COMPUTER SCIENCE TRIPOS Part IB

Thursday 4 June 2009 1.30 to 4.30

COMPUTER SCIENCE Paper 6

*Answer **five** questions.*

*Submit the answers in five **separate** bundles, each with its own cover sheet. On each cover sheet, write the numbers of **all** attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

STATIONERY REQUIREMENTS
Script paper
Blue cover sheets
Tags

SPECIAL REQUIREMENTS
Approved calculator permitted

1 Complexity Theory

Consider the following decision problems.

1. (PROB1) Given a graph $G = (V, E)$, does it contain a path that visits every **edge** exactly once?

2. (PROB2) Given a graph $G = (V, E)$, does it contain a path that visits every **node** exactly once?

(a) Which of the two problems is in P and which is NP-complete? [2 marks]

(b) Describe a polynomial time algorithm for the problem in P. [6 marks]

(c) Prove that the other problem is in fact NP-complete. [12 marks]

2 Complexity Theory

(a) Define precisely what we mean when we write $L_1 \leq_P L_2$. [4 marks]

(b) What is the difference between NP-completeness and NP-hardness? [2 marks]

(c) Let 3COL denote the following decision problem.

Given a graph $G = (V, E)$, is it 3-colourable?

(i) Is 3COL in NP? Why? [2 marks]

(ii) Show that 3SAT $\leq_P$ 3COL. [10 marks]

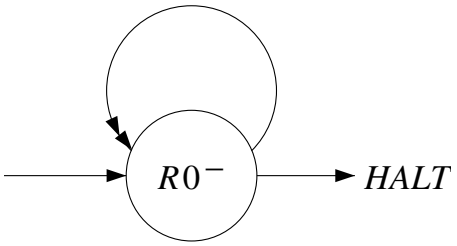
(iii) Argue that 3COL is NP-complete. [2 marks]

3 Computation Theory

(a) What is meant by a *state* (or *configuration*) of a register machine? [2 marks]

(b) A register machine program *Prog* is said to *loop at* $x \in \mathbb{N}$ if, when started with register $R1$ containing x and all other registers set to zero, the sequence of states *Prog* computes contains the same non-halted state at two different times.

(i) At which x does the following program loop?



[2 marks]

(ii) Show that if *Prog* loops at x, then the computation of *Prog* does not halt when started with register $R1$ containing x and all other registers set to zero. Is the converse true? [4 marks]

(iii) Consider the set $S = \{\langle e, x\rangle \mid Prog_e \text{ loops at } x\}$ of codes of pairs of numbers (e, x) such that the register machine program $Prog_e$ with index e loops at x. By adapting the usual proof of undecidability of the halting problem, or otherwise, show that S is an undecidable set of numbers. [Hint: if M were a register machine that decided membership of S, first consider replacing each *HALT* instruction (and each jump to a label with no instruction) with the program in part (i).] [12 marks]

4 Computation Theory

(a) Define what it means for a subset $S \subseteq \mathbb{N}$ to be a *recursively enumerable* set of numbers. [2 marks]

(b) Show that if S and S' are recursively enumerable sets of numbers, then so are the following sets (where $\langle x, y\rangle = 2^x(2y+1) - 1$).

(i) $S_1 = \{x \mid x \in S \text{ or } x \in S'\}$

(ii) $S_2 = \{\langle x, x'\rangle \mid x \in S \text{ and } x' \in S'\}$

(iii) $S_3 = \{x \mid \langle x, x'\rangle \in S \text{ for some } x' \in \mathbb{N}\}$

(iv) $S_4 = \{x \mid x \in S \text{ and } x \in S'\}$

Any standard results about partial recursive functions you use should be clearly stated, but need not be proved. [16 marks]

(c) Give an example of a subset $S \subseteq \mathbb{N}$ that is not recursively enumerable. [2 marks]

5 Foundations of Functional Programming

(*a*) Define the Church numerals giving the encodings of zero $\underline{0}$, one $\underline{1}$ and an arbitrary number $\underline{n}$. [3 marks]

(*b*) Define λ-terms to perform the following operations on Church numerals. You may assume standard definitions for Booleans (true, false, if, and, and or) and pairs (pair, fst, and snd). For each part, you may assume solutions to the previous parts of the question. You may *not* use a fixed-point combinator.

(*i*) Test for zero. [2 marks]

(*ii*) Successor. [2 marks]

(*iii*) Predecessor (where predecessor of zero is zero). [4 marks]

(*iv*) Less than or equal. [3 marks]

(*v*) Equality. [2 marks]

(*vi*) Successor modulus n (where $\mathsf{succn}\,\underline{n}\,\underline{m} = \underline{0}$ if $n = m + 1$, and $\mathsf{succn}\,\underline{n}\,\underline{m} = \underline{m+1}$ otherwise). [2 marks]

(*vii*) Modulus (e.g $\mathsf{mod}\,\underline{n}\,\underline{m} = \underline{m \bmod n}$). [2 marks]

6 Foundations of Functional Programming

(*a*) Define what it means for a λ-calculus term to be in normal form. Is it possible for a λ-term to have two normal forms that are not α-equivalent? Provide justification for your answer. [3 marks]

(*b*) For each of the following, give an example of a λ-term that

(*i*) is in normal form;

(*ii*) is not in normal form but has a normal form; and

(*iii*) does not have a normal form.

For (*ii*), you should also present the term's normal form, and for (*iii*) you should show that the term does not have a normal form. [4 marks]

We define a λ-term N to be *non-trivial* iff there exist A and B such that $N\,A \to^* \mathsf{true}$ and $N\,B \to^* \mathsf{false}$, where true and false encode the Booleans.

(*c*) Give an example of a λ-term that is non-trivial, and show that it is non-trivial. [2 marks]

We define a λ-term N as *total* iff for each λ-term M, either $N\,M \to^* \mathsf{true}$ or $N\,M \to^* \mathsf{false}$

(*d*) Give an example of a λ-term that is total, and show that it is total. [2 marks]

(*e*) Prove that there is no non-trivial and total λ-term.

[Hint: Suppose N is non-trivial and total where $N\,A \to^* \mathsf{true}$ and $N\,B \to^* \mathsf{false}$, and consider the term $N(Y\,L)$ where $L \equiv (\lambda x.\ \mathsf{if}\ (N\,x)\,B\,A)$ and where Y is the fixed-point operator.]

[7 marks]

(*f*) What consequences does this have for defining a general equality λ-term such that

$$\begin{array}{llll} \mathsf{equal}\,A\,B & \to^* & \mathsf{true} & \text{if } A = B \\ \mathsf{equal}\,A\,B & \to^* & \mathsf{false} & \text{otherwise} \end{array}$$

[2 marks]

7 Logic and Proof

(*a*) What is an S4 modal frame? [2 marks]

(*b*) For each of the following formulae of S4 modal logic, present either a sequent calculus proof or a falsifying interpretation.

(*i*) $(\Diamond(P \to Q) \wedge \Box P) \to \Diamond Q$ [6 marks]

(*ii*) $\Box(P \vee Q) \to (\Box P \vee \Diamond Q)$ [6 marks]

(*iii*) $(\Diamond\Box P \wedge \Diamond\Box Q) \to \Diamond(P \wedge Q)$ [6 marks]

8 Logic and Proof

(*a*) Briefly indicate the differences between the tableau calculus and the sequent calculus. [2 marks]

(*b*) Prove the formula $[\forall x\,(P(x) \to R(x,x))] \to [\forall x\,\exists y\,(R(x,y) \vee \neg P(y))]$ using the tableau calculus, or exhibit a falsifying interpretation. [8 marks]

(*c*) Give a model for the following set of clauses, or prove that none exists. Here a is a constant, while x and y are variables. Explain your reasoning clearly.

$$\{\neg R(x,a), R(x,x)\} \qquad \{\neg R(x,x), R(x,a)\}$$

$$\{\neg R(y,f(x)), \neg R(y,x)\} \qquad \{R(y,x), R(y,f(x))\}$$

[10 marks]

9 Semantics of Programming Languages

Consider the following syntax for a pure untyped functional language.

Booleans $b \in \mathbb{B} = \{\textsf{true}, \textsf{false}\}$
Integers $n \in \mathbb{Z} = \{..., -1, 0, 1, ...\}$
Variables $x \in \mathbb{X}$ for a set $\mathbb{X} = \{\mathrm{x}, \mathrm{y}, \mathrm{z}, ...\}$
Operations $op ::= + \mid \geq$
Expressions

$$e ::= \textsf{skip} \mid n \mid b \mid e_1 \;\; op \;\; e_2 \mid \textsf{if} \;\; e_1 \;\; \textsf{then} \;\; e_2 \;\; \textsf{else} \;\; e_3 \mid \textsf{fn} \;\; x \Rightarrow e \mid e_1 \; e_2 \mid x \mid \textsf{fix} \; e$$

The language supports recursion with a fixed-point operator **fix** e, which has semantics defined by the rule below.

$$\frac{}{\textsf{fix} \; e \longrightarrow e(\textsf{fix} \; e)}$$

(a) Give the semantic rules for function application for call-by-value, call-by-name, and full-beta reduction for this language (do not give the rules for binary operators, conditional, or fix). You should define a small-step reduction relation $e \longrightarrow e'$, stating precisely what notion of values v you are using. [10 marks]

(b) For the call-by-value semantics, characterise the expressions e from the grammar above that have an *immediate* runtime error in their outermost (top-level) construct. [3 marks]

(c) For each pair of semantics (call-by-value and call-by-name, call-by-name and full-beta, and full-beta and call-by-value), give an expression with different possible termination behaviours in each element of the pair. [4 marks]

(d) For each of your three semantics, explain a disadvantage in using that semantics for a programming language. [3 marks]

10 Semantics of Programming Languages

Consider the variant of untyped L1 with syntax as below and a standard small-step semantics $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ (this is identical to L1 except that it has equality testing $e_1 = e_2$ on integers instead of $\geq$ and that here stores are total functions).

Booleans $b \in \mathbb{B} = \{\mathsf{true}, \mathsf{false}\}$
Integers $n \in \mathbb{Z} = \{..., -1, 0, 1, ...\}$
Locations $\ell \in \mathbb{L} = \{l, l_0, l_1, l_2, ...\}$
Stores s, total functions from $\mathbb{L}$ to $\mathbb{Z}$
Values $v ::= \mathsf{skip} \mid n \mid b$
Operations $op ::= \quad = \mid +$
Expressions

$$e ::= \mathsf{skip} \mid n \mid b \mid e_1 \quad op \quad e_2 \mid \mathsf{if} \quad e_1 \quad \mathsf{then} \quad e_2 \quad \mathsf{else} \quad e_3 \mid \ell := e \mid !\ell \mid e_1; e_2 \mid \mathsf{while} \quad e_1 \quad \mathsf{do} \quad e_2$$

Define $[\![e]\!]$ to be the function that takes any store s and either is $\perp$ (undefined), if $\langle e, s\rangle \longrightarrow^{\omega}$, or is $\langle v, s'\rangle$, if $\langle e, s\rangle \longrightarrow^{*} \langle v, s'\rangle$.

Define (untyped) semantic equivalence $e_1 \simeq e_2$ iff $[\![e_1]\!] = [\![e_2]\!]$.

(*a*) State what it means for $\simeq$ to be a congruence. [2 marks]

(*b*) For each of the constructs of the expression grammar, define an explicit characterisation of $[\![e]\!]$ in terms only of the semantics $[\![e']\!]$ of its subexpressions e', without using the reduction relation. (For example, for n (which has no subexpressions) $[\![n]\!] = \lambda s.\langle n, s\rangle$.) [12 marks]

(*c*) Consider $(\mathsf{if}\ !l = 1\ \mathsf{then}\ e\ \mathsf{else}\ e) \simeq e$. Either prove it, using your answer to part (*b*), or exhibit a counterexample. [3 marks]

(*d*) Consider $(\mathsf{while}\ e_1\ \mathsf{do}\ e_2) \simeq (\mathsf{while}\ e_1\ \mathsf{do}\ (e_2; e_2))$ where e_1 does not read any store locations. State whether this is true or false, with an informal explanation of the possible cases. [3 marks]

END OF PAPER