07/03/2021 **Problem Sheet - 5** 106119100

CSPC 41 Rajneesh Pandey

Question 1 :

(a) $L = \{a^n b^m : n \leq m+3\}$

$L = \{\phi, b, bb, bbb, \ldots, a, aa, aaa, ab, \ldots\}$

there should be at most 3'a without b.

$$\left.\begin{array}{l} S \longrightarrow AAAB \\ A \longrightarrow a \mid \lambda \\ B \longrightarrow ABb \mid \lambda \end{array}\right\}$$

Valid : ab, aab, bb

Invalid : aaaa, aaaaab, aaaaaabb,

(b) $L = \{a^n b^m : m-1 \neq n\}$ $L_2 = \{a^n b^m, n > m-1\}$

$L_1 = \{a^n b^m : n < m-1\}$

$L_1 = \{a^n b^{n} b^k : n \geq 0, k \geq 2\}$ $L_2 = \{a^k a^n b^n ; k \geq 0, n \geq 0\}$

$(L = L_1 \cup L_2)$

Let $X \longrightarrow aXb \mid \lambda$

$A \longrightarrow aA \mid \lambda$

$B \longrightarrow bB \mid bb.$

$$\left.\begin{array}{l} S \longrightarrow XB \mid AX \\ X \longrightarrow aXB \mid \lambda \\ A \longrightarrow aA \mid \lambda \\ B \longrightarrow bB \mid bb \end{array}\right\}$$

(c) $L = \{a^n b^m : n \neq 2m\}$

$L_1 = \{a^n b^m : n < 2m\}$ $\quad$ $L_2 = \{a^n b^m ; n > 2m\}$

$L_1 = \{a^n b^m : 0 \leq n < 2m\}$ $\quad$ $L_2 = \{a^n b^m ; n > 2m \geq 0\}$

$S_1 \rightarrow aaS_1b \mid A$
$A \rightarrow aA \mid a$

$S_2 \rightarrow aaS_2b \mid AB$
$A \rightarrow a \mid \lambda$
$B \rightarrow bB \mid b$

$S \rightarrow S_1 \mid S_2.$

(d) $L = \{a^n b^m : 2n \leq m \leq 3n\}$

for $m \geq 2n$: $S_1 \rightarrow aS_1bb \mid B$
$B \rightarrow bB \mid \lambda$

for $m \leq 3n$: $S_2 \rightarrow aS_2BBB$
$B \rightarrow b \mid \lambda$

$S \rightarrow S_1 \cap S_2$: $S \rightarrow aSbb \mid aSbbb \mid \lambda.$

(f) $L = \{w = \{a,b\}^* : na(v) \geq nb(v)$, where v is any prefix of $w\}$

$w = aaababbb$ we can them given grammar.

$S \rightarrow aSb \rightarrow aaSSbb \rightarrow aaaSbSbb \rightarrow aaabaSbbb.$
$\downarrow$
aaaba bab.

All strings would start with 'a'

$\therefore$, Any prefix 'v' of a string 'w' has $na(v) \geq nb(v)$

such as $v = a, aa, aaa, aaab, aaababb$ are all prefixes of 'w' with $na(v) \geq nb(v)$

Grammer $S \rightarrow aSb \mid SS \mid \lambda$

(e) $L = \{ w \in \{a,b\}^* : n_a(w) \neq n_b(w) \}$

$L_1 = \{ w \in \{a,b\}^* : n_a(w) < n_b(w) \}$

$\Rightarrow$ L_1 is generated by $S_1 \rightarrow bS_1 \mid aS_1S_1 \mid S_1aS_1 \mid S_1S_1a \mid b$.

$L_2 = \{ w \in \{a,b\}^* : n_a(w) > n_b(w) \}$

$\Rightarrow$ L_2 is generated by $S_2 \rightarrow aS_2 \mid bS_2S_2 \mid S_2bS_2 \mid S_2S_2b \mid a$

$L = L_1 \cup L_2 \Rightarrow \underline{S_1 \rightarrow S_1 \mid S_2}$.

(g) $L = \{ w \in \{a,b\}^* : n_a(w) = 2n_b(w) + 1 \}$

The grammer used to generate the language $L_1 = \{ a^n b^m : n = 2m \}$ is $S_1 \rightarrow aaS_1b \mid \lambda$!

The language $L_2 = \{ w \in \{a,b\}^* : n_a(w) = 2n_b(w) \}$ differs from L_1 in just the fact that L_2 contains all possible permutation of each string in L_1.

$\therefore$, The grammar used to generate L_2 is

$S_2 \rightarrow aaS_2b \mid aS_2ab \mid abS_2a \mid aS_2ba \mid bS_2aa \mid baS_2a$

$\mid S_2S_2 \mid aS_2bS_2a \mid \lambda$.

$\therefore$, we can modify S_2 to form a grammar S to generate languge L.

i.e., by changing the terminal condition of S_2

S_2 as follow

$S \rightarrow aaSb \mid aSab \mid absa \mid asba \mid bSaa \mid baSa \mid SS \mid$
$aSbSa \mid a$

(2) $S \rightarrow AB \mid \lambda$, $A \rightarrow aB$, $B \rightarrow Sb$ $\Big\}$ $A \rightarrow aSb$, $S \rightarrow ASb \mid \lambda$

Grammer, $S \rightarrow ASb \mid \lambda$

$A \rightarrow aSb$.

Language $L = \{\phi, ab, aabb, aaabbb, ---\}$

$L = \{a^n b^n, n \in N\}$.

(3) $S \rightarrow aaB$.

$A \rightarrow bBb \mid \lambda$

$B \rightarrow Aa$ $\Big\}$ $B \rightarrow Aa \Rightarrow B \rightarrow bBba \mid a$

$L = \{aaa, aababa, aabbababa, aabbbabababa, ---\}$

$L = \{aaa, aab^n(ba)^{n+2} ; n \in N\}$.

(4)

<statement> → <literal> | <statement> |
<for-statement> | <statement> |
<if-else statement> | <statement> |
<do-statement> <statement>.
<compound statement> | <statement> |
<return statement> <statement> | λ

<expression> → <identifier> | <constant> | <condition expression>
| <assignment expression>

(c) <if-else statement> → if (<expression>) <statement>
else <statement>.

(a) <constants> → <integer constant> |
<floating paint constant> |
<character constant> |
<string constant>.

<literal - statement> → <constants> |
const <identifier><constant> |
define <identifier> <constants>

(b) <for - statement> → for (<assignt expression>;
<condition expression>;
<assignment expression>)
{<statement>}.

(d) <do-statement> → do {<statement>}
while (<condition - statement>)

(f) <return - statement> → return <expression>

(e) <compound- statement> → {
<statement>
}.