Date CS 302 HW4 Nofil Iqbal

Q1 a)

$G = (\{S, E, V, D\}, \{0,1,2,3,4,5,6,7,8,9,+,-,*,/,.,(,)\}, P, S)$

P:

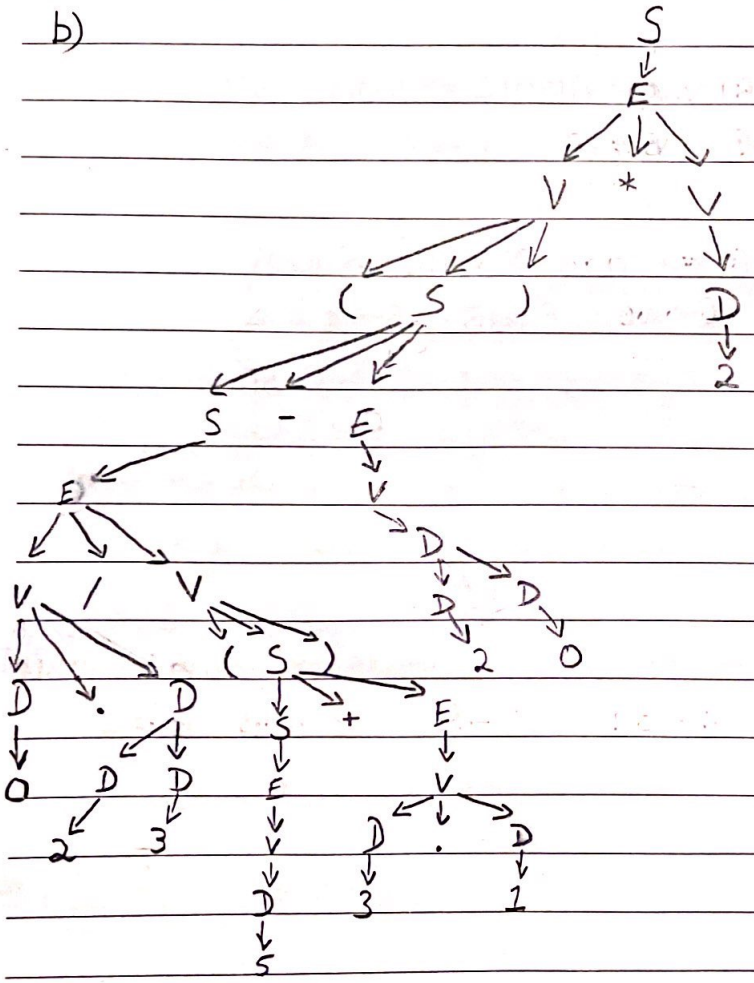
$S \rightarrow S+E \mid S-E \mid E$

$E \rightarrow V*V \mid V/V \mid V$

$V \rightarrow D \mid D.D \mid (S)$

$D \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 0 \mid DD$

b)



Date

Q2.

$G=(\{S, E, A\}, \{\wedge, \vee, \neg, T, F, x, y, z\}, P, S)$

P:

$S \rightarrow S \vee E \mid S \wedge E \mid E$

$E \rightarrow (S) \mid \neg(S) \mid A$

$A \rightarrow x \mid y \mid z \mid T \mid F \mid \neg A$

Q3.

5.1.2 b)

$E \Rightarrow_{rm} 1001$: $S \Rightarrow A1B \Rightarrow A10B \Rightarrow A100B \Rightarrow A1001B \Rightarrow A1001 \Rightarrow 1001$

$S \rightarrow A1B \quad B \rightarrow 0B \quad B \rightarrow 0B \quad B \rightarrow 1B \quad B \rightarrow \varepsilon \quad A \rightarrow \varepsilon$

$E \Rightarrow_{Lm} 1001$: $S \Rightarrow A1B \Rightarrow 1B \Rightarrow 10B \Rightarrow 100B \Rightarrow 1001B \Rightarrow 1001$

$S \rightarrow A1B \quad A \rightarrow \varepsilon \quad B \rightarrow 0B \quad B \rightarrow 0B \quad B \rightarrow 1B \quad B \rightarrow \varepsilon$

c)

$E \Rightarrow_{rm} 00011$: $S \Rightarrow A1B \Rightarrow A11B \Rightarrow A11 \Rightarrow 0A11 \Rightarrow 00A11 \Rightarrow 000A11 \Rightarrow 00011$

$S \rightarrow A1B \quad B \rightarrow 1B \quad B \rightarrow \varepsilon \quad A \rightarrow 0A \quad A \rightarrow 0A \quad A \rightarrow 0A \quad A \rightarrow \varepsilon$

$E \Rightarrow_{Lm} 00011$: $S \Rightarrow A1B \Rightarrow 0A1B \Rightarrow 00A1B \Rightarrow 000A1B \Rightarrow 0001B \Rightarrow 00011B \Rightarrow 00011$

$S \rightarrow A1B \quad A \rightarrow 0A \quad A \rightarrow 0A \quad A \rightarrow 0A \quad A \rightarrow \varepsilon \quad B \rightarrow 1B \quad B \rightarrow \varepsilon$

Date

5.1.3:

Base case: We start with single characters with no operations. To recognize a single character "a", we can have a CFG: $S \rightarrow a$

Inductive Step: Suppose that for any regular expression, with fewer than k operators, we can construct a CFG that produces the same language. Then take a language L be represented by the regular expression R, such that R has k operators.

We have 3 possible operations:

1) Union: If $R = R_1 + R_2$, assume R_1 is produced by a CFG with start symbol S_1 and that R_2 is produced by a CFG with start symbol S_2. We create a new start state S with the production rule $S \rightarrow S_1 \mid S_2$. If the first production we use is $S \rightarrow S_1$, then we will produce exactly the strings matched by R_1, by our inductive hypothesis. If the first production we use is $S \rightarrow S_2$, then we will produce exactly the strings matched by R_2

2) Concatenation: If $R = R_1 R_2$, then we create a new CFG with start state S and the production rule $S \rightarrow S_1 S_2$. By our inductive hypothesis, S_1 produces exactly the strings matched by R_1 and S_2 produces exactly the strings matched by R_2. So S will produce exactly the strings in $R_1 R_2$

3) Kleene Star: If $R = R_1^*$, then we create a new CFG with start state $S \rightarrow S S_1 \mid \varepsilon$. Then we can prove by induction that a string with any number of copies of string matched by R_1 can be generated

Therefore, since every regex has an equivalent CFG, every regular language is context free.

Date

5.1.7 a)

Inductive Hypothesis:

Sentential form A_n of length n has exactly 1 non-terminal (i.e. S) and has the form a^iSb^j $(0 \leq i,j)$

Inductive Step:

From production rules of grammar, you can see that after substituting any of the first two productions, form of Sentential form A_{n+1} remains $a^{i+1}Sb^j$ $(0 \leq i,j)$ or a^iSb^{j+1} $(0 \leq i,j)$. Upon applying production 3 or 4 form of string derived will have form $a^{i+1}b^j$ $(0 \leq i,j)$ or a^ib^{j+1} $(0 \leq i,j)$ respectively.

Basis of Induction:

For sentential form of length 1, our assumption that it has form a^iSb^j $(0 \leq i,j)$ is true

Now, because every string derived from gammar have form $a^{i+1}b^j$ $(0 \leq i,j)$ or a^ib^{j+1} $(0 \leq i,j)$, we can say that substring "ba" will never occur in any string derived from this grammar.

b) Set of all strings having arbitrary number of a's followed by arbitary number of b's

Date

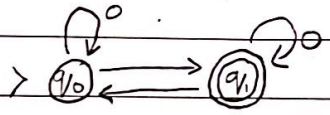
5.1.4 a) A right linear grammar is a 4 tuple : (V, T, P, S) where

V → finite set of non-terminals
T → finite set of terminals
P → finite set of productions
S → Start symbol

Converting a right linear grammar to an equivalent Finite State Automata:

1: S is the start state
2: Associate with each rule of the form $A \rightarrow wB$ a transition in a finite state automata from state A to state B reading w
3: Associate each rule of the form $A \rightarrow w$ with a transition from state A reading w to a final state F.

This produces the final state automata:



Hence, any language that produces finite state automata, must generate regular grammar.