2202-COL226 Minor1

Utkarsh Singh

TOTAL POINTS

23 / 40

QUESTION 1

Q1 8 pts

1.1 Q1 (a) **4 / 4**

✓ **+ 4 pts** *regular expression specifies exactly A+*

+ 2 pts minor errors (regular expression specifies A*)

+ 0 pts Incorrect/Unattempted

1.2 Q1 (b) **4 / 4**

✓ **+ 4 pts** *DFA recognizes exactly A+*

+ 2 pts minor errors (DFA recognizes A*/NFA recognizing A+)

+ 0 pts Incorrect/Unattempted

QUESTION 2

2 Q2 **9 / 10**

(a) Unambiguous CFG design

✓ **+ 5 pts** *(a) Correct unambiguous grammar (with only left-linear and right-linear productions)*

+ 4 pts Correct unambiguous grammar with one rule of the form $$N \rightarrow xy, ~ N \in \mathcal{N}, x, y \in \mathcal{T}$$, and other rules being left-linear/right-linear

+ 2 pts CFG recognizes $$\{a^nb^n ~|~ n\geq 0\}$$ and is left-linear/right-linear

+ 0 pts (a) Correct grammar but not left-linear/right-linear

+ 0 pts (a) Grammar does not recognize $$\{a^nb^n ~|~n > 0\}$$ / Grammar is ambiguous / Not Attempted

(b) Proof of unambiguity

+ 5 pts (b) All correct

✓ **+ 1 pts** *(b) Correct Induction Variable*

+ 1 pts (b) Correct Base Case

✓ **+ 1 pts** *(b) Correct IH*

✓ **+ 2 pts** *(b) Correct invocation of IH*

+ 0 pts (b) Incorrect / Not Attempted

The base case is string "ab" here not "aSb".

QUESTION 3

3 Q3 **2 / 7**

+ 7 pts Correct CFG and justification of $$L = L(G)$$

+ 5 pts Correct CFG but justification of $$L = L(G)$$ missing/incorrect

✓ **+ 1 pts** *CFG generates superset of $$a^mb^n | m,n >= 0, m \neq n$$*

✓ **+ 1 pts** *A language L is said to be a context-free language (CFL), if there exists a CFG G, such that $$L=L(G) $$*

+ 0 pts CFG does not generate $$a^mb^n | m,n >= 0, m \neq n$$

+ 0 pts Incorrect/Not attempted/No CFG defined

Check ab

QUESTION 4

4 Q4 **4 / 15**

EBNF

- **0 pts** Correct
- **2 pts** Incorrect precedence order of operators
- **2 pts** Incorrect associativity of operators

✓ **- 1 pts** *Minor errors in EBNF specification*

- **5 pts** Incorrect / Not attempted

CFG

- **0 pts** Correct
- **3 pts** Simply converted EBNF to CFG without removing left recursion.
- **1 pts** Minor errors in CFG specification

✓ **- 5 pts** *Incorrect / Not attempted*

FIRST and FOLLOW sets

- **0 pts** Correct
- **1 pts** Incorrect FIRST of start symbol
- **1 pts** Incorrect FOLLOW of start symbol
- **1.5 pts** Incorrect FIRST of other non-terminals
- **1.5 pts** Incorrect FOLLOW of other non-terminals

✓ **- 5 pts** *Incorrect / Not attempted*

1 1 + 2 cannot be generated. Further EBNF specification was required.

COL226: Programming Languages

II semester 2022-23

Thu 09 Feb 2023 | Minor 1 | LH325 | 60 minutes | Max Marks 40

Instructions.
1. Answer only in the space provided for each question in the question paper itself.
2. Write your name, Entry number and signature on the top line of every page
3. No extra sheets will be provided. You may do your rough work in the separately provided sheets. 4. Answers will be judged for correctness, efficiency and elegance.
5. If there are minor mistakes in the question, correct them explicitly and answer the question accordingly.
5. If the question is totally wrong, give adequate reasons why it is wrong with detailed counter-examples, if necessary.

---

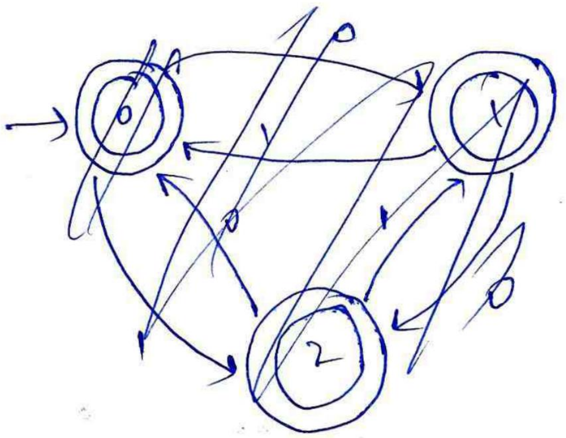
1. **[4 + 4 = 8 marks]** By now you are all aware that the ASCII characters are encoded as bit strings. For the purpose of this problem assume that the alphabet $A = \{a, b, c, d\}$, $B = \{0, 1\}$ the characters in A are encoded as the bit-strings $\{00, 01, 10, 11\}$ respectively. Hence the string $acdaab \in A^+$ is encoded as 001011000001.

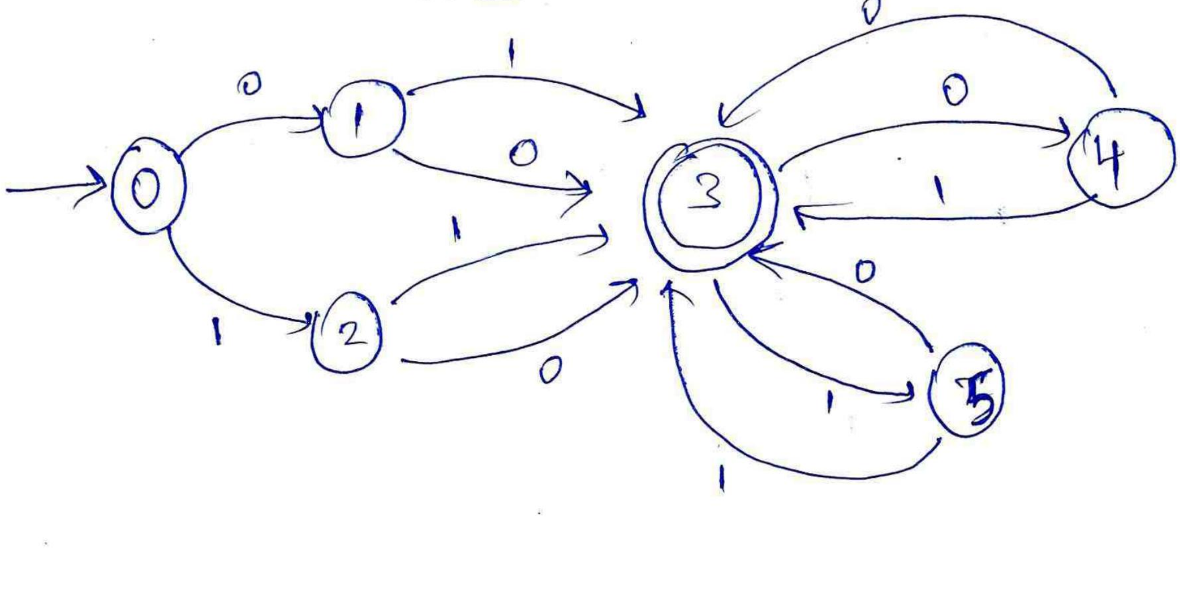
   (a) Design a regular expression to specify the encodings of the language A^+.

   (b) Design a DFA to recognize exactly the language encoded by A^+ (*Use your intuition, don't waste time trying a formal implementation of the algorithms described in class*).

a) $(00|01|10|11)^+$

b)





2. **[5 + 5 = 10 marks]** Even though the language $\{a^n b^n \mid n > 0\}$ cannot be generated by a pure right-linear grammar or a pure left-linear grammar, it can still be generated by a combination of left-linear and right-linear productions.

   (a) Design an unambiguous grammar (with only left-linear and right-linear productions) that generates the language $\{a^n b^n \mid n > 0\}$.

   (b) Give a formal proof that your grammar is unambiguous.

a) $G = \langle \{S, X, Y\}, \{a, b, \epsilon\}, P, S \rangle$

$S \longrightarrow aX$
$X \longrightarrow Yb$
$Y \longrightarrow S \mid \epsilon$

b) As we can see, if we start from $S \rightarrow aX$, then our next step is automatically fixed, i.e we have no other option. $X \rightarrow Yb$, so till now our generated string will be aSb. This implies we will get a unique tree to form aSb.

Induction Base case: aSb is formed by a unique tree.

Induction hypothesis: (aaa... n times S bbb... n times) is formed by a unique tree.

Induction step: To show: (aaa... n+1 times S bbb... n+1 times) is also formed by a unique tree.

aaa... (n times) S bbb... (n times) $\Longrightarrow$ aaa... (n+1) times X bbb... n times
$\Downarrow$
aaa... (n+1) times S bbb... (n+1) times $\Longleftarrow$ aaa... (n+1) times Y bbb... (n+1) times

$\therefore$ I have a unique derivation to reach aaa... (n+1) times S bbb... (n+1) times

$\therefore$ My induction step is also proved.

So aaa... n times S bbb... n times $\Longrightarrow$ $a^n b^n$ (unique step)
this shows $a^n b^n$ can be formed by a unique derivation.

3. [**7 marks**] Prove that the language $\{a^m b^n \mid m, n \geq 0, m \neq n\}$ is context-free.

Inorder to prove the language context free, I will just show a grammar that is context free and forms the language.

$$G = \langle \{S\}, \{a, b, \epsilon\}, P, S \rangle$$

$$S \longrightarrow \epsilon \mid a \mid b \mid aS \mid Sb$$

$\because$ for each production rule

$$\alpha \longrightarrow \beta$$

$$\alpha \in \{S\} \text{ and } \beta \in \{S \cup \{a, b, \epsilon\}\}^*$$

$\because$ My grammar is context free

$\therefore$ My language is also context free

Name: Utkarsh Singh Entry: 2021CS10558 Signature: Utk

4. [**5 + 5 + 5 = 15 marks**] Consider an expression language involving the parentheses symbols "(", ")" and infix operators "+", "-" "*", "/", " ^ " for addition, subtraction, multiplication, division and exponentiation respectively such that

- "+" and "-" are left associative and have equal precedence and are lower than
- "*" and "/" which are also left associative and have equal precedence and in turn are lower than
- " ^ " which is right associative.

(a) Design an EBNF specification for the above expression language. You may use `id` and `co` to denote identifiers and constants repsectively.

(b) Modify the CFG defined by your EBNF to obtain a LL(1) grammar for generating all meaningful expressions containing identifiers and constants.

(c) Validate the correctness of your LL(1) grammar by computing FIRST and FOLLOW sets for each non-terminal.

Grammar = < {S, T, X}, {(,), +, -, *, /, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0}, P, S >

$S \longrightarrow (S+T) \mid (S-T) \mid (T)$

$T \longrightarrow (T*X) \mid (T/X) \mid (X) \mid (S)$

$X \longrightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 0 \mid X$

We can prove that the above grammar is correct.