COMPUTER SCIENCE 520 (FALL TERM, 2006)
PRINCIPLES OF PROGAMMING LANGUAGES

Solutions for End-of-Term Examination

DECEMBER 16, 2006

Problem 1. ***Type Reconstruction and Unification (20 points).*** We consider typability of terms of the pure lambda-calculus in the system of simple types (in the first 2 parts), denoted $\boldsymbol{\lambda}_{\rightarrow}$ as usual, and in the same calculus augmented with polymorphic `let`, denoted here $\boldsymbol{\lambda}^{\mathtt{let}}_{\rightarrow}$ (in the last 2 parts).

(5 pts) Write the set of constraints, call it C_1, corresponding to the typability of the term $(\lambda \mathtt{x}.\, \mathtt{x}\,\mathtt{x})$ in $\boldsymbol{\lambda}_{\rightarrow}$.

> $C_1 = \{\mathtt{X} \doteq \mathtt{X} \rightarrow \mathtt{A}\}$ where $\mathtt{X}$ corresponds to (unknown type of) the term variable $\mathtt{x}$.

Does C_1 have a solution? If yes, write down the solution. If no, justify your answer carefully.

> No. C_1 has no solution, i.e., there is no substitution S such that $S(\mathtt{X}) = S(\mathtt{X} \rightarrow \mathtt{A})$.

(5 pts) Write the set of constraints, call it C_2, corresponding to the typability of the term $(\lambda \mathtt{x}.\, \mathtt{x}\,\mathtt{x})\,(\lambda \mathtt{y}.\, \lambda \mathtt{z}.\, \mathtt{y}\,)$ in $\boldsymbol{\lambda}_{\rightarrow}$.

> $C_2 = \{\mathtt{X} \doteq \mathtt{X} \rightarrow \mathtt{A}, \mathtt{X} \rightarrow \mathtt{A} \doteq \mathtt{Y} \rightarrow \mathtt{Z} \rightarrow \mathtt{Y}\}$ where $\mathtt{X}, \mathtt{Y}$ and $\mathtt{Z}$ correspond to (the unknown types of) the term variables $\mathtt{x}, \mathtt{y}$ and $\mathtt{z}$, respectively.

Does C_2 have a solution? If yes, write down the solution. If no, justify your answer carefully.

> No. The first constraint in C_2 does not have a solution, so C_2 does not have a solution either.

(5 pts) Write the set of constraints, call it C_3, corresponding to the typability of the term `let` $\mathtt{x} = (\lambda \mathtt{y}.\, \lambda \mathtt{z}.\, \mathtt{y}\,)$ `in` $\mathtt{x}\,\mathtt{x}$.

> $C_3 = \{\mathtt{Y}_1 \rightarrow \mathtt{Z}_1 \rightarrow \mathtt{Y}_1 \doteq (\mathtt{Y}_2 \rightarrow \mathtt{Z}_2 \rightarrow \mathtt{Y}_2) \rightarrow \mathtt{A}\}$ where $\mathtt{Y}_1$ and $\mathtt{Z}_1$ (respectively, $\mathtt{Y}_2$ and $\mathtt{Z}_2$) correspond to the unknown types of term variables $\mathtt{y}$ and $\mathtt{z}$ after substituting $(\lambda \mathtt{y}.\, \lambda \mathtt{z}.\, \mathtt{y}\,)$ for the first occurrence of $\mathtt{x}$ (respectively, the second occurrence of $\mathtt{x}$) in the body of the `let` expression.

Does C_3 have a solution? If yes, write down the solution. If no, justify your answer carefully.

> Yes. C_3 has a solution: $S(\mathtt{Y}_1) = \mathtt{Y}_2 \rightarrow \mathtt{Z}_2 \rightarrow \mathtt{Y}_2$ and $S(\mathtt{A}) = \mathtt{Z}_1 \rightarrow \mathtt{Y}_1$.

(5 pts) Write the set of constraints, call it C_4, corresponding to the typability of the term `let` $\mathtt{x} = (\lambda \mathtt{m}.\, \lambda \mathtt{n}.\, \mathtt{m}\;(\mathtt{m}\;\mathtt{n})\;)(\lambda \mathtt{y}.\, \lambda \mathtt{z}.\, \mathtt{y}\;)$ `in` $\mathtt{x}\,\mathtt{x}$.

> $C_4 = C_{4,1} \cup C_{4,2}$ where
>
> $$C_{4,i} = \{\mathtt{M}_i \doteq \mathtt{N}_i \rightarrow \mathtt{A}_i, \mathtt{M}_i \doteq \mathtt{A}_i \rightarrow \mathtt{B}_i, \mathtt{M}_i \rightarrow \mathtt{N}_i \rightarrow \mathtt{B}_i \doteq (\mathtt{Y}_i \rightarrow \mathtt{Z}_i \rightarrow \mathtt{Y}_i) \rightarrow \mathtt{C}_i\}$$
>
> for $i \in \{1, 2\}$, where $\mathtt{M}_i$, $\mathtt{N}_i$, $\mathtt{Y}_i$ and $\mathtt{Z}_i$ with $i = 1$ (respectively, with $i = 2$) correspond to the unknown types of term variables $\mathtt{m}$, $\mathtt{n}$, $\mathtt{y}$ and $\mathtt{z}$ after substituting $(\lambda \mathtt{m}.\, \lambda \mathtt{n}.\, \mathtt{m}\;(\mathtt{m}\;\mathtt{n})\;)(\lambda \mathtt{y}.\, \lambda \mathtt{z}.\, \mathtt{y}\;)$ for the first occurrence of $\mathtt{x}$ (respectively, the second occurrence of $\mathtt{x}$) in the body of the `let` expression.

Does C_4 have a solution? If yes, write down the solution. If no, justify your answer carefully.

No. C_4 does not have a solution. Each of $C_{4,1}$ and $C_{4,2}$ by itself does not have a solution, so that C_4 does not either. Consider $C_{4,1}$, the same argument applies to $C_{4,2}$. Its first two constraints give rise to the following constraints $\{\mathtt{M}_1 \doteq \mathtt{A}_1 \to \mathtt{A}_1, \mathtt{N}_1 \doteq \mathtt{A}_1, \mathtt{B}_1 \doteq \mathtt{A}_1\}$, for which there is a solution: $S(\mathtt{M}_1) = \mathtt{A}_1 \to \mathtt{A}_1$, $S(\mathtt{N}_1) = \mathtt{A}_1$ and $S(\mathtt{B}_1) = \mathtt{A}_1$. Applying the substitution S to the third constraint in $C_{4,1}$, we obtain:

$$(\mathtt{A}_1 \to \mathtt{A}_1) \to \mathtt{A}_1 \to \mathtt{A}_1 \doteq (\mathtt{Y}_1 \to \mathtt{Z}_1 \to \mathtt{Y}_1) \to \mathtt{C}_1$$

which gives rise to the constraint $\mathtt{A}_1 \to \mathtt{A}_1 \doteq \mathtt{Y}_1 \to \mathtt{Z}_1 \to \mathtt{Y}_1$ which has no solution.

Problem 2. ***Simple Types vs. Universal Types (10 points).*** We compare typability of terms of the pure lambda-calculus in two type systems: the system of simple types (denoted $\boldsymbol{\lambda}_{\to}$) and the system of simple types augmented with universal types (denoted here $\boldsymbol{\lambda}_{\to,\forall}$). The latter is also called System F. Justify your answers carefully, in at most 3 lines. If your answer is ***yes***, provide an example of such a term, possibly with an explanation if you so desire. If your answer is ***no***, provide a careful explanation.

(5 pts) Is there a term of the pure lambda-calculus which is typable in the system of universal types, but not in the system of simple types.

Yes. The term $\lambda \mathtt{x}.\,\mathtt{x}\,\mathtt{x}$ is typable in the system of universal types, but not in the system of simple types.

(5 pts) Is there a term of the pure lambda-calculus which is typable in the system of simple types, but not in the system of universal types.

No. The system of simple types is a subsystem of the system of universal types, i.e., every typing derivation in the first is a legal typing derivation in the second.

Problem 3. ***Positive Recursive Types vs. Recursive Types (10 points).*** Justify your answers carefully, in at most 3 lines. If your answer is ***yes***, provide an example of such a term, possibly with an explanation if you so desire. If your answer is ***no***, provide a careful explanation.

(5 pts) Is there a term of the pure lambda-calculus which is typable in the system of recursive types, but not in the system of positive recursive types.

Yes. Every term of the pure lambda-calculus is typable in the system of recursive types, in particular, every term which is not strongly normalizable, such as $(\lambda \mathtt{x}.\,\mathtt{x}\,\mathtt{x})\,(\lambda \mathtt{x}.\,\mathtt{x}\,\mathtt{x})$. On the other hand, every term of the pure lambda-calculus which is typable in the system of positive recursive types is strongly normalizable, which implies $(\lambda \mathtt{x}.\,\mathtt{x}\,\mathtt{x})\,(\lambda \mathtt{x}.\,\mathtt{x}\,\mathtt{x})$ is not typable in it.

(5 pts) Is there a term of the pure lambda-calculus which is typable in the system of positive recursive types, but not in the system of recursive types.

No. The system of positive recursive types is a subsystem of the system of universal types. Alternatively, not all terms of the pure lambda-calculus are typable in the system of positive recursive types, whereas all terms of the pure lambda-calculus are typable in the system of recursive types.

Problem 4. ***Congruence Relations (20 points).*** In lecture and lecture notes we defined $\approx$ as the least congruence on recursive types satisfying two axioms:

1. $\boxed{\mu X.T \;\approx\; \mu Y.[X \mapsto Y]T}$
   i.e., if two recursive types are the same up to renaming of bound variables then the two are congruent.

2. $\boxed{\mu X.T \;\approx\; [X \mapsto \mu X.T]T}$
   i.e., if a recursive type (on the right of $\approx$) is obtained from another recursive type (on the left of $\approx$) by "unwinding" once the loop corresponding to the μ-binding, then the two are congruent.

We want to extend the congruence $\approx$ in four different ways. In each of the first 3 cases, we state the congruence in English, together with an example, and we want you to write the corresponding axiom formally (in the style of the grayboxes above). If you wish, you may add a short explanation.

3. If a recursive type is obtained from another recursive type by permuting two adjacent μ-bindings, then the two types are congruent. For example, we want to be able to derive:

   $\mu\mathtt{X}.\mu\mathtt{Y}.\mathtt{X} \rightarrow (\mathtt{Nat} \times \mathtt{Y}) \approx \mu\mathtt{Y}.\mu\mathtt{X}.\mathtt{X} \rightarrow (\mathtt{Nat} \times \mathtt{Y})$

   (2 pts) $\boxed{\mu X.\mu Y.T \;\approx\; \mu Y.\mu X.T}$

4. If a recursive type is obtained from another recursive type by merging two adjacent μ-bindings, then the two types are congruent. For example, we want to be able to derive:

   $\mu\mathtt{X}.\mu\mathtt{Y}.\,\mathtt{X} \rightarrow (\mathtt{Nat} \times \mathtt{Y}) \approx \mu\mathtt{X}.\,\mathtt{X} \rightarrow (\mathtt{Nat} \times \mathtt{X})$

   (2 pts) $\boxed{\mu X.\mu Y.T \;\approx\; \mu X.[Y \mapsto X]T}$

5. If a recursive type is obtained from another recursive type by permuting the two components of a product, then the two types are congruent. For example, we want to be able to derive:

   $\mathtt{Nat} \times \mathtt{Bool} \approx \mathtt{Bool} \times \mathtt{Nat}$

   (2 pts) $\boxed{T_1 \times T_2 \;\approx\; T_2 \times T_1}$

In addition to the preceding 5 axioms, we consider an additional one:

6. $\boxed{\mu X.T_1 \rightarrow T_2 \rightarrow T_3 \;\approx\; T_1 \rightarrow \mu X.T_2 \rightarrow T_3 \qquad \text{where } X \notin FTV(T_1)}$

Let A be a subset of the 6 axioms above, i.e., $A \subseteq \{1,2,3,4,5,6\}$. For arbitrary recursive types T_1 and T_2, we write $T_1 \approx_A T_2$ iff there is a derivation to show that T_1 is congruent to T_2 using only axioms in A. Thus, $\approx_{\{1,2\}}$ is the same as $\approx$ presented in lecture and in lecture notes.

If T is a recursive type, let $\lceil T \rceil$ denote the possibly infinite type (or parse tree) obtained by repeatedly unwinding all the μ-bindings in T. We say that a set A of axioms is *sound* if $T_1 \approx_A T_2$ implies $\lceil T_1 \rceil = \lceil T_2 \rceil$.

(10 pts) Determine which of the following sets of axioms is sound, justifying your answer in at most 3 lines:

(4 pts) $A = \{1,2,3\}$

Sound. Permuting adjacent μ bindings does not affect the infinite type obtained by repeatedly unwinding them.

(4 pts) $A = \{1, 2, 4\}$

Sound. Merging adjacent μ bindings does not affect the infinite type obtained by repeatedly unwinding them.

(2 pts) $A = \{1, 2, 5\}$

Unsound.

(4 pts) Show that the set of axioms $A = \{1, 2, 6\}$ is not sound, justifying your answer in at most 3 lines.

Consider the recursive type $T \triangleq \mu\mathtt{X}.\mathtt{Nat} \to \mathtt{Nat} \to \mathtt{X} \to \mathtt{Nat}$ which, according to axiom 6, is congruent to the recursive type $T' \triangleq \mathtt{Nat} \to \mu\mathtt{X}.\mathtt{Nat} \to \mathtt{X} \to \mathtt{Nat}$. But $\lceil T \rceil \neq \lceil T' \rceil$.

Problem 5. ***Subtyping Relation (20 points).*** For each of the 8 pairs of types below, T and U, determine which of the 4 answers $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$ holds.

A. $T <: U$ but not $U <: T$.

B. $U <: T$ but not $T <: U$.

C. Both $T <: U$ and $U <: T$.

D. Neither $T <: U$ nor $U <: T$.

In each case, there is exactly one correct answer. Assume $\mathtt{Nat} <: \mathtt{Real}$.

1. $T = \mathtt{Nat} \to \mathtt{Bool}$ and $U = \mathtt{Nat} \times \mathtt{Bool}$.
   Answer: D
2. $T = (\mathtt{Nat} \to \mathtt{Nat}) \to \mathtt{Nat}$ and $U = (\mathtt{Real} \to \mathtt{Nat}) \to \mathtt{Real}$.
   Answer: A
3. $T = \mathtt{Nat} \to \mathtt{Nat} \to \mathtt{Nat}$ and $U = \mathtt{Real} \to \mathtt{Nat} \to \mathtt{Real}$.
   Answer: D
4. $T = \{\mathtt{a} : \mathtt{Nat}, \mathtt{b} : \mathtt{Bool}, \mathtt{c} : \mathtt{Nat}\}$ and $U = \{\mathtt{c} : \mathtt{Nat}, \mathtt{a} : \mathtt{Nat}, \mathtt{b} : \mathtt{Bool}\}$.
   Answer: C
5. $T = \{\mathtt{a} : \mathtt{Nat}, \mathtt{b} : \mathtt{Bool}\}$ and $U = \{\mathtt{c} : \mathtt{Nat}, \mathtt{d} : \mathtt{Bool}\}$.
   Answer: D
6. $T = \{\,\}$ and $U = \{\mathtt{a} : \mathtt{Top}\}$.
   Answer: B
7. $T = \{\,\}$ and $U = \mathtt{Top}$.
   Answer: A
8. $T = \mathtt{Top}$ and $U = \{\mathtt{a} : \mathtt{Top}\}$.
   Answer: B