

**CISC 603-51- A-2021/SUMMER - THEORY OF COMPUTATION**

**Assignment - 4**

**Transforming Grammars, Normal Forms**

**Pushdown Automata**

**By,**

**Satheesh Swaminathan**

**Student ID:** **266582**

**6.1 #1**

S 🡪 a SB | bB

B 🡪 aA|b

A has no derivation

S 🡪 a SB | bB

B 🡪 b

Substitute for ‘b’

S 🡪 aSb|bb

**6.1 #3**

Let G = (V,T,S,P) is a CFG

A 🡪 x1Bx2

Let A and B are different variables

B 🡪 y1 | y2 | … yn

Define a new grammar G^ = (V,T,S,P^)

P^ is defined by eliminating A 🡪 x1Bx2 from P for B

A 🡪 x1y1x2| x1y2x2| … | x1ynx2

Then, S => G^u1Au2 => G^u1x1yix2u2

Generates same string in both cases

Therefore L(G^) = L(G)

**6.1 #5**

L(G) = L(G^)

A 🡪 x1Bx2

B 🡪 y1|y2| …|yn

Using theorm 6.1

A --> Ax1 | Ax2 | ... | Axn 🡪 Axi for i = 1, 2, 3 ... m

A --> y1 | y2 | ... | yn 🡪 yj where j = 1, 2, 3 ... n

L(G) cannot be equal to L(G^)

Unless

A 🡪 yj | yjB

B 🡪 xi | xiB

Therefore A and B has to be different variables.

**6.2 #4**

Remove unwanted productions

Remove lambda productions

S 🡪 baAB | baB | baA | ba

A 🡪 bAB | bB | bA | b

B 🡪 BAa | A | Aa | Ba | a

Since B 🡪 A, substitute for B 🡪 bAB | bB | bA | b

S 🡪 baAB | baB | baA | ba

A 🡪 bAB | bB | bA | b

B 🡪 BAa | Aa | Ba | a | bAB | bB | bA | b

Convert to Chomsky normal form

Introduce new variables Ca and Cb

S 🡪Cb CaAB | CbCaB | CbCaA | CbCa

A 🡪 CbAB | CbB | CbA | Cb

B 🡪 BACa | ACa | BCa | Ca | CbAB | CbB | CbA | Cb

Ca 🡪 a

Cb 🡪b

Introduce new variables D1, D2, D3, D4 and D5

S 🡪 CbD1 | CbD2 | CbD3 | CbCa

A 🡪 CbD4 | CbB | CbA | Cb

B 🡪 BD5 | ACa | BCa | Ca | CbD4| CbB | CbA | Cb

D1 🡪 CaD4

D2 🡪 CaB

D3 🡪 CaA

D4 🡪 AB

D5 🡪 ACa

Ca 🡪 a

Cb 🡪 b

**6.2 #5**

**6.2 #12**

Greibach normal form should be of the form A 🡪 ax

Substituting for a and b, we get the Greibach normal form

S --> aB | aS | aAS | aSS

A --> a

B --> b

**7.1 #1**

The strings accepted by this PDA is

Λ, abb, aabbbb, aabbbbbb, …..

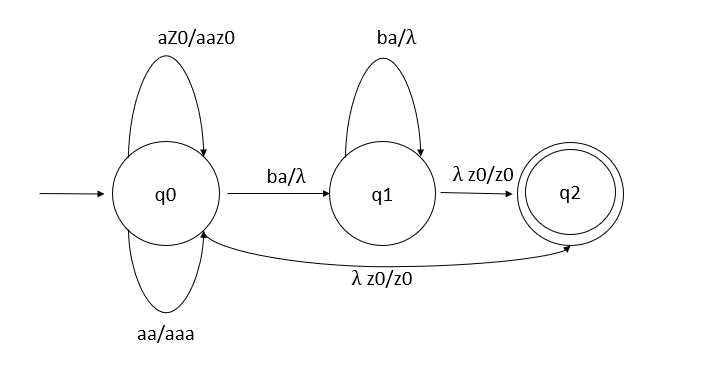
Where transition function δ is present as

δ(q0, a, z) = {(q1, aaz)}

δ(q1, a, a) = {(q1, aaa)}

δ(q1, b, a) = {(q2, λ)}

δ(q2, b, a) = {(q2, λ)}

δ(q2, λ, z) = {(q3, λ)}

**7.1 #2**

q0 is the initial state

each of pda contains the state, stack and input test strings

State q0 -- aabbbb – stack z

State q1, bbbb, aaz

State q1, bbbb, aaaaz

State q2, bbb, aaaz

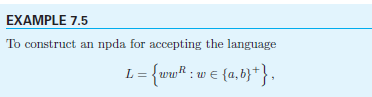
State q2, bb,aaz

State q2, b, az

State q2, λ, z

State q3, e, λ

**7.1 #5**



Q2 is the final state and q0 is the input state

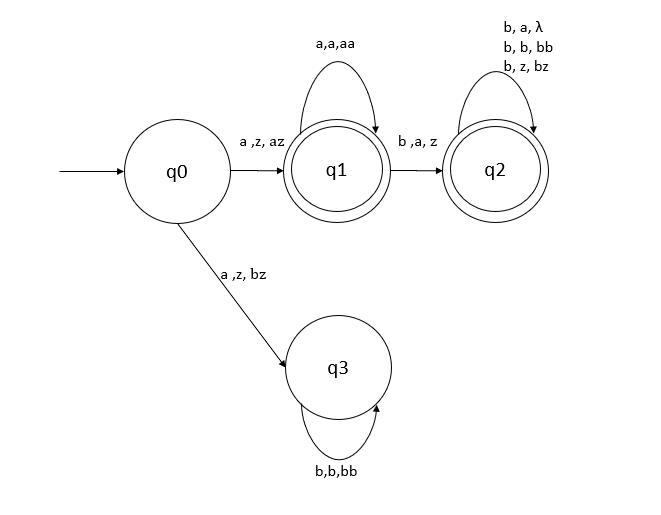
Let w1 is input while at state q1 and w2 is the string at state q2

Only when the test string is of the form w2R = w1, the pda will accept the test string and move to the final state q2

w2R = w1 – wwR

therefore we can say that the above npda accept only strings of the form wwR

**7.1 #7**



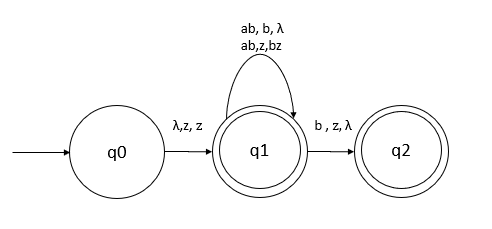
**7.2 #1**

Test strings the pda accepts of the form

abb

ababbb

abababbbb

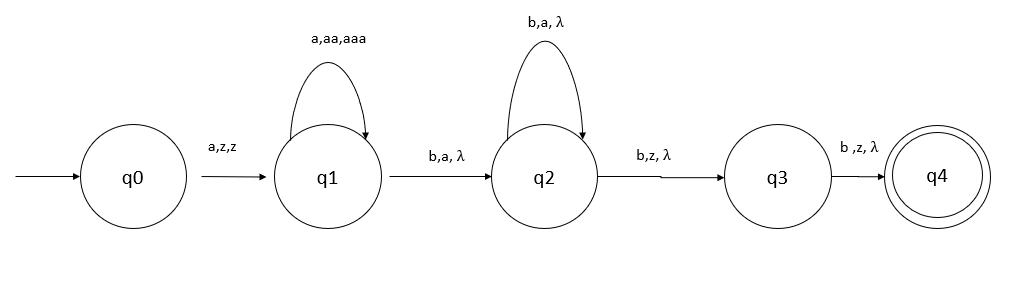


**7.2 #3**

S 🡪 abb

S 🡪 aabbbb

Language syntax: an+1b2n+2



**7.2 #5**

S 🡪 aaSbbbbS 🡪 aaSbbbb 🡪 aaaaSbbbbbbbb 🡪 aiSb2i

S 🡪 a

S 🡪 ai+1 b2i

Therefore the language generated byu the pda is of the from an+1 b2n

Which accepts the language described in this question.

**7.2 #7**

Convert to Greibach Normal Form

Remove unit productions

B 🡪 A

S 🡪 aABB|aAA

A 🡪 aBB|a

B 🡪 bBB|aBB|a

NPDA:

M = {{q0,q1,q2}, {a,b}, {S, A, B, z}, z, q0, z, {q2}}

{q0,q1,q2} – set of all states

{a,b} – inputs

{S, A, B, z} – stack symbols

Final state – q2

Transitions in npda are

δ(q0, λ, z) = {q1, Sz}

δ(q1, a, S) = {(q1, ABB), (q1, AA)}

δ(q1, a, A) = {(q1, BB), (q1, λ)}

δ(q1, a, B) = {(q1, BB)}

δ(q1, b, B) = {(q1, BB), (q1, λ)}

δ(q1, λ, z) = {(q2, z)}

