COMPUTER SCIENCE TRIPOS Part II – 2021 – Paper 9

15 Types (nk480)

(*a*) In a simply-typed lambda calculus augmented with first-class continuations, booleans, a list type and its iterator (i.e., fold, but not full recursion), write a function

$$\texttt{every} : (X \to \mathsf{Bool}) \to \mathsf{List}\, X \to \mathsf{Bool}$$

such that $\texttt{every}\, p\, xs$ returns true if every element of xs satisfies p, and false otherwise. This function should also stop iterating over the list as soon as it finds a false element. You may use SML- or OCaml-style notation if desired, but explain any notation used beyond the basic lambda calculus.

[4 marks]

(*b*) In the monadic lambda calculus with state, suppose we change the typing rule for reading locations to not cause a monadic effect: If we suggest changing the monadic lambda calculus to permit treating reads as pure:

$$\frac{l : X \in \Sigma}{\Sigma; \Gamma \vdash\, !l : X}$$

(*i*) Is this rule still typesafe? Informally but carefully justify your answer. [2 marks]

(*ii*) Is the following *common subexpression elimination* transformation sound? Either give an argument why it is, or supply a counterexample and explain why it shows it is not. [6 marks]

```
let x = return e1;                 let x = return e1;
let y = e2;             =====>     let y = e2;
let z = return e1;                 [z/x]e3
e3
```

(*c*) In System F augmented with existential types, give an existential type for the interface of the natural numbers, and give an implementation for it. [8 marks]