B22 MODELS OF COMPUTATION

2 May 1996 Examination

Time: 2 hours 30 minutes

Answer any 3 questions

1. (a) Give a formal definition of a finite state machine and the language that it recognises.

[10 marks]

(b) For each of these 3 languages, decide if it can be recognised by a finite state machine. If it can, draw the machine that recognises it.

(i) Strings over {a,b} with an even number (including 0) of "a"s.

(ii) Strings over {a,b} that end with a "b" but do not contain the substring "aa".

(iii) Strings over {a,b} of the form NbN, where N consists of n successive occurrences of "a".

[10 marks]

(c) Given a finite state machine M that recognises a language L over an alphabet A, give a construction for a finite state machine M', based on M, for each of the following 2 languages.

(i) A* - L, i.e. all the strings not recognised by M.

(ii) Strings s such that reverse(s) is in L, where reverse(s) reverses the string s.

<Hint: Pay particular attention to transitions that may not be defined in M.>

[13 marks]

2. (a) Define, formally, what is meant by a regular language.

[6 marks]

(b) State Kleene's Theorem.

[3 marks]

(c) Give an outline proof of either half of Kleene's Theorem.

[16 marks]

(d) For an arbitrary alphabet A, is the set of regular languages over A countable? Give an informal proof of your answer.

[8 marks]

3. A Turing machine is said to recognise a language L over an alphabet A if, when given any string s, it halts with output "1" if s is in L and it halts with output "0" otherwise.

(a) Draw a Turing machine that recognises the language over {a,b} consisting of strings of the form NbN, where N denotes a string containing n successive occurrences of "a".

[12 marks]

(b) Given two Turing machines M1 and M2 that recognise the languages L1 and L2 respectively, outline the construction of a Turing machine to recognise each of these two languages:

(i) L1 - L2, i.e. containing just those strings in L1 that are not in L2.

[8 marks]

(ii) L1L2, i.e. the concatenation of L1 and L2.

[13 marks]

Concentrate on the high-level operation of the machines that constitute your solutions. Full marks can be obtained by showing how such machines may be constructed from basic operations that are clearly computable by Turing machines.

<Hint: You may find it easier to use a multi-tape Turing machine.>

4. (a) Define what is meant by a "decision problem", and what it means to say that one problem P "reduces to" another problem P'.

[6 marks]

(b) State and give an outline proof of the Halting Theorem for Turing machines.

[18 marks]

(c) Prove that no Turing machine can solve either of the following problems:

(i) Given a Turing machine M, does M halt when given a blank tape?

(ii) Given a Turing machine M, does M halt for all finite input strings that it may be given?

[9 marks]

5. (a) Define the terms "alpha reduction", "beta reduction" and "normal-order reduction" as they apply to the lambda calculus.

[6 marks]

(b) Where L stands for the Greek lower-case lambda, reduce the following lambda expression to its simplest form using normal-order reduction:

(L f . (L x . f(f x)))(L x . x x)x

Are any other reduction orders possible with the expression? If so, do any of them give a different result?

[6 marks]

(c) Functional programming languages usually provide a list-building operation, called "cons", and two functions to return respectively the head ("hd") and tail ("tl") of such a constructed list.

Give definitions of these 3 functions that are suitable for modelling lists in the pure lambda calculus.

[10 marks]

(d) Give a persuasive informal argument to show that the model of computation embodied by the lambda calculus is no more powerful than that embodied by Turing machines and vice versa.

[11 marks]