ implies both $T_i <: U_j$ and $U_j <: T_i$. Using S-RCD twice we conclude $T <: U$ and $U <: T$

Problem 5

Add the following to the algorithmic typing fucntion `typeof(G,t)`,

IF $t = \{l_1 = t_1, l_2 = t_2, ..., l_n = t_n\}$ THEN
   $T = \{\}$
   FOR i = 1 to n
      IF typeof(G,t_i) = T_i THEN
         $T = T \cup \{l_i : T_i\}$
      ELSE
         ''error''
   END-FOR
   return T
END-IF

IF $t = t'.l_i$ THEN
   IF typeof(G,t') = $\{l'_1 : T'_1, l'_2 : T'_2, ..., l'_m : T'_m\}$ THEN
      IF $l_i \in \{l'_1, l'_2, ..., l'_m\}$ THEN
         let j such that $l_i = l'_j$
         return T'_j
      ELSE ''error''
   ELSE ''error''
END-IF

Problem 6

New lines for join algorithm,

1. $Char$ if $S = T = Char$
2. Nat if $S = T = Nat$
3. Nat if $S = Nat$ and $T \in \{Bool, Char\}$.
   Similar rule with S and T reversed
4. Nat if $S = Bool$ and $T = Char$.
   Similar rule with S and T reversed

New lines for meet algorithm,

1. $Char$ if $S = T = Char$
2. Nat if $S = T = Nat$
3. $Bool$ if $S = Bool$ and $T = Nat$.
   Similar rule with S and T reversed

4. $Char$ if $S = Char$ and $T = Nat$.
   Similar rule with S and T reversed

Lemma A.10: If $L <: S$ and $L <: T$, then $S \wedge T = M$ for some M.
Changes to lemma A.10: Following cases are added to the second paragraph of the proof.

- Case $S = T = Bool$ handled by Pierce.
- Case $S = T = Char \implies S \wedge T = Char$ by first new definition in meet.
  Similar for $S = T = Nat$.
- Case $S = Char$ and $T = Nat \implies S \wedge T = Char$ by fourth new rule.
- Other cases can be enumerated in a similar fashion and seen to match the new definitions in meet.

Lemma A.11 does not need to be altered.
Lemma A.12 part(1): Suppose that $S \vee T = J$ and for some U, that $S <: U$ and $T <: U$. Then $J <: U$.
Following cases are added to second paragraph,

- Case $U = Bool$: already handled in Pierce.
- Case $U = Char$: inversion lemma implies $S = T = Char$. Hence $J = S \vee T = Char$ and $J <: U$.
- Case $U = Nat$: inversion lemma implies $S \in \{Char, Bool, Nat\}$ and $T \in \{Char, Bool, Nat\}$. In these cases $J = S \vee T \in \{Char, Bool, Nat\}$ and hence $J <: U$.

Lemma A.12 part(2): Suppose that $S \wedge T = M$ and for some L, that $L <: S$ and $L <: T$. Then $L <: M$.
Following cases are added to first paragraph,

- Case $L = Bool$: inversion lemma implies $S \in \{Bool, Nat, Top\}$ and $T \in \{Bool, Nat, Top\}$. Hence $M = S \wedge T \in \{Bool, Nat, Top\}$. Clearly $L <: M$.
- Case $L = Char$: similar to $Bool$ case.
- Case $L = Nat$: inversion lemma implies $S \in \{Nat, Top\}$ and $T \in \{Nat, Top\}$ and hence $M \in \{Nat, Top\}$. Clearly $L <: M$.