Name: Ayush Jain
SAP ID: 60004200132
Div: B
Computer Engineering

FLAT- Tutorial 5

Q. 1) Use the pumping lemma to show that the given language is non-regular $L = \{a^n b^{n+1} \mid n > 1\}$

→ Given: $L = \{a^n b^{n+1} \mid n > 1\}$

$\therefore$ for $n = 2$, $L = aabbb$
$n = 3$, $L = aaabbbb$
$n = 4$, $L = aaaabbbbb$

LEMMA STATEMENT: It states that given any sufficiently long string accepted by FSM we can find a substring near the begining, that may be repeated as many times as we like and the resulting string will still be accepted by the same FSM.

As we can see that n consecutive 'a's are followed by 'n+1' consecutive 'b's. According to the pumping lemma there exist a constant 'm' such that 'z' is any word in L such that the length of 'z' is atleast m $(|z| \geq m)$ and we can write $z = uvw$ in such a way that

(1) $|uv| \leq m$
(2) $|v| \geq 1$
(3) $\forall i \geq 0$, uv^iw is in L

Let us choose a sufficiently large string z such that
$z = a^l b^{l+1}$
for some large l, where $l > 1$
$|z| = 2l+1$

Scanned with CamScanner

Now as per Pumping lemma, every string uv^iw for all $i \geq 0$ is in L. Likewise $|v| \geq 1$; i.e. $v \neq \epsilon$

Now, take $L = 3$

$Z = aaabbbb$

In the string, let us consider

$u = a$, $v = aab$, $w = bbb$

Now, for uv^iw, take $i = 1, 2$

$uv^1w = aaabbbb$

$uv^2w = aaabaabbbb$

$|uv^2w| = 10$

But for the language, the length of string should be $2l+1$, i.e. an odd number and it should have consecutive 'a's followed by consecutive b's. Also, the no. of b's should be one more than no. of a's.

Thus, the given language is not regular language.

Q. 2> Construct a PDA for the language consisting of equal number of a's and b's.

→ Step 1 : Defination

$(Q, \Sigma, \delta, Z_0, \Gamma, q_0, F)$

$Q \rightarrow$ set of states

$\Sigma \rightarrow$ input symbols (alphabets)

$\delta \rightarrow$ transitions.

$Z_0 \rightarrow$ stack top symbol

$\Gamma \rightarrow$ Stack alphabet

$q_0 \rightarrow$ initial state

$F \rightarrow$ Final state

Sundaram®

FOR EDUCATIONAL USE

Scanned with CamScanner

Step 2: Logic

If symbol read is 'a' and stack top is Z_0 or has 'x' as a stack top, push 'x' on stack.
If symbol read is 'b' and stack top is Z_0 or has 'y' as a stack top, push 'y' on stack.
If stack top is 'x' and b is read, pop 'x'
If stack top is 'y' and a is read, pop 'y'

$\therefore Q = \{q_0, q_f\}$ $\qquad \Sigma = \{a, b\}$

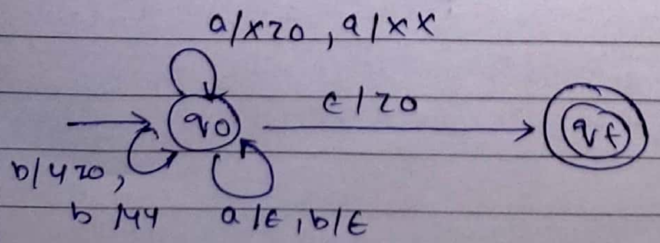
$Z_0 = Z_0$ $\qquad \Gamma = \{x, y\}$

$q_0 = q_0$ $\qquad F = q_f$

Step 3: Instantaneous description

$\delta(q_0, a, Z_0) = (q_0, xZ_0)$
$\delta(q_0, a, x) = (q_0, xx)$
$\delta(q_0, b, Z_0) = (q_0, yZ_0)$
$\delta(q_0, b, y) = (q_0, yy)$
$\delta(q_0, a, y) = (q_0, \epsilon)$
$\delta(q_0, b, x) = (q_0, \epsilon)$
$\delta(q_0, \epsilon, Z_0) = (q_f, Z_0)$

Step 4: Transition diagram



Scanned with CamScanner

Step 5 : Simulation

Let us consider the string aabbbaba

$\vdash (q_0, aabbbaba, z_0)$

$\vdash (q_0, abbbaba, xz_0)$

$\vdash (q_0, bbbaba, xx)$

$\vdash (q_0, bbaba, xz_0)$

$\vdash (q_0, baba, z_0)$

$\vdash (q_0, aba, yz_0)$

$\vdash (q_0, ba, z_0)$

$\vdash (q_0, a, yz_0)$

$\vdash (q_0, \epsilon, z_0)$

$\vdash (q_f, z_0)$

String is accepted.

Q. 3) Design a PDA for CFL that check the well formedness of parenthesis i.e. that language L of all ~~"balanced"~~ "balanced" String of two types of parenthesis say "( )" and "[ ]". Trace the sequence of moves made corresponding to input string (([ ])[ ])

→ Step 1 : Defination

$(Q, \Sigma, \delta, \Gamma, q_0, z_0, F)$

Q → Set of States

Σ → input alphabet

δ → transitions

Γ → Stack alphabet

q_0 → initial state

z_0 → stack symbol

F → Final State

Step 2: Logic

For every push '(' push 1 'x'
For every ')' pop 1 'x'
For every '[', push 1 '4'
For every ']' pop 1 '4'

$\therefore Q = \{q_0, q_1, q_f\}$ $\Sigma = \{(,), [,]\}$

$\Gamma = \{x, 4\}$ $q_0 = q_0$

$z_0 = z_0$ $F = q_s$

Step 3: Instantaneous description

$\delta(q_0, (, z_0) = (q_0, xz_0)$

$\delta(q_0, [, z_0) = (q_0, 4z_0)$

$\delta(q_0, (, x) = (q_0, xx)$

$\delta(q_0, [, x) = (q_0, 4x)$

$\delta(q_0, (, 4) = (q_0, 44)$

$\delta(q_0, [, 4) = (q_0, 44)$

$\delta(q_0,), x) = (q_1, \epsilon)$

$\delta(q_0,], 4) = (q_1, \epsilon)$

$\delta(q_1, (, x) = (q_0, xx)$

$\delta(q_1, [, x) = (q_0, 4x)$

$\delta(q_1, (, 4) = (q_0, x4)$

$\delta(q_1, [, 4) = (q_0, 44)$

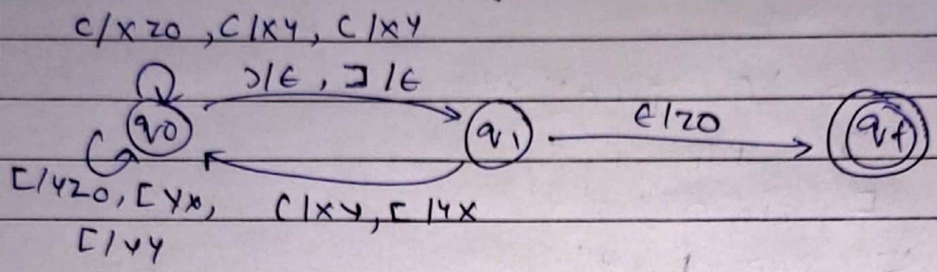
$\delta(q_1, \epsilon, z_0) = (q_f, z_0)$

Sundaram®

FOR EDUCATIONAL USE

Scanned with CamScanner

Step 4: Transition diagram



Step 5: Simulation

let string be (([])[])

⊢ (q0, (([])[]), z0)

⊢ (q0, ([])[]), xz0)

⊢ (q0, [])[]), xx)

⊢ (q0,])[]), yx)

⊢ (q1,])[]), xx)

⊢ (q1,)[]), xz0)

⊢ (q1, []), xz0)

⊢ (q0,]), yx)

⊢ (q1,), xz0)

⊢ (q1, ϵ, z0)

⊢ (q1, ϵ, z0)

⊢ (qf, z0)

String is accepted.

Q. 4 Let G be the grammar given by

$S \rightarrow aABB \mid aAA$

$A \rightarrow aBB \mid a$

$B \rightarrow bBB \mid A$

Construct NPDA that accepts the language generated by this grammar.

→ Step 1: Simplification

Since the given grammar consists of unit production $B \rightarrow A$ we eliminate it.

$S \rightarrow aABB \mid aAA$

$A \rightarrow aBB \mid a$

$B \rightarrow bBB \mid aBB \mid a$

The given grammar is in GNF format where $S \rightarrow a\alpha$, α can be non-terminals or ϵ.

Step 2: Instantaneous description

$\delta(q, \epsilon, S) = \{(q, aABB) \mid (q, aAA)\}$ R1

$\delta(q, \epsilon, A) = \{(q, aBB) \mid (q, a)\}$ R2

$\delta(q, \epsilon, B) = \{(q, bBB) \mid (q, aBB) \mid (q, a)\}$ R3

$\delta(q, a, a) = (q, \epsilon)$ R4

$\delta(q, b, b) = (q, \epsilon)$ R5

Sundaram

FOR EDUCATIONAL USE

Scanned with CamScanner

Step 3: Simulation

Let us consider String 'aabbaaaa'

⊢ (q, aabbaaaa, S)

⊢ (q, aabbaaaa, aABB)

⊢ (q, abbaaaa, ABB)

⊢ (q, abbaaaa, aBB)

⊢ (q, bbaaaa, BB)

⊢ (q, bbaaaa, bBBB)

⊢ (q, baaaa, BBB)

⊢ (q, baaaa, bBBBB)

⊢ (q, aaaa, BBBB)

⊢ (q, aaaa, aBBB)

⊢ (q, aaa, BBB)

⊢ (q, aaa, aBB)

⊢ (q, aa, BB)

⊢ (q, aa, aB)

⊢ (q, a, B)

⊢ (q, a, a)

⊢ (q, ε)

String is accepted.

Sundaram

FOR EDUCATIONAL USE

Q. 5) Differentiate between : DPDA and NDDA

Non-Deterministic Push Down Automata	Deterministic Push Down Automata.
1) Contains 7 tuples $(Q, \Sigma, \delta, \Gamma, z_0, q_0, F)$	1) Contains 7 tuples $(Q, \Sigma, \delta, \Gamma, z_0, q_0, F)$
2) For a particular input, NPDA will give different outputs)	2) For a particular iput, DPDA will give only one output.
3) It cannot determine next step of execution.	3) It can determine step of execution.
4) More powerful than DPDA	4) less powerful than NPDA
5) Language accepted is not a subset of languoge accepted by DPDA.	5) language accepted is a subset of language accepted by NPDA.
6) Not possible to convert every NPDA to corresponding DPDA.	6) Possible to convert every DPDA to corresponding NPDA.

Scanned with CamScanner

6) Write a short note on Chomsky hierarchy?

→ Hierarchy of grammars according to Chomsky is explained below as per the grammar types:

Type 0 - It is an Unrestricted grammars.

An unrestricted grammar is a 4-tuple (T, N, P, S) which consist of:

T = set of terminals

N = set of non-terminals

P = as set of productions, of the form $V \to \omega$

S = Start symbol.

example - Turing Machine (TM)

Type 1 - Context-sensitive grammars

All productions are of form.

$V \to \omega$ where $|V| < |\omega|$

$UAV \to U\omega V$ with $\omega != \epsilon$

i.e. $A \to \omega$ but only in the context of $U-V$

A context-sensitive grammar is equivalent to a linear bounded and context-sensitive language.

example - Linear bounded Automation (LBA)

Type 2 - Context-free grammars.

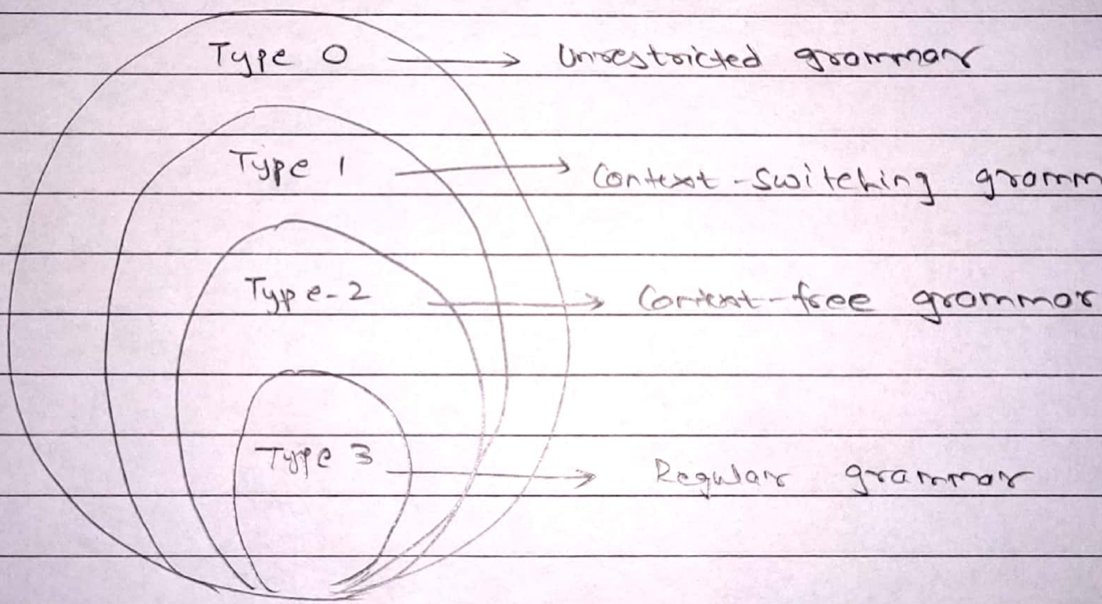
All productions are of form $A \to x$, where A is non terminal, x is a string of non-terminal and terminals. A context free grammar is equivalent to a pushdown automation (PDA) and to context free languages.

example - Pushdown Automation (PDA)

Scanned with CamScanner

Type 3- Regular grammars
All productions are of form A → xB
A → xB where A, B are non-terminal, x is a string of terminals.
Regular grammars are equivalent to regular sets and to finite automata.
example - Finite Automation (FA)

The Chomsky hierarchy is depicted in diagram given below:



Scanned with CamScanner