COL 352 Intro to Automata and Theory of Comput.

Minor 2, Sem II 2018-19, Max 40, Time 1 hr

Name _____ **Entry No.** _____ **Group**

Note (i) Write your answers neatly and precisely in the space provided with with each question including back of the sheet. You won't get a second chance to explain what you have written.
(ii) You can quote any result covered in the lectures without proof but any other claim should be formally justified.
(iii) You can make use of Dirichlet theorem - in any sequence $a + b \cdot i, i > 0$ where a, b are relatively prime there are infinitely many primes congruent to a modulo b.

1. Consider the language $L = (00 + 11)^+$. Describe the equivalence classes of strings over $\{0, 1\}$ of the relation R_L (Myhill Nerode relation). **(5)**

   There are five equivalence classes corresponding to the five states of the min state DFA. These are $\epsilon, (00 + 11)^+, (00 + 11)^* \cdot 0, (11 + 00)^* \cdot 1, ((r_3 \cdot 1 + r_4 \cdot 0) \cdot (0 + 1)^*$ where r_3 and r_4 represent the third and the fourth regular expressions. It can be verified that none of the states are equivalent as per R_L and only the second set of strings correspond to the accepting state.
   Common mistake was not even getting the number of classes correct. This was heavily penalised since it doesn't even account for all strings and shows a deficiency in understanding of minimum state automaton. Minor penalty if only states were given without characterization of the strings.

2. Are the following languages CFL ? Justify or prove otherwise. **(5×2)**

   (a) The language PAREN2 consists of all balanced strings over $(,), [,]$. For example, $([[]])[()()]$ is balanced but $([])]]$ is not. In other words, the two distinct parenthised strings should be individually balanced over the pairs $(,)$ and $[,]$ respectively but the balancing cannot be interspersed. Either the two expressions should be disjoint or one should be embedded inside the other.

   $$S \rightarrow ()|[]|SS|(S)|[S]$$

   Most solution came up with the grammar which is easy to verify and rigorous proof was not expected. Minor penalty if there were too many redundant productions without any justification. The PDA based justification requires more effort. In the least, for the PDA description, like set of states, stack symbols should have been given and and some commentary on the working of this machine. Just listing some transition tuples without any comment about the purpose is not adequate. Primarily one needs to bring out why this PDA is similar or different from the PDA for balanced parenthesis over one kind.

   (b) $\{0^i | i$ is composite $\}$.
   By using pumping lemma on a sufficiently long string $0^N = uvwxy$ where $1 \leq |v| + |x| \leq n \leq N$, all strings $0^{ki} \cdot 0^{N-k} \in L$ Let $N - k = a$ and $k = b$. To apply Dirichlet theorem, we need to ensure that a, b are co-prime. Note that we can choose $N > n$ where n is the parameter of the PL. So we can choose a prime $p > n$ and $N = p^2$ since N must be composite.
   The common mistakes were (i) Not applying pumping lemma properly by trying to select $|v| + |x|$ (ii) Trying to apply Dirichlet's theorem without satisfying the preconditions of co-prime. Note that $a + b \cdot i$ is never prime if b is a multiple of a. (iii) Assuming CF is closed under complementation and proving it for primes.
   A beautiful result called **Parikh's theorem** says that over a unary alphabet CFL = Regular but we have not covered it in this course and hence cannot be invoked without a proof which is non-trivial.

3. Describe a procedure to convert a well-formed (valid) regular expression r into an equivalent CFG G with some underlying justification. Illustrate this on the r.e. $(0 \cdot 1 + 1^*)^*$ for all strings over $\{0, 1\}$. **(10)**
Hint: Use the recursive definition of r.e.

   We will do this by induction on the length of the regular expressions. The base cases are the unit length symbols of Σ and ϵ. So these are the productions of the form $S \to a \quad a \in \Sigma$
   Suppose we can represent all regular expr upto length n. Then either

   - $r = r_1 + r + 2$ where $|r_i| < n$. Then add a production $S \to S_1|S_2$ where r_i can be generated by CFGs with start symbol S_i.
   - $r = r_1 \cdot r_2$, then using similar idea $S \to S_1 \cdot S_2$
   - $r = r'^*$ where $|r'| < n$. Then add the productions
     $S \to SS' \quad |\epsilon$ where S' derives r'.
   - $r = (r')$ where $|r'| < n$, then add $S \to (S)$.

   Since all the r.e.s can be constructed recursively, we can obtain a CFG for any given r.e. For the given expression, we can use the above construction in a bottom up fashion using adequate number of variables.

   Let $A \to 0 \quad B \to 1$ and $C \to A \cdot B$. Thus C derives the r.e. 01.
   Similarly $D \to D \cdot B|\epsilon$ allows D to derive 1^*. Subsequently let $E \to C|D$ allows E to derive the r.e. $01 + 1^*$. Finally $S \to SE|\epsilon$ derives the original r.e.

   Note that we can also add another production $S \to (S)$ to take care of parenthesis.
   Common deficiences were (i) Missing base cases. It becomes important when you have productions like $A \to 01$ where $01 \notin \Sigma$ but $01 \in \Sigma^2$. Without the correct rules you cannot have productions of the form $A \to \Sigma^k$ for any $k > 1$. (ii) Not applying the transformation according to the description, especially wrt definition of additional variables.

4. Given a CFL L describe an algorithm to decide if it contains any string NOT of the form $(0 \cdot 1)^i$ for some $i > 0$. (It need not contain all such strings). **(6)**

   Since $(0 \cdot 1)^i$ is regular so are strings that belong to the complement of this set, say R. Assume that the CFL is given in CNF. Using closure property of CFL under intersection with a regular language is a CFL. Therefore $R \cap L$ is a CFL. Now we can use the emptiness testing on $L' = R \cap L$ as an algorithm for the decision problem. We can set the parameter of the PL as $n = 2^k$ where k is the number of variables of the CFL L'.

   Many solutions modified the emptiness testing of L to also include checking for $(01)^i$ upto some length n that is related to the PL for the CFL L. This is not correct since the CFL L' has a different parameter value.
   Other attempts were related to generating all possible strings using a Grammar and checking. Without a proof of correctness and termination, no marks were given. Many of the attempts tried to somehow exploit the structure of the r.e. $(0 \cdot 1)^i$ when they should have focussed on just using the fact thatit is regular.

5. Consider the languages
   $L_1 = \{(01)^i | i \geq 0\}, \quad L_2 = \{0^i \cdot 1^i | i \geq 0\} \quad L_3 = \{0^i 1^i 2^i | i \geq 0\}$.
   Consider the following machine models where Q: states Σ input alphabet Γ: tape alphabet/Stack alphabet
   M_1 ordinary Turing machine $\delta_1 : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
   M_2 A Turing machine that is not allowed to overwrite, with transition function $\delta_2 : Q \times \Gamma \to Q \times \{L, R\}$
   M_3 A deterministic PDA with two stacks. $\delta_3 : Q \times \Sigma \cup \{\epsilon\} \times \Gamma \times \Gamma \to Q \times \Gamma^* \times \Gamma^*$.

For each Machine model, identify which languages it can recognize[1]. **(3 + 3 +3)**

Machine	Which languages does it recognize (proof not needed)
M_1	L_1 L_2 L_3
M_2	L_1
M_3	L_1 L_2 L_3

The underlying reasoning is that M_2 is equivalent to DFA and M_3 is equivalent to M_1. The proofs are not expected and takes some effort especially the characterization of M_2.

[1]Marks will be given only if you correctly identify all the languages for each machine