COMPUTER SCIENCE TRIPOS Part IB

Thursday 8 June 2017 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

Let $S \subseteq \mathbb{N}$ be a set of numbers. We write bin$S$ for the set of binary strings (i.e. strings in $\{0,1\}^*$) x such that x is a binary representation of a number in S. We also write un$S$ for the set $\{a^k \mid k \in S\}$ where the notation a^k means the string consisting of k repetitions of the letter a.

(a) Suppose bin$S \in \mathsf{TIME}(2^{cn})$ for some constant c. Prove that un$S \in \mathsf{P}$. [4 marks]

(b) Give definitions of the complexity classes L and NL. [4 marks]

(c) Prove that:

(i) bin$S \in \mathsf{SPACE}(n)$ if, and only if, un$S \in \mathsf{L}$; and

(ii) bin$S \in \mathsf{NSPACE}(n)$ if, and only if, un$S \in \mathsf{NL}$.

[6 marks]

(d) Recall that Reach is the problem of reachability in directed graphs. Using part (c) or otherwise, show that, if Reach were in L, it would follow that $\mathsf{SPACE}(n) = \mathsf{NSPACE}(n)$. State carefully any standard results that you use. [6 marks]

2 Complexity Theory

Define DOUBLE-SAT to be the set of Boolean formulas ϕ in CNF such that ϕ is satisfied by *at least two* distinct assignments of truth values to its variables.

Your task is to prove that DOUBLE-SAT is NP-complete. In order to do this, you need to do the following.

(a) Give a definition of *polynomial-time reductions*. [2 marks]

(b) Give a definition of the complexity class NP. [2 marks]

(c) Using parts (a) and (b), give a definition of NP-hardness. [2 marks]

(d) Using the above, give a definition of NP-completeness. [2 marks]

(e) Give a proof that polynomial-time reductions are closed under composition. [3 marks]

(f) Prove that DOUBLE-SAT satisfies all parts of the definition in (d). In your proof, you should make clear where each of the above parts is used. In addition, you may use any standard results about the complexity of Boolean satisfiability, as long as they are clearly stated. [9 marks]

3 Computation Theory

(a) Give register machine programs for the following operations:

(i) Non-destructively copying the contents of register `X` to register `Y`. [2 marks]

(ii) Assigning to registers `X` and `Y` the values 0 and $2^x(2y+1)$, where x and y are the starting values in `X` and `Y`. [3 marks]

(iii) Assigning to registers `X` and `Y` the values x and y, where 0 and $2^x(2y+1)$ are the starting values in `X` and `Y`. [5 marks]

(b) What does it mean that a register machine *decides the Halting Problem*? Prove that no such register machine exists. [10 marks]

4 Computation Theory

(*a*) Explain what it means for a partial function h to be defined by *primitive recursion* from partial functions f and g. Why is h a totally defined function if f and g are? [5 marks]

(*b*) (*i*) Define the class of *primitive recursive* functions. [5 marks]

(*ii*) For each $n \in \mathbb{N}$, show that the constant function $\mathbb{N}\to\mathbb{N}$ with value n is primitive recursive. [2 marks]

(*iii*) Explain why it is the case that not every function $\mathbb{N}\to\mathbb{N}$ is primitive recursive, carefully stating any general results you use. [3 marks]

(*c*) Given $e \in \mathbb{N}^2\to\mathbb{N}$ and $n \in \mathbb{N}$, let $e_n \in \mathbb{N}\to\mathbb{N}$ be the function given by $e_n(x) = e(n, x)$. Suppose that e is primitive recursive.

(*i*) Show that each e_n is primitive recursive. [1 mark]

(*ii*) Using a suitable diagonalisation argument, or otherwise, prove that it cannot be the case that for all primitive recursive functions $f \in \mathbb{N}\to\mathbb{N}$ there exists $n \in \mathbb{N}$ with e_n equal to f. [4 marks]

5 Logic and Proof

(*a*) Exhibit an interpretation in S4 modal logic that simultaneously satisfies the formulas $P \wedge Q$, $\Box(P \vee Q)$, $\Diamond\neg P$, $\Diamond\neg Q$ at a particular world, w. [5 marks]

(*b*) For each of the following sets of clauses, either exhibit a model or show that none exists. Below, a and b are constants, while x, y and z are variables. Briefly justify your answers.

(*i*)

$$\{\neg R(x,y), R(f(x),f(y))\}$$
$$\{R(a,b)\} \quad \{\neg R(x,x)\}$$
$$\{\neg R(y,x), R(y,z), \neg R(x,z)\}$$

[7 marks]

(*ii*)

$$\{\neg Q(x,y), \neg Q(y,x), R(x)\}$$
$$\{\neg P(a,y), Q(y,y)\}$$
$$\{\neg Q(x,y), P(b,x)\}$$
$$\{P(z,b), P(x,y)\}$$
$$\{\neg R(b), \neg R(y)\}$$

[8 marks]

6 Logic and Proof

(*a*) Draw ordered binary decision diagrams (BDDs) for each of the following formulas, thereby identifying which of them are logically equivalent.

$$P \to (Q \to R)$$
$$P \to (R \to Q)$$
$$(\neg Q \vee R) \vee \neg P$$

[8 marks]

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

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

Present the corresponding left-side sequent calculus rule, $(\odot l)$, along with the truth table for $\odot$. [6 marks]

(*c*) For the following formula, either exhibit a formal proof (using the sequent calculus, augmented with the $(\odot r)$ rule above) or exhibit a falsifying interpretation:

$$\Rightarrow \exists x(P(x) \odot Q(x)),\ (\forall x\, P(x)) \wedge (\forall x\, Q(x))$$

[6 marks]

7 Mathematical Methods for Computer Science

(*a*) For an inner product space $V = \mathbb{R}^4$ with Euclidean norm and a set of vectors $\{w, x, y, z\} \in V$

$$w = \left(-\tfrac{1}{2},\ \tfrac{1}{2},\ -\tfrac{1}{2},\ \tfrac{1}{2}\right) \qquad x = \left(-\tfrac{1}{2},\ \tfrac{1}{2},\ \tfrac{1}{2},\ -\tfrac{1}{2}\right)$$

$$y = \left(\tfrac{1}{2},\ \tfrac{1}{2},\ -\tfrac{1}{2},\ -\tfrac{1}{2}\right) \qquad z = \left(\tfrac{1}{2},\ \tfrac{1}{2},\ \tfrac{1}{2},\ \tfrac{1}{2}\right)$$

demonstrate both whether the vectors $\{x, y, z, w\}$ form an *orthogonal system*, and whether they form an *orthonormal system*. [4 marks]

(*b*) Let $f[n]$ for $n = 0, 1, 2, \cdots, N-1$ be a discrete data sequence of N complex values, where $N = 2^Z$ is some integer power of 2.

(*i*) Define the Discrete Fourier Transform (DFT) of $f[n]$, a discrete sequence $F[k]$, as a series using complex exponentials. In terms of N, how many complex multiplications would be required to compute $F[k]$ explicitly? Then define W, the primitive N^{th} root of unity, and now re-express your series for $F[k]$ in terms of W. [4 marks]

(*ii*) Now express your DFT sequence $F[k]$ using a series with only $N/2$ terms, by capturing the second half of the series within the first half, and show that fewer complex multiplications are required. [4 marks]

(*iii*) Now show that separating $F[k]$ into two new sequences of half-length, each of which computes only $N/2$ coefficients $F[k]$, leads to a very efficient recursive form. How many times can this halving process be repeated? Ultimately how many complex multiplications are therefore required for this Fast Fourier Transform (FFT)? For 'big data' applications requiring a DFT on a billion data values, what speed-up factor can be expected by using an FFT instead of explicitly computing a DFT? [4 marks]

(*c*) Piecewise continuous and absolutely integrable functions $f(x), g(x) : \mathbb{R} \to \mathbb{C}$ have Fourier transforms $F(\omega)$ and $G(\omega)$, respectively. Let $h(x) = (f * g)(x)$ be the convolution of $f(x)$ and $g(x)$:

$$h(x) = \int_{-\infty}^{\infty} f(x-y)g(y)dy$$

Prove that $H(\omega)$, the Fourier transform of $h(x)$, is simply the product:

$$H(\omega) = 2\pi F(\omega) \cdot G(\omega)$$

[4 marks]

8 Mathematical Methods for Computer Science

(*a*) (*i*) State the central limit theorem. [2 marks]

(*ii*) Consider a binomially distributed random variable T with parameters Bin(n,p) where n is a positive integer and $0 < p < 1$. Using the central limit theorem derive an approximation to the probability $\mathbb{P}(T > d)$ where $d \in (0, n)$ and where n is sufficiently large. [4 marks]

(*b*) Let $(X_n)_{n\geq 1}$ be a Markov chain on the states $\{0, 1, 2\}$ with transition matrix

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1-\alpha & \alpha \\ 1-\alpha & \alpha & 0 \end{pmatrix}$$

where $0 < \alpha < 1$.

(*i*) Draw the state space diagram for the Markov chain X_n. [2 marks]

(*ii*) Explain why X_n is an irreducible, recurrent and aperiodic Markov chain. [6 marks]

(*iii*) Define an equilibrium distribution $\pi = (\pi_0, \pi_1, \pi_2)$ for the Markov chain X_n and determine π. [6 marks]

9 Semantics of Programming Languages

Consider a language with abstract syntax

$$e ::= \quad n \mid x \mid \textbf{let}\, x = e_1 \,\textbf{in}\, e_2 \mid \textbf{alloc} \mid \textbf{free}\, e \mid e_1 := e_2 \mid !e \mid e_1; e_2 \mid e_1 + e_2$$

This is intended to allow computation over data allocated in a concrete block of memory: n ranges over a set $W = \{0, \ldots, 2^{32} - 1\}$ of machine words, used both as values and as addresses. A memory state is described by a total function $m : W \to W$, giving the value at each address, and a set $a \subseteq W$, identifying the locations that are currently allocated. The term x ranges over a set of non-mutable variables, not allocated in memory. The expression $e := e'$, $!e$, **alloc**, and **free** e are respectively assignment, dereferencing, allocation, and free of single words.

(*a*) Define a reasonable deterministic operational semantics for this language, as a transition relation

$$\langle e, m, a\rangle \longrightarrow \langle e', m', a'\rangle$$

and a predicate

$$\langle e, m, a\rangle\, \textbf{error}$$

that identifies the configurations that are runtime errors. You can omit the rules for $e_1; e_2$ and $e_1 + e_2$ and the standard definition of substitution.

Your definition should ensure (though you need not prove) that for any configuration $\langle e, m, a\rangle$, either e is a value n, or there is exactly one transition $\langle e, m, a\rangle \longrightarrow \langle e', m', a'\rangle$ from that configuration, or there is exactly one derivation of a runtime error $\langle e, m, a\rangle\, \textbf{error}$.

Note and explain your choices. [17 marks]

(*b*) One could rule out some of those runtime errors with a simple type system that keeps addresses and the numbers used for arithmetic distinct, with types

$$T ::= \quad \textbf{address} \mid \textbf{number}$$

and type rules that constrain assignment, dereferencing, allocation, free, and arithmetic.

Discuss which of your runtime errors could be prevented by this. [3 marks]

10 Semantics of Programming Languages

Let x range over a set X of identifiers, n range over the natural numbers $\mathbb{N}$, and s range over stores: total functions from X to $\mathbb{N}$.

Consider a language with the following abstract syntax.

$e ::= \quad n \mid x := e \mid !x \mid e_1; e_2$

(a) Define a conventional deterministic small-step operational semantics $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ for the language. Comment briefly on the choices you make. [5 marks]

(b) If your language is deterministic and terminating, the operational semantics implicitly defines a more abstract semantics: we can regard each expression as a function over stores $[\![e]\!]$ that takes store s to the unique number n and store s' such that

$$\langle e, s\rangle \longrightarrow^* \langle n, s'\rangle \wedge \nexists e'', s''.\ \langle n, s'\rangle \longrightarrow \langle e'', s''\rangle$$

This language is quite limited in expressiveness. Describe, as clearly and precisely as you can, the set of functions from stores to (number, store) pairs that are expressible as $[\![e]\!]$ for some e. [5 marks]

(c) The primitive contexts C for this language are expressions with a single hole:

$C ::= \quad x := \_ \mid e_1; \_ \mid \_; e_2$

Write $C[e]$ for the expression resulting from replacing the hole in C by e.

Say a binary relation $\sim$ over expressions is a congruence if $e \sim e'$ implies $\forall C.\ C[e] \sim C[e']$.

Say a binary relation $\sim$ over expressions respects final values if $e \sim e'$ implies $\forall s_0, n, n', s, s'.\ (\langle e, s_0\rangle \longrightarrow \langle n, s\rangle \wedge \langle e', s_0\rangle \longrightarrow \langle n', s'\rangle) \Rightarrow n = n'$.

Use your characterisation of part (b) to define an equivalence relation over expressions that is a congruence and respects final values. Explain briefly why it has those properties. [4 marks]

(d) Define a terminating algorithm that, for any expressions e and e', computes whether $e \sim e'$ or not. Explain informally why it is correct. *Hint:* you may want to adapt your semantics from part (a) to compute symbolically. [6 marks]

END OF PAPER