UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

TYPES AND PROGRAMMING LANGUAGES

Specimen Questions
February 2005

Fourth Year Courses

INSTRUCTIONS TO CANDIDATES

This paper contains two specimen questions.
The real exam paper will have three questions.
In the exam, you will be asked to choose to answer two questions out of the three.
The time allowed will be 1hr45 minutes.

Solution hints:

1. See Chapter 15 of *Types and Programming Languages* and Cardelli's *Type Systems* survey paper.
2. Study the language FOFL studied in lectures and the practical assignments; Chapters 11 and 13 of *Types and Programming Languages* are also relevant.

1. Consider the type $\texttt{Ref}\ T$ of references to values of type T. The typing rules for the relevant term constructors are:

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash \texttt{ref}\ t : \texttt{Ref}\ T} \qquad \frac{\Gamma \vdash t : \texttt{Ref}\ T}{\Gamma \vdash\ !\, t : T} \qquad \frac{\Gamma \vdash t : \texttt{Ref}\ T \qquad \Gamma \vdash t' : T}{\Gamma \vdash t{:=}t' : \texttt{Unit}}$$

   (a) Some functional languages (e.g., Haskell) do not include a type of references, while others (e.g., OCaml) do. Briefly mention what are the advantages and disadvantages of adding reference types to a functional language. [*5 marks*]

   (b) A type of (imperative) arrays can be given in terms of reference types, a natural numbers type `Nat` and product types, as

$$\texttt{Array}\ T \ = \ \texttt{Nat} \times (\texttt{Nat} \to \texttt{Ref}\ T)$$

   In terms of this definition, give typings for terms for the following array operations:

   - declaring an array;
   - indexing into an array;
   - updating an element of an array;
   - returning the bound of an array.

   Give (correctly-typed) derived terms for each of the above operations using the underlying term constructors for the type definition. You may assume terms for numeric comparisons and a term $\texttt{error}_T$ for an array indexing error. [*5 marks*]

   (c) Consider the extension of the language with subtyping. Explain why references must be considered *invariant* with respect to the subtyping relation. [*4 marks*]

   (d) Java considers arrays to be covariant with respect to the subtyping relation. In terms of the above type definition for $\texttt{Array}\ T$, and assuming the presence of Java's `instanceof` operator, give modified definitions of the array operations above which preserve safety. [*4 marks*]

   (e) Explain (by giving subtyping rules and informal justification) how the array type $\texttt{Array}\ T$ can be split into two subtypes, $\texttt{ReadArray}\ T$ and $\texttt{WriteArray}\ T$ which are respectively covariant and contravariant in terms of the subtyping relation. Show how instances of the transitivity of subtyping involving array types can be eliminated. [*7 marks*]

2. (a) The syntactic notion of type safety in terms of *preservation* and *progress* theorems is applied to languages with a small-step semantics.

   i. State the usual theorems. [*2 marks*]

   ii. In languages with big-step semantics, the evaluation relation has the form $t \Downarrow v$. What is the safety theorem for this form of language semantics? [*2 marks*]

   iii. Compare these two alternatives, giving some advantages and disadvantages of the small-step approach. [*2 marks*]

   (b) Recall the FOFL language whose programs consist of mutually recursive first-order functions. A useful extension of FOFL is to add global value declarations to programs. The syntax is as follows:

$$
\begin{array}{lcl}
t & ::= & \texttt{true} \mid \texttt{false} \mid 0 \mid \texttt{succ}\, t \mid \texttt{pred}\, t \mid \texttt{iszero}\, t \\
 & & \quad \mid \texttt{if}\, t\, \texttt{then}\, t\, \texttt{else}\, t \mid x \mid f(t_1, \ldots, t_n) \\
T & ::= & \texttt{Nat} \mid \texttt{Bool} \\
d & ::= & \texttt{dec}\, x = t : T \mid \texttt{def}\, f(x_1 : T_1, \ldots, x_n : T_n) : T = t \\
P & ::= & \texttt{prog}\, d^*\, \texttt{gorp}
\end{array}
$$

   (the phrase $\texttt{dec}\ x = t : T$ is the only adjustment). The idea is that before the start of execution (e.g. calling a `main` function), the global declarations are executed to initialise the global values.

   i. An operational semantics for FOFL with global declarations can be given by using a big-step evaluation relation extended with an *environment* ρ. The environment is a mapping from variables to values. Give the rules for the judgement $t \Downarrow_\rho v$, assuming that the values for global variables are defined in the environment. [*8 marks*]

   ii. Give operational rules for defining the initial environment from a program, the judgement $P \Downarrow \rho$. (**Hint:** to do this you may *restrict* the language so that the initial values of global variables may be defined in sequence and are not referenced before initialisation). [*3 marks*]

   iii. Now give the definition of a suitable corresponding type system, defining judgements for checking terms and for checking programs. (**Hint:** you will need to introduce a static counterpart of the environment and explain how it is defined before type-checking function bodies). [*5 marks*]

   iv. Give a a suitable safety theorem for this semantics. [*3 marks*]