May 2019

Coventry University

Faculty of Engineering, Environment and Computing

380CT

Theoretical Aspects of Computer Science

Instructions to candidates

Time allowed: **2** hours

This is **a Closed Book** Examination

Answer: All Questions

The total number of questions in this paper: 4

All questions carry equal marks (25 marks each)

Write your answers in this questions paper, in the provided framed boxes labelled Solution.

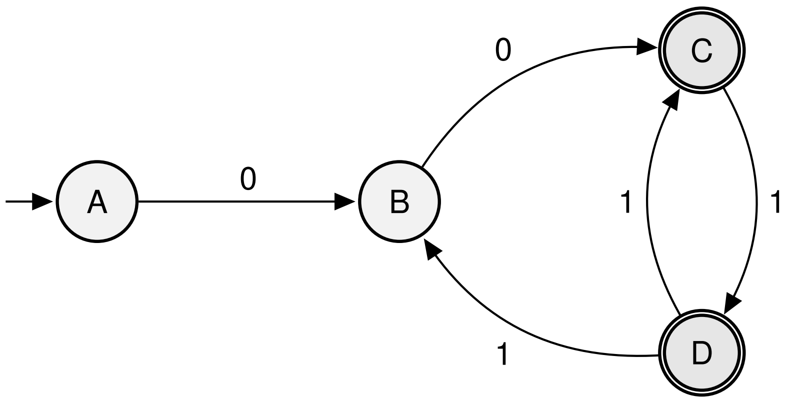
If more space is needed then use the extra pages provided at the end of this paper, and write a note in the appropriate box.

These pages have Extra page .../2 in their header.

These pages can also be used for rough work. (Please cross it out.)

You must hand this question paper in at the end of the examination**.**

❶ Consider the Non-deterministic Finite Automaton (NFA) $\mathcal{M}$ defined by the following transition diagram:



6 marks

a) Produce the formal specification of $\mathcal{M}$.

Solution

$\mathcal{M} = (Q, \Sigma, \delta, q_{\text{start}}, F)$ where:

- $Q = \{$A, B, C, D$\}$
- $\Sigma = \{0, 1\}$
- $q_{\text{start}} =$ A
- $F = \{$C, D$\}$
- Transitions table for δ: (Complete the transition table below)

	0	1
$\rightarrow$A	{B}	$\emptyset$
B	{C}	$\emptyset$
$*$C	$\emptyset$	{D}
$*$D	$\emptyset$	{B, C}

("$\rightarrow$" denotes the start state, and "$*$" denotes accepting states.)

Question ❶ is continued on the next page

5 marks

b) For each of the following strings, state if it will be *accepted* or *rejected* by $\mathcal{M}$. (Tick a box).

Solution

String	Accept	Reject
ε		✓
11		✓
000		✓
111		✓
00110	✓	

6 marks

c) Use the *subset construction method* to produce the transition table for a Deterministic Finite Automaton (DFA) equivalent to $\mathcal{M}$.

Solution

	0	1
$\rightarrow\{A\}$	$\{B\}$	$\emptyset$
$\{B\}$	$\{C\}$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$
$*\{C\}$	$\emptyset$	$\{D\}$
$*\{D\}$	$\emptyset$	$\{B,C\}$
$*\{B,C\}$	$\{C\}$	$\{D\}$

(Use "$\rightarrow$" to denote the start state, and "$*$" to denote the accepting states.)

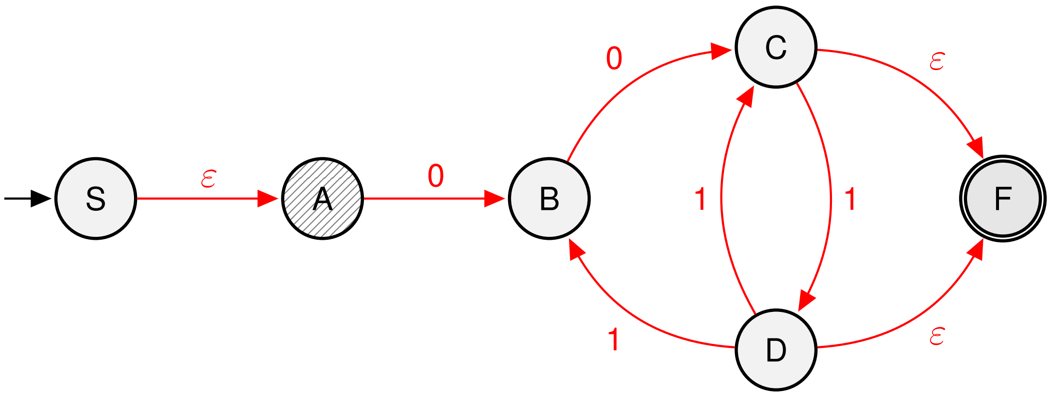
Question ❶ is continued on the next page

8 marks

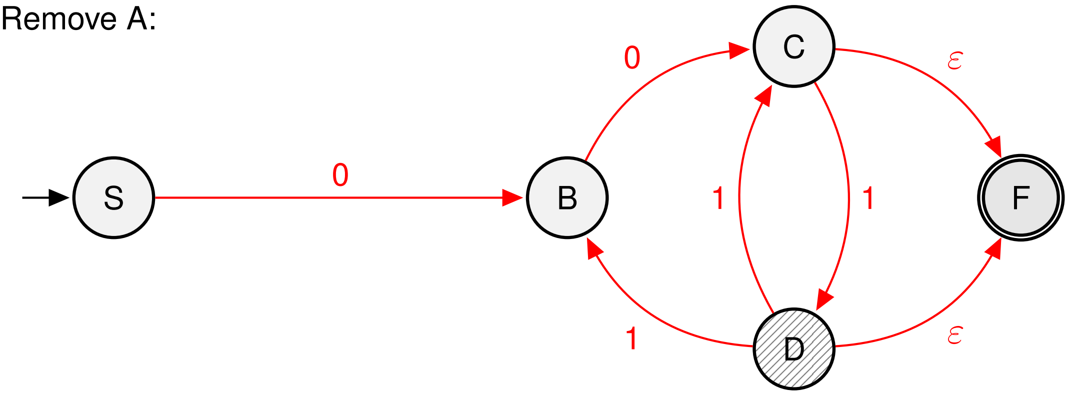
d) Use the *GNFA algorithm* to produce a regular expression for the language recognized by $\mathcal{M}$. (Add the missing transitions and labels.)

Solution

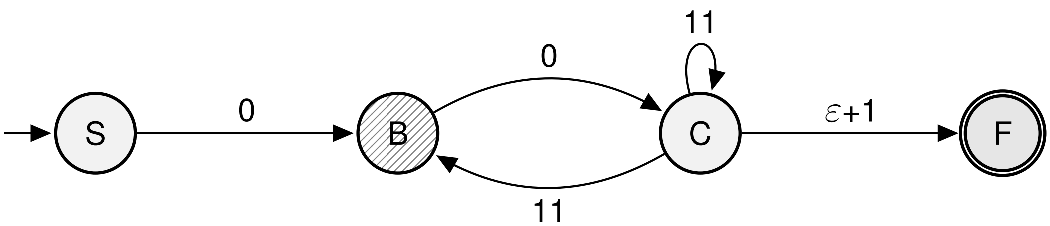
First convert the NFA into a Generalized NFA (GNFA)



Remove A:

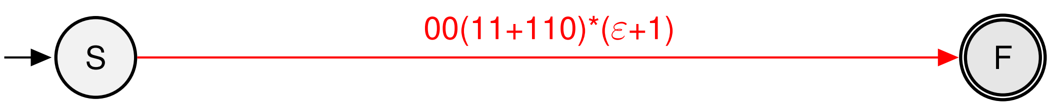


Remove D: (This is completed fully – complete the next ones.)



Remove B:

Remove C:



Continue

❷ Consider the following context free grammar G with alphabet $\Sigma = \{\mathsf{a}, \mathsf{b}\}$ and production rules:

$$\begin{array}{lcllll} S & \to & A & \mid B & \mid AB \\ A & \to & AA & \mid B & \mid \mathsf{a} \\ B & \to & BB & \mid A & \mid \mathsf{b} \end{array}$$

3 marks

a) Use a regular expression to describe the pattern of strings generated by the rule $V \to V\,V \mid \mathsf{a}$.

Solution

$\mathsf{a}^+ = \mathsf{aa}^* = \mathsf{a}^*\mathsf{a}$

3 marks

b) Use a regular expression to describe the pattern generated by the last two rules:

$$\begin{array}{lcllll} A & \to & AA & \mid B & \mid \mathsf{a} \\ B & \to & BB & \mid A & \mid \mathsf{b} \end{array}$$

Solution

$(\mathsf{a}+\mathsf{b})^+ = \Sigma^+ = \Sigma\Sigma^* = \Sigma^*\Sigma$

3 marks

c) G actually generates every string in Σ^* except one string – what is it?

Solution

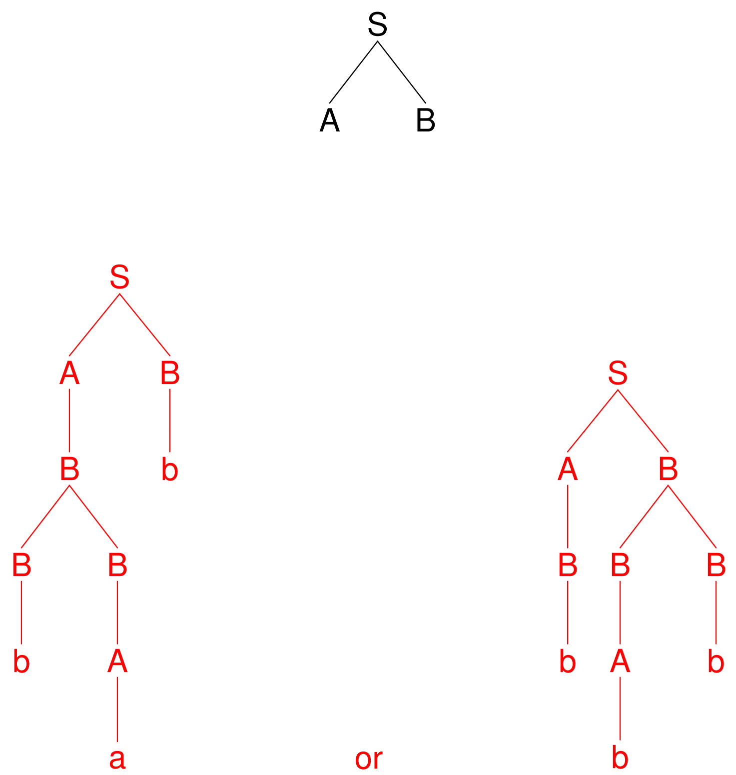
ε, the empty string.

Question ❷ is continued on the next page

3 marks

d) Finish the below **parse tree** for the string bab:

Solution



3 marks

e) Finish the below **derivation** for the string bba:

Solution

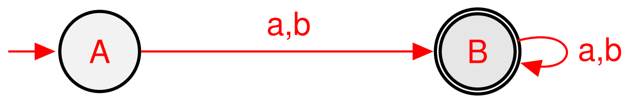
$S \to AB \to BB \to \text{b}B \to \text{b}BB \to \text{bb}B \to \text{bb}A \to \text{bba}$

Question ❷ is continued on the next page

4 marks

f) Design an NFA with 2 states whose language is exactly the one generated by G.

Solution



2 marks

g) Can a 1-state NFA do the same? Justify your answer.

Solution

No.

If the state is accepting then it accepts ε, which is not acceptable. But if it is not accepting then it rejects all strings.

4 marks

h) The language generated by G can also be generated by a simpler grammar. Complete the set of rules given below to find it.

Write the missing *variables* or *terminals* in the given four boxes.

Solution

$$S \to \mathrm{a}\,\boxed{S} \mid \mathrm{b}\,\boxed{S} \mid \boxed{\mathrm{a}} \mid \boxed{\mathrm{b}}$$

Continue

8 marks

❸ a) Below is a list of claims about languages.

For each claim state if it is **correct** or **incorrect** giving a short **justification**.

The first one has been done as an example.

Solution

$\{a^\ell b^m c^n \mid \ell, m, n \geq 0\}$ is not regular.

Incorrect. *It is generated by the regular expression:* a*b*c*.

i. The language of all the month names in a year is regular.

Correct. Finite languages are all regular.

ii. The language generated by the grammar $S \to \mathrm{a} \mid SS$ is regular.

Correct. Generated language is a^+.

iii. The language of all syntactically valid Python code is context free.

Correct. They are designed to be so, to be able to build efficient compilers.

iv. It is possible to have a language L and its complement $\overline{L}$ both recognizable, yet L is not decidable.

Incorrect. When the recognizer of one of the two languages stops we know if it is in L or $\overline{L}$, so we can decide if it is in L (or outside it).

Question ❸ is continued on the next page

6 marks

b) Use O-notation to express the order of growth of the following expressions.

Solution

Expression	O-notation
2019	$O(1)$
$2019+n$	$O(n)$
$2019n+n^2$	$O(n^2)$
$n^{2019}+2019^{2019}$	$O(n^{2019})$
$n+\log n+n\log n$	$O(n\log n)$
$n^{2019}+n^{\log n}$	$O(n^{\log n})$

3 marks

c) You are given that

$$f(n)=O(\log n), \quad \text{and} \quad g(n)=O(2^n).$$

What is the order of the following functions:

Solution

- $f(n)+g(n)=O(2^n)$
- $f(n)\times g(n)=O(2^n\log n)$
- $f\big(g(n)\big)=O(n)$

Question ❸ is continued on the next page

d) Below is pseudocode for an algorithm that tries to decide if a given integer n is prime or not.

```
for d ← 1, ..., n do
    if d divides n then
        return false
    end if
end for
return true
```

4 marks

Why will this algorithm fail to decide the primality of n?

Solution

It will always return false, irrespective of whether n is prime or not. This is because all numbers are divisible by 1 and themselves (d takes the values 1 and n).

4 marks

e) How would you change the above pseudocode to correct this algorithm?

Solution

```
if n < 2 then
    return false
else if n = 2 then
    return true
end if
for d ← 2, ..., n − 1 do
    if d divides n then
        return false
    end if
end for
return true
```

Continue

❹ The *Shortest Common Superstring (SCS)* problem is defined as follows:

Shortest Common Superstring (SCS)

Given a finite set of strings $\{s_1, \ldots, s_n\}$ over an alphabet Σ, find a string that is as short as possible and contains each string s_i as a substring.

SCS is **NP-hard** given that $|\Sigma| \geq 2$. It has application in DNA sequencing.

Examples:

- $\{cab, ba, abc\}$ admits $cababc$ as the shortest superstring.
- $\{ab, cd\}$ admits $abcd$ as the shortest superstring (no overlap possible).

2 marks

a) Classify the above problem formulation as a **decision**, **search** or **optimization** problem. Justify your answer.

Solution

Optimization. Minimizing a function.

2 marks

b) Find a solution to $\{abc, a, ca, b\}$

Solution

cabc or abca.

2 marks

c) Find a solution to $\{bc, bca, abc, cab\}$

Solution

bcabc

Question ❹ is continued on the next page

3 marks

d) Complete the following **exhaustive search** algorithm to solve this problem.

Write in the provided three rectangles.

Solution

1: $best \leftarrow$ Concatenation of all the strings $s_1, \ldots, s_n$.
2: **for all** permutations $(t_1, \ldots, t_n)$ of $(s_1, \ldots, s_n)$ **do**
3: $\quad candidate \leftarrow t_1$
4: $\quad$ **for** $i \leftarrow 2, \ldots, n$ **do**
5: $\quad\quad$ Merge t_i into $candidate$ while maximizing their overlap.
6: $\quad$ **end for**
7: $\quad$ **if** $candidate$ is [shorter] than $best$ **then**
8: $\quad\quad best \leftarrow$ [candidate]
9: $\quad$ **end if**
10: **end for**
11: **return** [best]

3 marks

e) Estimate its worst-case cost using O-notation. Justify your answer.

Assume that the merging step costs $O(1)$ only.

(Refer to the line numbers in the pseudocode above.)

Solution

1: n (order only, not exact value)

2: $n!$

3: $1 \times n!$

4-6: $n \times n!$

7-9: $n!$

(10: 0)

11: 1

So, overall cost is: $O(n \cdot n!)$.

Question ❹ is continued on the next page

f) If $|\Sigma| = 1$ then SCS can be solved in polynomial time. Outline an algorithm that achieves this over $\Sigma = \{1\}$.

Hint: What is the solution to $\{11, 1111, 1\}$?

2 marks

Solution

Then, SCS asks for the shortest string 1^k that contains the set of strings $\{1^{\ell_1}, 1^{\ell_2}, \ldots, 1^{\ell_n}\}$. It suffices to choose $k = \max\{\ell_1, \ell_2, \ldots, \ell_n\}$

g) Consider the following greedy solution method for this version:

Greedy

1: Select a random permutation $(t_1, \ldots, t_n)$ of $(s_1, \ldots, s_n)$
2: $solution \leftarrow t_1$
3: **for** $i \leftarrow 2, \ldots, n$ **do**
4: Merge t_i into $solution$ to maximize their overlap.
5: **end for**
6: **return** $solution$

5 marks

Simulate the greedy method on $S = \{01, 11, 1011, 0110\}$.

Solution

E.g. $(t_1, t_2, t_3, t_4) \leftarrow (01, 11, 1011, 0110)$, and $solutions \leftarrow 01$.

i	t_i	$solution$
2	11	011
3	1011	1011
4	0110	10110

Question ❹ is continued on the next page

6 marks

h) Specify a meta-heuristic which can be used to improve the greedy method.

Write **pseudocode** and **be specific** to SCS.

Solution

For example, GRASP:

1: **while** termination condition is not met (e.g. at least 90% vertices visited, or simply repeat 100 times) **do**
2: Generate a candidate solution s using a subsidiary greedy randomized constructive search
3: Perform subsidiary local search around s
4: **end while**

Termination condition could be to repeat the loop a fixed number of times, or to watch if improvements are made or not.

subsidiary greedy randomized constructive search could be the greedy algorithm with a different permutations of S.

subsidiary local search could be implemented by swapping used and unused elements of the set.

The End