COMPUTER SCIENCE TRIPOS Part IB

Thursday 3 June 2010 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

(*a*) Give precise definitions of *polynomial-time reductions* and ***NP**-completeness*. [2 marks each]

(*b*) Prove that for any language L, L is polynomial-time reducible to some problem in **NP** if, and only if, L is in **NP**. [6 marks]

(*c*) In a simple graph $G = (V, E)$, a set of vertices $X \subseteq V$ is said to be a *vertex cover* of G if every edge $e \in E$ has one endpoint in X. A set $X \subseteq V$ is an *independent set* of G if there is no edge between any two vertices in X.

Vertex Cover is defined as the decision problem where, given a graph $G = (V, E)$ and a positive integer k, we are to determine whether G contains a vertex cover with k or *fewer* vertices.

Independent Set is defined as the decision problem where, given a graph $G = (V, E)$ and a positive integer k, we are to determine whether G contains an independent set with k or *more* vertices.

(*i*) Show that a set X is a vertex cover of G if, and only if, its complement $V \setminus X$ is an independent set of G. [2 marks]

(*ii*) Use this to show that Vertex Cover is polynomial-time reducible to Independent Set and *vice versa*. [6 marks]

(*iii*) What can you conclude about the complexity of Vertex Cover? [2 marks]

2 Complexity Theory

(*a*) Give precise definitions of the complexity classes **L** and **NL**. [3 marks each]

(*b*) Explain why $\mathbf{NL} \subseteq \mathbf{P}$. [6 marks]

(*c*) The problem DIRECTED REACHABILITY is known to be **NL**-complete under *logarithmic space reductions* and the problem CVP is known to be **P**-complete under logarithmic space reductions.

Given just this information what can you conclude about the truth of the following statements?

(*i*) DIRECTED REACHABILITY is logarithmic-space reducible to CVP.

(*ii*) CVP is logarithmic-space reducible to DIRECTED REACHABILITY.

(*iii*) DIRECTED REACHABILITY is polynomial-time reducible to CVP.

(*iv*) CVP is polynomial-time reducible to DIRECTED REACHABILITY.

[2 marks each]

3 Computation Theory

(*a*) Define the notion of a *register machine* and the computation it carries out. [5 marks]

(*b*) What does it mean for a partial function $f(x_1, \ldots, x_n)$ of n arguments to be *register machine computable*? [3 marks]

(*c*) Why do there exist partial functions that are not register machine computable? (Any standard results you use in your answer should be carefully stated.) [3 marks]

(*d*) Consider the following register machine program.

$$
\begin{aligned}
L_0 &: R_1^- \rightarrow L_1, L_6 \\
L_1 &: R_2^- \rightarrow L_2, L_4 \\
L_2 &: R_0^+ \rightarrow L_3 \\
L_3 &: R_3^+ \rightarrow L_1 \\
L_4 &: R_3^- \rightarrow L_5, L_0 \\
L_5 &: R_2^+ \rightarrow L_4 \\
L_6 &: \mathrm{HALT}
\end{aligned}
$$

Assuming the contents of registers R_0 and R_3 are initially zero, what function of the initial contents of registers R_1 and R_2 does this program compute in register R_0 upon halting? (You may find it helpful to consider the graphical representation of the program.) [4 marks]

(*e*) Let $f(x_1, x_2)$ be the partial function that is equal to $x_1 - x_2$ if $x_1 \geq x_2$ and is undefined otherwise. Give a register machine program that computes f. [5 marks]

4 Computation Theory

(*a*) Define Church's representation of numbers n as λ-terms $\underline{n}$. [3 marks]

(*b*) What does it mean for a partial function $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ to be λ-*definable*? What is the relationship between λ-definability and computability? [3 marks]

(*c*) Show that $\texttt{succ}(x_1) = x_1 + 1$ is λ-definable. [4 marks]

(*d*) Ackermann's function $ack \in \mathbb{N}^2 \to \mathbb{N}$ is a total function of two arguments satisfying

$$\begin{aligned} ack(0, x_2) &= x_2 + 1 \\ ack(x_1 + 1, 0) &= ack(x_1, 1) \\ ack(x_1 + 1, x_2 + 1) &= ack(x_1, ack(x_1 + 1, x_2)). \end{aligned}$$

By considering $\lambda x.\, x\, T\, S$ where $T = \lambda f\, y.\, y\, f\, (f\, \underline{1})$ and S is chosen suitably, prove that Ackermann's function is λ-definable. [10 marks]

5 Logic and Proof

(*a*) Write brief notes on the use of binary decision diagrams (BDD) to represent propositional formulae. Illustrate your answer by constructing the BDD corresponding to the formula $[p \to (q \wedge s)] \wedge [s \vee (r \to s)]$, ordering the variables alphabetically. [8 marks]

(*b*) Exhibit a model for the following set of clauses, or prove that they are inconsistent:

$$\{\neg p(x, y), r(x, y), q(x), \neg p(y, x)\}$$

$$\{\neg r(x, y), \neg q(y), r(y, x)\}$$

$$\{r(x, y), \neg q(x), \neg q(y)\}$$

$$\{p(a, b)\} \qquad \{p(b, a)\} \qquad \{\neg r(a, b)\}$$

Here a and b are constants, while x and y are variables. [12 marks]

6 Logic and Proof

(*a*) Use the sequent or tableau calculus to prove the formula

$$\exists x\,(P(x) \to Q) \to \forall x\,(P(x) \to Q)$$

[6 marks]

(*b*) A mysterious propositional connective, $\odot$, has the following sequent calculus rule, $(\odot r)$:

$$\frac{\Gamma \Rightarrow \Delta, A, B \qquad \Gamma, A, B \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, A \odot B}$$

What is the corresponding left-side sequent calculus rule, $(\odot l)$? Justify your answer, for example by giving the truth table for $\odot$. [6 marks]

(*c*) Use the DPLL method to find a model of the following set of clauses, or alternatively to prove that they are inconsistent.

$$\{P, R, \neg S\}\ \ \{\neg Q, R\}\ \ \{\neg P, \neg S, \neg R\}\ \ \{\neg P, S, Q\}\ \ \{S, Q, P\}\ \ \{\neg Q, \neg R\}\ \ \{\neg S, \neg R, P\}$$

[8 marks]

7 Mathematical Methods for Computer Science

(*a*) What is an orthonormal basis? Why is it important that a basis be orthonormal? [4 marks]

(*b*) A real, periodic function, $f(x)$, can be expressed as a Fourier series. This can be shown in several ways. One is as a sum of weighted, offset, cosine functions:

$$f(x) = \sum_{k=0}^{\infty} A_k \cos\left(xk\frac{2\pi}{T} - \theta_k\right)$$

A second way is as a sum of complex exponentials with complex coefficients:

$$f(x) = \sum_{k=-\infty}^{\infty} c_k \exp\left(ixk\frac{2\pi}{T}\right)$$

where the complex coefficients, c_k, have the constraint $c_k = c^*_{-k}$ for $f(x)$ real.

(*i*) Prove that these two alternative expressions of the Fourier series are equivalent. [8 marks]

(*ii*) Express the complex coefficient c_k in terms of the real parameters A_k and θ_k. [2 marks]

(*c*) Consider the box function:

$$b(x) = \begin{cases} 1, & |x| \leq \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}$$

and the tent function:

$$t(x) = b(x) * b(x) = \begin{cases} x+1, & -1 \leq x < 0 \\ 1-x, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

(*i*) Find the Fourier transform of $b(x)$. [4 marks]

(*ii*) Find the Fourier transform of $t(x)$. [2 marks]

The following formulæ may be useful.

$$\begin{aligned} e^{i\phi} &= \cos\phi + i\sin\phi \\ \sin(a+b) &= \sin a\cos b + \cos a\sin b \\ \sin(a-b) &= \sin a\cos b - \cos a\sin b \\ \cos(a+b) &= \cos a\cos b - \sin a\sin b \\ \cos(a-b) &= \cos a\cos b + \sin a\sin b \\ \cos^2\phi + \sin^2\phi &= 1 \end{aligned}$$

8 Mathematical Methods for Computer Science

(*a*) Let $\{X_n : n = 0, 1, \cdots\}$ be a two-state Markov chain with transition probabilities given by the matrix

$$P = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}$$

Let $N_{i,j} = \mathbb{E}\,(\text{number of visits to state } j \text{ before first return to state } i | X_0 = i)$ for $i \neq j$. Prove that

$$N_{2,1} = \frac{1-q}{1-p}$$

giving careful attention to any special cases.

[Hint: Consider $\phi_{i,j}^{(n)} = \mathbb{P}(X_1 = j, X_2 = j, \ldots, X_n = j, X_{n+1} = i | X_0 = i)$.]

[8 marks]

(*b*) Two marksmen, Alice and Bob, take turns shooting at a target. They agree that Alice will shoot after each hit, while Bob will shoot after each miss. Suppose Alice hits the target with probability α, while Bob hits the target with probability β. Over a long period of time, what proportion of shots hit the target? State carefully any theorems that you use in arriving at your answer. Again, check any special cases. [12 marks]

9 Semantics of Programming Languages

A very simple imperative language, L0, has the following syntax and semantics.

Locations:	$l,\ l_1,\ l_2,\ \ldots$ (infinite)
Syntax:	$e\ ::=\ \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid l := e \mid\ !l$
Store:	finite partial functions s from locations to $\{\mathsf{true}, \mathsf{false}\}$
Configuration:	pairs $\langle e, s\rangle$ of an expression e and a store s
Type:	bool (this is the only type)
Environment:	a finite set Γ of locations

(r-if1) $\langle \mathsf{if\ true\ then}\ e_1\ \mathsf{else}\ e_2, s\rangle \longrightarrow \langle e_1, s\rangle$

(r-if2) $\langle \mathsf{if\ false\ then}\ e_1\ \mathsf{else}\ e_2, s\rangle \longrightarrow \langle e_2, s\rangle$

(r-if3) $$\frac{\langle e, s\rangle \longrightarrow \langle e', s'\rangle}{\langle \mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2, s\rangle \longrightarrow \langle \mathsf{if}\ e'\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2, s'\rangle}$$

(r-deref) $\langle !l, s\rangle \longrightarrow \langle b, s\rangle$ if $l \in \mathrm{dom}(s)$ and $s(l) = b$

(r-assign1) $\langle l := b, s\rangle \longrightarrow \langle b, s\{l \mapsto b\}\rangle$ if $l \in \mathrm{dom}(s)$ and $b = \mathsf{true}$ or $b = \mathsf{false}$

(r-assign2) $$\frac{\langle e, s\rangle \longrightarrow \langle e', s'\rangle}{\langle l := e, s\rangle \longrightarrow \langle l := e', s'\rangle}$$

(t-bool1) $\Gamma \vdash \mathsf{true} : \mathsf{bool}$ (t-bool2) $\Gamma \vdash \mathsf{false} : \mathsf{bool}$

(t-deref) $\Gamma \vdash !l : \mathsf{bool}$ if $l \in \Gamma$ (t-assign) $\dfrac{\Gamma \vdash e : \mathsf{bool}}{\Gamma \vdash l := e : \mathsf{bool}}$ if $l \in \Gamma$

(t-if) $$\frac{\Gamma \vdash e : \mathsf{bool} \quad \Gamma \vdash e_1 : \mathsf{bool} \quad \Gamma \vdash e_2 : \mathsf{bool}}{\Gamma \vdash \mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 : \mathsf{bool}}$$

(*a*) State the Progress theorem for well-typed L0. [2 marks]

(*b*) Prove the Progress theorem, by rule induction on the structure of type derivations. [9 marks]

(*c*) Define a notion of semantic equivalence for L0. Give a constraint on the syntax of e under which ($\mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_1$) is semantically equivalent to (e_1). [4 marks]

(*d*) We now write $(e; e')$ as a shorthand for ($\mathsf{if}\ e\ \mathsf{then}\ e'\ \mathsf{else}\ e'$). We say that two L0 expressions, e_1 and e_2, form a "snap-back pair" if for every L0 expression e, the expression $((e_1; e); e_2)$ is semantically equivalent to (true). Either exhibit a snap-back pair, or argue informally why there are no snap-back pairs in L0. [5 marks]

10 Semantics of Programming Languages

Below is the syntax and operational semantics for a pure functional language.

Types: $T ::= \mathsf{bool} \mid T \to T$
Variables: $\{x, y, z, \ldots\}$
Expressions: $e ::= \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid \mathsf{fn}(x : T) \Rightarrow e \mid e\, e'$.

In the expression $\mathsf{fn}(x : T) \Rightarrow e$, the variable x is binding in e.

$$\text{(if1)} \quad (\mathsf{if\ true\ then}\ e_1\ \mathsf{else}\ e_2) \longrightarrow e_1$$

$$\text{(if2)} \quad (\mathsf{if\ false\ then}\ e_1\ \mathsf{else}\ e_2) \longrightarrow e_2$$

$$\text{(if3)} \quad \frac{e \longrightarrow e'}{(\mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2) \longrightarrow (\mathsf{if}\ e'\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2)}$$

$$\text{(app)} \quad \frac{e_1 \longrightarrow e_1'}{e_1\, e_2 \longrightarrow e_1'\, e_2}$$

$$\text{(fn)} \quad (\mathsf{fn}(x : T) \Rightarrow e)\, e' \longrightarrow \{e'/x\}e$$

(There is no need for a store because there are no store access operations.)

(*a*) Is this a call-by-value or a call-by-name language? Revise the operational semantics to demonstrate the other calling convention. [4 marks]

(*b*) A type environment is a finite partial function Γ from variables to types. Define a typing relation $\Gamma \vdash e : T$ by giving a set of rules. [6 marks]

(*c*) Are the following expressions typable?

$$e_1 = \mathsf{fn}(f : (\mathsf{bool} \to \mathsf{bool}) \to \mathsf{bool}) \Rightarrow \big(\mathsf{fn}(f : \mathsf{bool} \to \mathsf{bool}) \Rightarrow f\ f\big)$$

$$e_2 = \mathsf{fn}(f : \mathsf{bool} \to (\mathsf{bool} \to \mathsf{bool})) \Rightarrow \big(\mathsf{fn}(x : \mathsf{bool}) \Rightarrow (f\ x)\ x\big)$$

[2 marks]

(*d*) State formally the following two theorems of the one-step reduction semantics at the top of the page and the type system that you defined in part (*b*): Progress and Type Preservation. Take care to explain what a value is. (No proofs are required for this part.) [3 marks]

(*e*) State and prove the Type Safety theorem. You may use the results stated in part (*d*). [5 marks]

END OF PAPER