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.

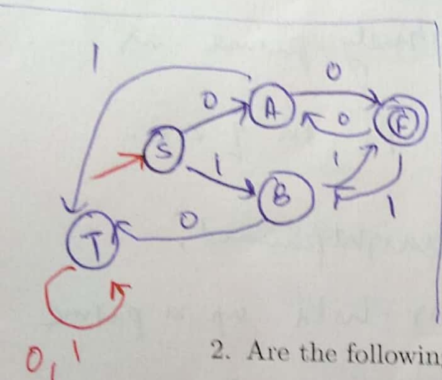
(iii) You can make use of Dirichlet theorem - in any sequence $a + b \cdot i, i > 0$ where a, b are relatively prime there are infinitely many primes congruent to a modulo b.

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)

The equivalence classes can be: A, B, T and F in the DFA shown. S contains only ε. Strings in F will only be accepted.

T = All ~~strings~~ strings where there is a 1 after odd number of 0's or a 0 after odd number of 1's : $T = \{0,1\}^* 01 \{0 1^*\} + \{0,1\}^* 10 \{0,1\}^*$

5



A = String that has odd 0's consecutively & otherwise property follows : ~~[illegible]~~

B = String that has odd 1's consecutively & o.w property follows

F = L.

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.

(Proof on next page bottom)

Yes,

$S \rightarrow () \mid [] \mid [S] \mid (S) \mid SS$. This grammar generates all ~~languages of~~ strings of PAREN2.

Moreover, I can also give a PDA s.t, I push a opening brace symbol on stack. When I encounter a close brace symbol, top of stack should have opening symbol of same type. When I read entire string, stack should be empty.

5

So as CFL & PDA exist, PAREN2 is a CFL.

(b) $\{0^i | i$ is composite $\}$.

NO, Consider 0^n. If this had been CFL, $\exists\, u, v, w, x, y$ s.t $u.v.w.x.y = 0^n$, $|vwx| < p$, $|v| + |x| \geq 1$, $n > p$ where p = pumping constant and $uv^iwx^iy \in L$ where L is this language

Then that means ~~$uvwxy^{i-1}$~~
$u\,v \cdot v^{i-1} w\, x\, x^{i-1} y \in L$

$0^{n+q_1(i-1)+q_2(i-1)} \in L$

$0^{n+(q_1+q_2)(i-1)} = 0^{n+q(i-1)} \in L$ when $q < p$.

If n & q are relatively prime then, $\exists$ a prime p'
s.t $p' = n+q(n)$ by dirichlet theorem

5

so 0^{p} will be in L which is a contradiction.

Now to choose n such that n & q are relatively prime.
Choose any to prime numbers p_1 & p_2 s.t $p_1 > p$, $p_2 > p$
and Take $n = p_1 \times p_2$.
Now for $\forall q < p$ q & n will be relatively prime as there will be no divisor of p_1 & p_2 in $q < p$.

Moreover if q happen to be 1, then it is straightforward, keep on increasing i till $n+q(i-1)$ hits up a prime number p'. We will get a contradiction then.

So, $(0^i \mid i$ composite$)$ is not a CFL.

Part a)

(a) Proof Grammar generates all desired strings → Induction on no. of rules applied
Base case: true.
Let it generate desired strings of by n rules $S \xrightarrow{n} \omega$ when $\omega \in L$.
Then $S \xrightarrow{n+1} \omega'$ then $\omega' = [\omega], (\omega), \omega\omega$ which statisfy the conditions.

(b) All strings of L are generated by grammar: Induction on length of string
Base: length 2 strings → [], ()
I.H: Let length n to be generated
length $n+2$ can be constructed as $[\omega], (\omega), \omega\cdot()$, $\omega\cdot[]$, $[]\cdot\omega$, (
which all are possible. So all strings in L are generated by grammar.

Scanned with CamScanner

Are you claiming that 1 variable suffices for all r.e.?
In your case since $S \to 0 | 1$, we should be able to derive $(0+1)^*$ from the productions you have provided

---

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.

Definition of an RE → If R_1 & R_2 is an R.E then $R_1 \cdot R_2$ is an R.E, $R_1 + R_2$ is an R.E, $(R_1)^*$ is an R.E and nothing else in R.E.

CFG: $S \to S \cdot S \mid S + S \mid (S)^* \mid a$ where a is a terminal of R.E.

$(0 \cdot 1 + 1^*)^* = S \to (S)^*$; $S \to S + S$, $S \to (S)^*$, $\to 1$, $S \to S \cdot S$, $\to 0$, $\to 1$

So $(0 \cdot 1 + 1^*)^*$ was generated by CFG provided.

5

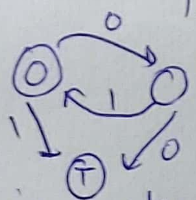
Proof: 1) Grammar generates an R.E. Induction on no. of rules applied
Base case: $S \to a$, a is an R.E so true
Let ~~...~~ a string after the application of n rules is an R.E.
i.e $S \xrightarrow{n *} \omega$ where ω is an R.E.
To prove $S \xrightarrow{n+1 *} \omega'$ where ω' is an R.E.
By application of all rules: $\omega' = (\omega)^*, \overset{a+\omega,\ \omega+a}{\uparrow}, \omega \cdot a, a \cdot \omega, \omega + \omega, \omega \cdot \omega$
and By definition of R.E all these are R.E if ω is R.E.

2) To prove that all R.E are generated by grammar.
This will be shown by rule mapping.
Say, ~~...~~ All R.E of length n are generated by S. Take ω_2 & ω_1 for instance.
Then new R.E of bigger length can be of the form $\omega_1\omega_2$, $\omega_2\omega_1$, $\omega_1+\omega_2$, $\omega_2+\omega_1$, ω_1+a, $a+\omega_1$, $a \cdot \omega_1$, $\omega_1 \cdot a$, $(\omega_1)^*$ by application of 1 more rule, which is possible by the rules given. Also $(a+\omega_1)$, (ω_1+a), $(\omega_1)^*$, $\omega_1 \cdot a$ & $a \cdot \omega_1$ will be all possible R.E ~~of~~ of length $n+1$. So grammar generates all R.E.

(Base case of Induction here is strings of length 1 i.e ~~...~~ a is an R.E & generated by $S \to a$)

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)

The Language $(0\cdot1)^i$ is a Regular Language i.e I can give a DFA as



(2)

So, $M = \{0,1\}^* - (0\cdot1)^i$ is also Regular as Regular Languages are closed under subtraction.

So, we need to show that CFL L doesn't accept any string of M.

Test all strings upto length 'n' in M in the CFL L when n is the pumping constant. If no string is accepted, then we have shown that L doesn't accept any string of M. why?

This is because, If there had been any string accepted of length $>n$, then I could use pumping lemma $u v^i w x^i y$ with $i=0$ to bring an accepted string of length $<n$ recursively. So, there had to be atleast 1 string of length $< L$ accepted.

... is the ... a CFL ... test ... emptiness. You should reduce to this. ... emptiness ... not but ... $\cap M$ is ... empty

5. Consider the languages
$L_1 = \{(01)^i | i \geq 0\}$, $L_2 = \{0^i \cdot 1^i | i \geq 0\}$ $L_3 = \{0^i 1^i 2^i | i \geq 0\}$.
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 ✓

(9)

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

Scanned with CamScanner