University of Dundee

University of Dundee
School of Science and Engineering
Examinations 2021

BSc Degrees in Computing

AC32008 Theory of Computation

Time allowed: **THREE** hours

Instructions

There are **SIX** questions.

Candidates must answer **FOUR** questions. All questions carry equal marks. Where appropriate, the value of each part of a question is given in square brackets.

Approved calculators may be used in this examination.

Do not turn over this question paper until instructed to by the Senior Invigilator

1. Finite Automata and Regular Languages

(a) Consider the following NFA-ε:

δ	ε	a	b	c
$\rightarrow p$	$\emptyset$	$\{p\}$	$\{q\}$	$\{r\}$
q	$\{p\}$	$\{q\}$	$\{r\}$	$\emptyset$
$*r$	$\{q\}$	$\{r\}$	$\emptyset$	$\{p\}$

with the start state p and one accepting state r.

(i) Compute the ε closure of each state. **[3 marks]**

(ii) Give all the strings of length three or less accepted by the automaton. **[4 marks]**

(iii) Convert the automaton to a NFA without ε transitions. **[6 marks]**

(b) Describe informally (in English) the language represented by the regular expression

$$(1+\varepsilon)(00^*1)^*0^*.$$

[4 marks]

(c) Let L be the language over $\{0,1\}$ consisting of all strings which do not contain the substring 1110 (thus $\varepsilon, 110, 01111, 000 \in L$, while $1110, 01111101 \notin L$). Find a regular expression $\mathbf{r}$ which represents the language L, i.e., so that $L(\mathbf{r}) = L$, and justify your answer. [Hint: You may find it helpful to consider the part of the string starting with the first 1 and ending with the last 0. This part consists of alternating non-empty blocks of ones and of zeros.] **[8 marks]**

2. Regular Languages and Pumping Lemma

(a) The Pumping Lemma can be stated as follows:

Suppose that L is a regular language. Then there is a constant n (equal to the number of states in the smallest DFA which accepts L) with the following property: For any string $z \in L$ with $|z| \geq n$, there exist three strings u, v, w such that $z = uvw$, $|uv| \leq n$, $|v| \geq 1$ and $uv^i w \in L$ for any $i \geq 0$.

Explain the following:

(i) Why we only consider strings z with $|z| \geq n$; **[3 marks]**

(ii) Why we know that $|v| \geq 1$; **[2 marks]**

(iii) Why we do *not* know that $|u| \geq 1$. **[3 marks]**

(b) Let L be the language over the alphabet $\{0, 1, 2\}$ given by

$$L = \{0^i 1^j 2^i | i \text{ and } j \text{ are arbitrary integers}\}.$$

Show that L is not regular. **[12 marks]**

(c) Suppose that L is a language over an alphabet Σ, and let $\overline{L}$ be its complement $\{x \in \Sigma^* | x \notin L\}$. By considering a deterministic finite automaton (DFA) M accepting the language L, show that if L is regular, then $\overline{L}$ is regular also. **[5 marks]**

3. Turing Machines, Decidability and Languages Accepted by TMs

(a) Give an example of a standard Turing machine which halts on input strings of odd length and does not halt on input strings of even length. **[5 marks]**

(b) Let L be the language over the alphabet $\{0,1\}$ given by $L = \{0^i 1 0^j 1 0^{ij} | i, j \geq 0\}$ that is, the set of binary strings with three blocks of zeroes, the length of the third block being the product of the length of the first two blocks (e.g. 0010001000000). Sketch the construction of a standard Turing Machine M which accepts the language L. (It is not necessary to give full details of the machine M; an informal description of its operation is sufficient provided it is convincing.) **[10 marks]**

(c) Suppose that M_a, M_b, M_c, M_d are four Turing machines and that M_a and M_b halt on all inputs, while M_c and M_d do not halt on all inputs.

Also suppose that the three languages L_1, L_2, L_3 satisfy

$$L_1 = L(M_a) = L(M_d), \quad L_2 = L(M_c), \quad L_3 = L(M_b).$$

Consider each of the following statements, and say for which of the languages L_1, L_2, L_3 the statement is true:

(a) The language is partially decidable.

(b) The language is totally decidable.

(c) There is not enough information to say whether the language is totally decidable or not.

In each case give a very brief explanation for your answer. **[6 marks]**

(d) Let L be the language consisting of all odd prime numbers, written in binary, but backwards (thus for example 19, which is 10011 in binary, is written as 11001). Can the set L be generated by a Turing machine in the standard order for binary strings? Give brief reasons for your answer. **[4 marks]**

4. Universal Turing Machine and Diagonalisation

(a) The universal language L_u is defined by:

$$L_u = \{\langle M, w\rangle | M \text{ accepts } w\}.$$

In what sense is the language "universal"? **[2 marks]**

(b) Define L_h to be the language

$$\{w_i | M_i \text{ halts on input } w_i\}.$$

Determine whether or not the string 11101001010010111 is in L_h.

[6 marks]

(c) The language L_u is accepted by a universal Turing machine U, that is, $L_u = L(U)$.

(i) Explain how a universal Turing machine is, in a sense, equivalent to a modern computer such as a PC. **[4 marks]**

(ii) How is the machine U able to simulate the working of a Turing machine which is larger that U itself, i.e., a machine that has more states than U and a larger transition table? **[4 marks]**

(d) Let L be the language $L = \{0^i 1^i \,|\, i \geq 0\}$. Show that there is a language L' which is a subset of L, (i.e., $L' \subseteq L$) such that L' is not partially decidable. [Hint: Let $x_i = 0^i 1^i$ and recall the diagonalisation process used to construct L_d.] **[9 marks]**

5. Complexity of Turing Machines

(a) Consider the Deterministic Turing Machine (DTM) defined below.

Input alphabet $\Sigma = \{0,1\}$

Tape alphabet $\Gamma = \{0,1,B\}$

States: $Q = \{q_0, q_1, q_2, q_N, q_Y\}$

Final states: $F = \{q_Y\}$

Transition function δ:

δ	q_0	q_1	q_2	q_N	q_Y
0	(q_1,B,R)	$(q_1,0,R)$	$(q_2,0,L)$	–	–
1	(q_1,B,R)	$(q_1,1,R)$	$(q_2,1,L)$	–	–
B	(q_Y,B,R)	(q_2,B,L)	(q_0,B,R)	–	–

(Note: You should not need to look at the transition table; the behaviour of the machine is described below.)

If the input is the empty string, the tape head moves right and the machine enters state q_Y, accepting in one step. Otherwise the first symbol is deleted (i.e., changed to a blank) then the head moves right until it finds a blank (not changing any cell), then left until it finds a blank (again not changing any cell), then the head moves one cell the right and the machine returns to the initial state.

What is the time complexity function $T_M(n)$?

[Hint: The sum of the first k odd numbers is k^2, i.e., $1+3+5+\ldots+(2k-1)=k^2$.]

[7 marks]

(b) A number of polynomial transformations are known, including the following:

1. SATISFIABILITY $\propto$ 3-SATISFIABILITY
2. TSP $\propto$ SATISFIABILITY
3. HAMILTONIAN CIRCUIT $\propto$ TSP
4. SATISFIABILITY $\propto$ 3-COLOURING
5. CLIQUE $\propto$ HAMILTONIAN CIRCUIT
6. HAMILTONIAN CIRCUIT $\propto$ CLIQUE
7. SATISFIABILITY $\propto$ CLIQUE

Assuming that it is known (from Cook's Theorem) that SATISFIABILITY is **NP**-complete, which transformation(s) from the list above can be used to show that TSP is **NP**-complete? **[4 marks]**

(c) Two theorems in complexity theory are the following:

1. If $L_1 \in \mathbf{P}$ and $L_2 \propto L_1$, then $L_2 \in \mathbf{P}$.
2. If $L_1 \propto L_2$ and $L_2 \propto L_3$, then $L_1 \propto L_3$.

In each case explain informally what these mean and why intuitively they are true. **[7 marks]**

(d) The class **NP** in intended to capture those problems in which every yes-instance has an easily checkable solution.

For a non-deterministic Turing machine, the time complexity is defined using the following:

- Time $t_M(x)$ for an individual input x is the *minimum* number of steps over all accepting computations on x;
- Time $T_M(n)$ for inputs of length n is the *maximum* of $t_M(x)$ over all accepted inputs x of length n.

Explain why it is appropriate to use a minimum for t_M and a maximum for T_M. **[7 marks]**

6. Polynomial Transformations and NP-Completeness

(a) Define a polynomial transformation and say what it means for a language to be **NP**-complete. Prove that if there is an **NP**-complete language L which is solvable in polynomial time (i.e. if $L \in \mathbf{P}$), then $\mathbf{P} = \mathbf{NP}$ (first show that if $L_1 \in \mathbf{P}$ and $L_2 \propto L_1$, then $L_2 \in \mathbf{P}$, then use this to show that if L is NP-complete and solvable in polynomial time, then $\mathbf{P} = \mathbf{NP}$. Explain the significance of this result. (You may use any standard result provided it is clearly stated.) **[13 marks]**

(b) The problem k-SATISFIABILITY is like 3-SATISFIABILITY except that there are exactly k literals per clause. Assuming from above that 3-SATISFIABILITY is **NP**-complete, show that 4-SATISFIABILITY is also **NP**-complete. Deduce that k-SATISFIABILITY is **NP**-complete for every $k \geq 3$. **[12 marks]**

END OF PAPER