Module - 4

Push Down Automata:

Definition of Push Down Automata.

Definition

* A Push Down Automata (PDA) is a Seven Tuple such as

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

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

where

$Q \rightarrow$ is a Non-Empty Finite set of States.

$\Sigma \rightarrow$ is a Non-Empty finite set of input alphabets.

(Tape Symbol) $\Gamma \rightarrow$ is a Non-empty finite set of stack alphabets.

$\delta \rightarrow$ is a transition function from $Q \times (\Sigma \cup \varepsilon) \times \Gamma$ to finite sub set of $Q \times \Gamma^*$

$q_0 \in Q \rightarrow$ is the start state of machine

$Z_0 \in \Gamma \rightarrow$ is the initial symbol on the stack.

$F \subseteq Q \rightarrow$ is a set of Final states.

Scanned with OKEN Scanner

* PDA is mainly used in order to overcome from the drawback of Finite Automata

* Finite Automata recognizes Regular languages, inorder to Recognize Context Free Languages (CFL) we make use of PDA.

F.A → RL
PDA → CFL.

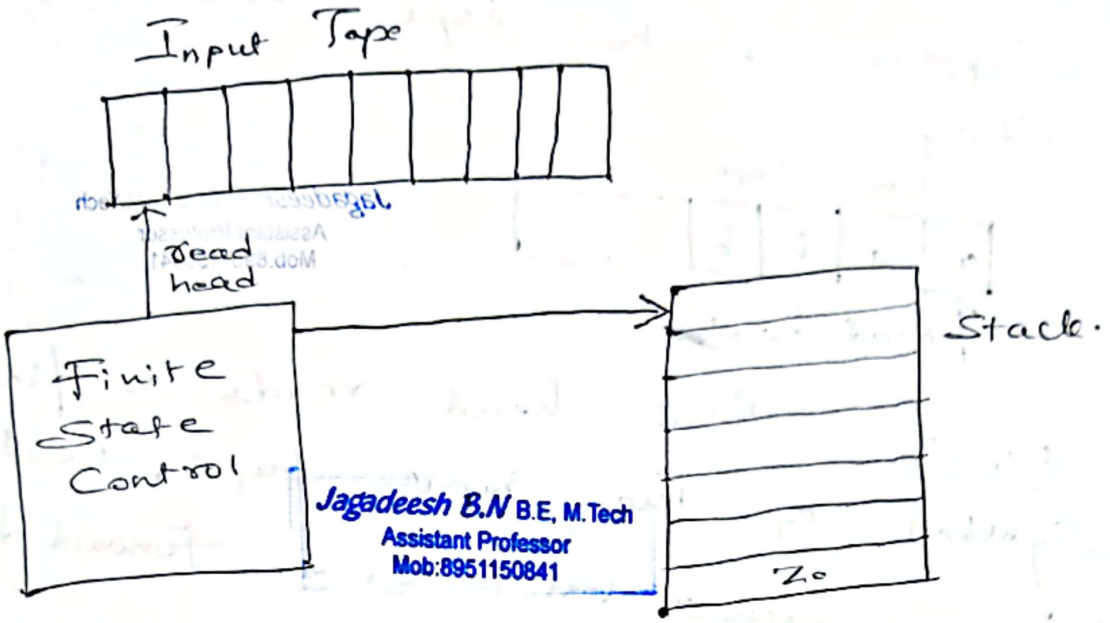
Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

* Problem with Finite Automata is that, F.A has Finite Amount of Memories and it can recognize only simple languages, Suppose if the language is Complex like $L=\{a^n b^n \mid n \geq 1\}$, F.A may not possible to use, because it requires More Memory to perform Comparisons and so on.

* Since Comparison Mechanisms in not their in F.A, we go for P.D.A to handle Complex languages.

Scanned with OKEN Scanner

* Model / Block diagram of PDA as shown below,



Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Block Diagram of PDA

(i) Input Tape :-

* Input Tape is mainly used to store the input string.

* Here the input tape is divided into number of cells, where each cell can store one - symbol at a time.

(ii) Finite State Control :-

* FSC is also a Finite Automata, which contains Finite State Machines.

* Finite State Control maintains 2 heads. i.e

(a) Read head

Scanned with OKEN Scanner

* Read head always points to the first symbol of the input tape.

* Suppose if the tape contains following string

Input Tape

a	a	b	b	

↑ Read head.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

* Firstly Read head reads the first symbol of the input tape, Read head increments. (or) moves in forward direction.

* Therefore symbol-by-symbol data will be read from the input tape

(iii) Stack :-

* Normal Finite Automata contains only 2 components viz Finite State Control and Input tape, but here it contains another component called "Stack".

* Stack is a Datastructure, which is used for storing the elements.

* On stack we can perform 2 operations viz Push Operation and Pop Operation

Scanned with OKEN Scanner

* Push Operation pushes the symbol onto the stack and Pop Operation removes the element from the stack.

* Push and Pop Operation will be performed on top of the stack.

* Second head always signifies the Top head.

* On the stack, initially top-most element of the stack contains Z_0.

* Stack produces either 2 states viz either Accept state (or) Reject state.

* Another advantage of using the stack is that the size of the stack is infinite/Unbounded i.e we can store any no of symbols onto the stack memory.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

The Languages of a PDA.

(1) Construct PDA for the language $L = \{ a^n b^n \mid n \geq 1 \}$.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

(Ans)

Step1:- languages $L = \{ \overset{n=1}{ab}, \overset{n=2}{aabb}, \overset{n=3}{aaabbb} \ldots \}$ procedures are has follows viz,

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob 8951150841

(i) Initially push all the a's on to the stack.

(ii) Whenever b occurs change the state, and pop a from the stack.

(iii) Repeat step (ii) until the stack is empty.

Step2:- Below shows the transition function of PDA,

String:- | a | a | a | b | b | b | ∈ |

Transition Function

(i) $\delta(q_0, a, Z_0) = (q_0, aZ_0)$

(ii) $\delta(q_0, a, a) = (q_0, aa)$

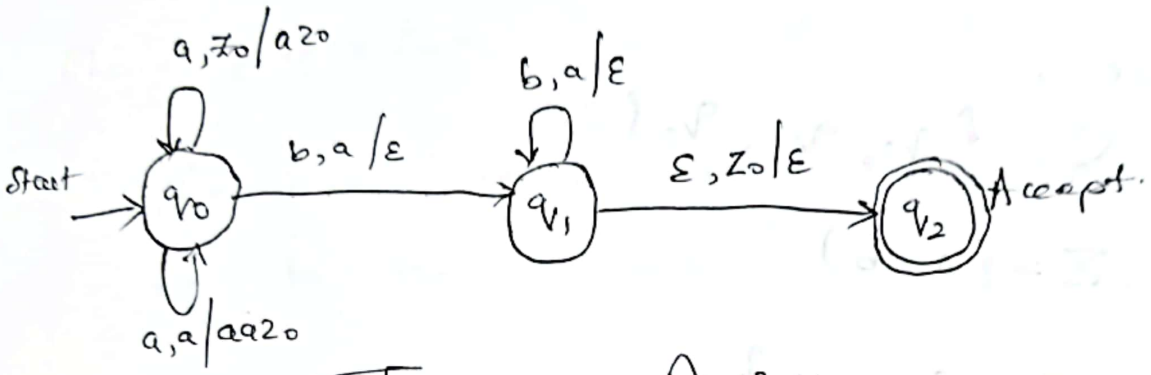
(iii) $\delta(q_0, b, a) = (q_1, \varepsilon)$ } Epsilon specifies pop

(iv) $\delta(q_1, b, a) = (q_1, \varepsilon)$ } Epsilon specifies pop

(v) $\delta(q_1, \varepsilon, Z_0) = (q_2, \varepsilon)$ } Epsilon specifies pop, here we have popped Z_0

Scanned with OKEN Scanner

Step 3:- Next we shall write the transition diagram for the above transition functions,



Transition Diagram

Step 4:- Let us assume a string "aabb", whether it is correctly processed/NOT.

String :- aabbϵ

$\delta(q_0, aabb, Z_0) \vdash$ ~~(q_0, aZ_0)~~ (q_0, abb, aZ_0)

(fallen T) $\vdash$ (q_0, bb, aaZ_0)

$\vdash$ (q_1, b, aZ_0)

$\vdash$ (q_1, ε, Z_0)

$\vdash$ (q_2, ε)

q_2 → Final state; ε → Stack is Empty.

Hence the input string is completely accepted by this PDA.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

Steps:- Let us write / Substitute the 7-tuples of PDA has shown below,

$$A = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$$

Where,

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Gamma = \{a, Z_0\}$

δ = Transition Function.

$q_0 = \{q_0\}$ is the Start / initial State

$Z_0 = \{Z_0\}$ is the top-most symbol on the stack

$F = \{q_2\}$ is the Final State / Accept State.

Note:- * ~~Instantaneous~~ Instaneous Description, is used to describe the Configuration of PDA.

* ID is defined by triples (Q, w, γ) gamma

Where, $Q \rightarrow$ Represents Current State

$w \rightarrow$ Represents string,

$\gamma \rightarrow$ gamma represents Contents of a stack.

Scanned with OKEN Scanner

2) Construct PDA for the language $L = \{a^n b^{2n}\}$ $n \geq 1\}$.

(Ans) Step1:- $L = \{\overset{n=1}{abb}, \overset{n=2}{aabbbb}, \overset{n=3}{aaabbbbbb} \ldots\}$

The procedures involved here are

<i> If we read a, push 2 a's on to the stack.

<ii> If we read b, pop the stack, and switch to new state

<iii> for the 1st pop operation change the state.

<iv> Repeat step <ii> until the stack is empty.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

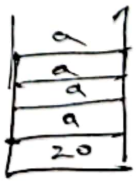
Step2:-

String :- | a | a | b | b | b | b | ∈ |

<i> $\delta(q_0, a, Z_0) = (q_0, aaZ_0) \rightarrow$

~~<ii> $\delta(q_0, a, a) = (q_0, aaZ_0)$~~

~~<iii> $\delta(q_0, b, a$~~

<ii> $\delta(q_0, a, a) = (q_0, aaa) \rightarrow$ 

<iii> $\delta(q_0, b, a) = (q_1, \epsilon) \rightarrow$

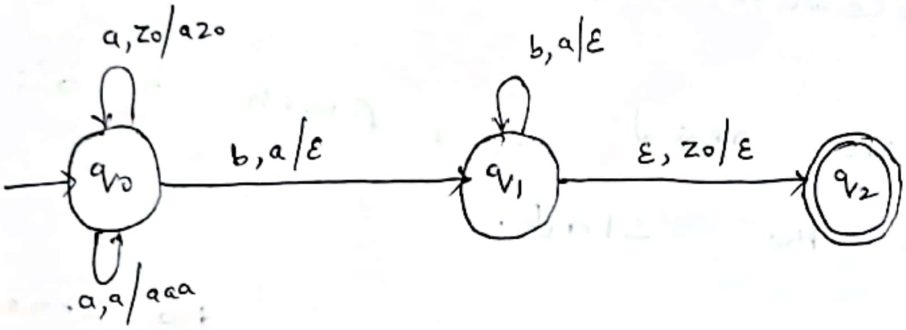
<iv> $\delta(q_1, b, a) = (q_1, \epsilon) \rightarrow$

<v> $\delta(q_1, \epsilon, Z_0) = (q_2, \epsilon) \rightarrow$ Pop Z_0

Scanned with OKEN Scanner

Step3:- Below shows the transition diagram for the above transition function.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841



Transition diagram

Step4:-

Accept string:-

$f(q_0, aabbbb, Z_0) \vdash (q_0, abbbb, aaZ_0)$

$\vdash (q_0, bbbb, aaaaZ_0)$

$\vdash (q_1, bbb, aaaZ_0)$

$\vdash (q_1, bb, aaZ_0)$

$\vdash (q_1, b, aZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

$\vdash (q_2, \varepsilon)$

Final state — Stack empty.

Hence the input string is accepted by PDA.

Scanned with OKEN Scanner

Reject String :-

$\delta(q_0, aabbbbb, z_0) \vdash (q_0, abbbbb, aaz_0)$

$\vdash (q_0, bbbbb, aaaaz_0)$

$\vdash (q_1, bbbb, aaaz_0)$

$\vdash (q_1, bbb, aaz_0)$

$\vdash (q_1, bb, az_0)$

$\vdash (q_1, b, z_0)$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Step 5 :- PDA Components viz

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

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$q_0 = \{q_0\}$

$F = \{q_2\}$

$\delta =$ Transition function of step (i)

$\Gamma = \{\epsilon, a, z_0\}$

$Z_0 = \{z_0\}$

Scanned with OKEN Scanner

(3) Construct PDA for the language $L = \{wcw^R \mid w \in (a,b)^*\}$

(Ans) Step 1:- Let $L = \{ \underset{w\ c\ w^R}{aba\,c\,aba}, \underset{w\ c\ w^R}{abab\,c\,baba} \ldots \}$

Center Marker

* Here we need to push all the symbols on to the stack we get c.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

* Suppose if we encounter the symbol "c", then don't perform any operation i.e keep the stack as it is.

* After Reading the symbol c, next top of the stack symbol is a/b, then pop the elements from the stack

Step 2:- Below shows the transition function for the following string :- | a | b | a | c | a | b | a | ε |

(i) $\delta(q_0, a, Z_0) = (q_0, aZ_0) \rightarrow \boxed{a \mid Z_0}$

~~(ii) $\delta(q_0, b, aZ_0) = (q_0, baZ_0)$~~

(ii) $\delta(q_0, b, Z_0) = (q_0, bZ_0) \rightarrow \boxed{b \mid Z_0}$

4 scenarios

(iii) $\delta(q_0, a, a) = (q_0, aa)$

(iv) $\delta(q_0, a, b) = (q_0, ab)$

(v) $\delta(q_0, b, a) = (q_0, ba)$

(vi) $\delta(q_0, b, b) = (q_0, bb)$

Scanned with OKEN Scanner

(vii) $\delta(q_0, c, a) = (q_1, a)$
(viii) $\delta(q_0, c, b) = (q_1, b)$ } Center Marker and change the state.

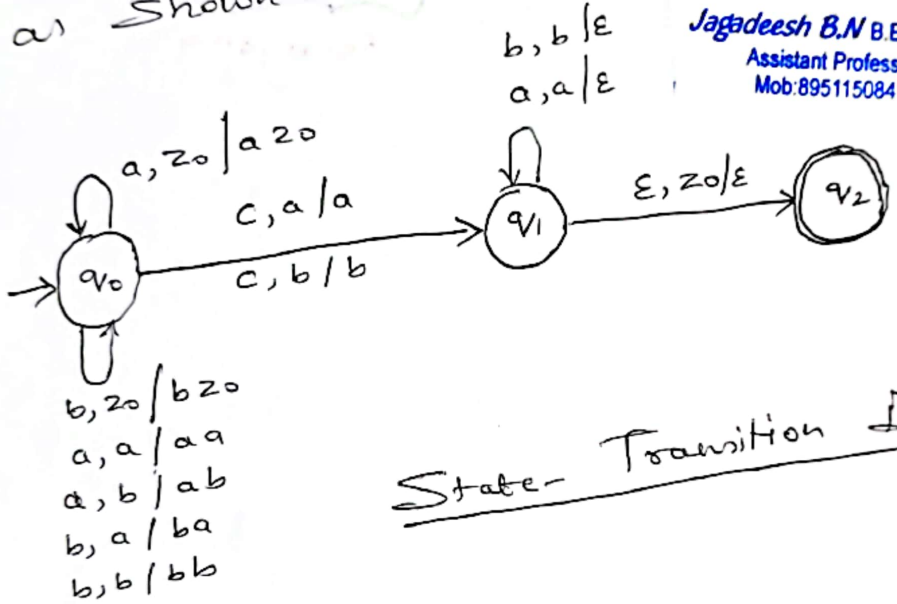
(ix) $\delta(q_1, a, a) = (q_1, \varepsilon)$

(x) $\delta(q_1, b, b) = (q_1, \varepsilon)$

(xi) $\delta(q_1, \varepsilon, z_0) = (q_2, \varepsilon)$ } stack empty by popping z_0.

Step 3:- Below shows the transition diagram as shown below,

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841



State-Transition Diagram

Step 4:- Accept string

$\delta(q_0, abacaba, z_0)$ ~~$\vdash (q_0, abacaba$~~

$\vdash (q_0, bacaba, az_0)$

$\vdash (q_0, acaba, baz_0)$

$\vdash (q_0, caba, abaz_0)$

$\vdash (q_1, aba, abaz_0)$

$\vdash (q_1, ba, baz_0)$

$\vdash (q_1, a, az_0)$

$\vdash (q_1, \varepsilon, z_0)$

$\vdash (q_2, \varepsilon)$

Scanned with OKEN Scanner

Steps:- Reject String

$\delta(q_0, abbcbbb) \vdash (q_0, bbcbbb, z_0)$

$\vdash (q_0, bcbbb, baz_0)$

$\vdash (q_0, cbbb, bbaz_0)$

$\vdash (q_1, bbb, bbaz_0)$

$\vdash (q_1, bb, baz_0)$

$\vdash (q_1, \underbrace{b, az_0}_{\text{Reject string.}})$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

(4) Design PDA for the language $L = \{ww^R \mid w \in (a+b)^*\}$

(Ans) Step1:- Let $L = \{abba, abaaba$

Step2:- Below shows the transition functions,

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

<i> $\delta(q_0, a, Z_0) = (q_0, aZ_0)$
<ii> $\delta(q_0, b, Z_0) = (q_0, bZ_0)$ } stack is empty push a/b

<iii> $\delta(q_0, a, b) = (q_0, ab)$
<iv> $\delta(q_0, b, a) = (q_0, ba)$ } Push either i/p a/b if stack top is a/b

<v> $\delta(q_0, a, a) = (q_0, aa)$
<vi> $\delta(q_0, b, b) = (q_0, bb)$ } Push, Remain in q_0

NPDA

<vii> $\delta(q_0, a, a) = (q_1, \varepsilon)$
<viii> $\delta(q_0, b, b) = (q_1, \varepsilon)$ } Pop, switch to q_1

~~<ix> $\delta(q_1, a, a) = (q_1, \varepsilon)$~~

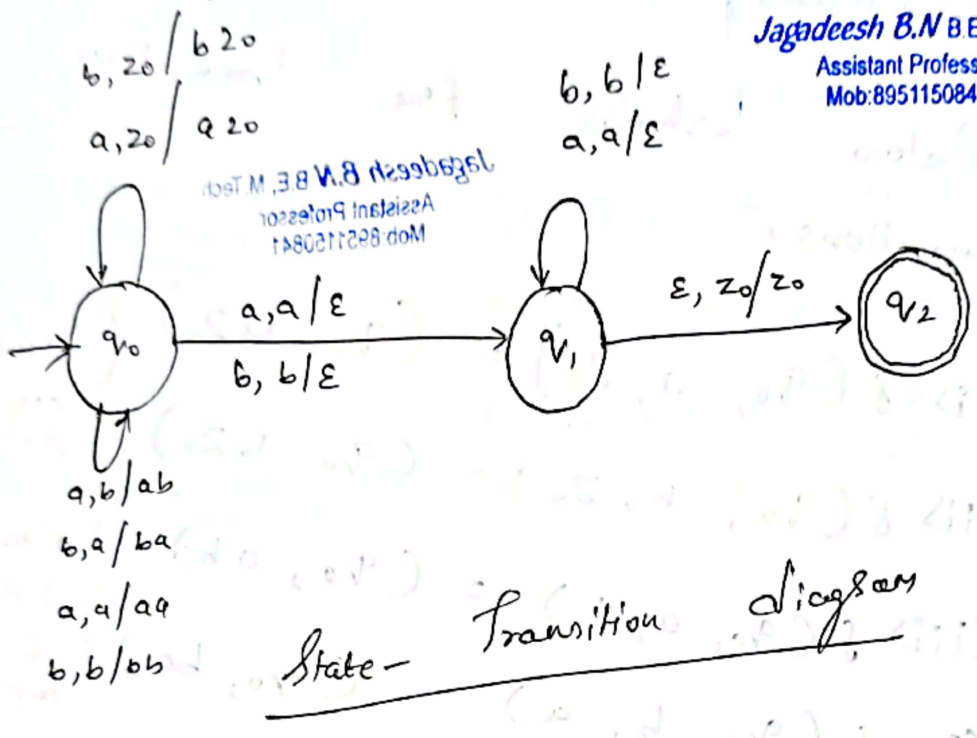
<ix> $\delta(q_1, a, a) = (q_1, \varepsilon)$
<x> $\delta(q_1, b, b) = (q_1, \varepsilon)$ } Pop

<xi> $\delta(q_1, \varepsilon, Z_0) = (q_2, \varepsilon)$ } Pop Z_0

Scanned with OKEN Scanner

Steps:- Below shows the transition diagram:

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841



State-Transition diagram

Scanned with OKEN Scanner

(5) Design PDA for the language $L = \{ w / n_a(w) = n_b(w) \}$

(Ans) Step1:- Let us assume the language

$L = \{\varepsilon, abab, baba, bbaa, aabb \ldots\}$

Step2:- Let us write the transition functions for all the possibilities as shown below,

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

(i) $\delta(q_0, a, z_0) = (q_0, az_0)$

(ii) $\delta(q_0, b, z_0) = (q_0, bz_0)$

(iii) $\delta(q_0, a, a) = (q_0, aa)$

(iv) $\delta(q_0, b, b) = (q_0, bb)$

~~(v) $\delta(q_0, a, b) = (q_0, ab)$~~ (v) $\delta(q_0, a, b) = (q_0, ab)$

~~(vi) $\delta(q_0, b, a) = (q_0, ba)$~~ (vi) $\delta(q_0, b, a) = (q_0, ba)$

(vii) $\delta(q_0, a, b) = (q_1, \varepsilon)$

(viii) $\delta(q_0, b, a) = (q_1, \varepsilon)$

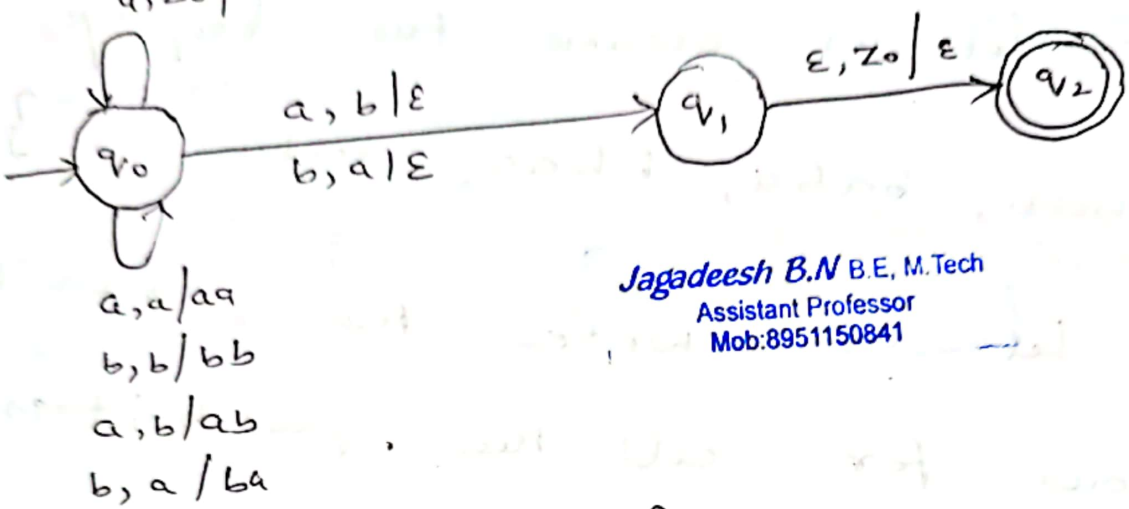
~~(ix) $\delta(q_1, a, b) = (q_1, \varepsilon)$~~

~~(x) $\delta(q_1, b, a) = (q_1, \varepsilon)$~~

(xi) $\delta(q_1, \varepsilon, z_0) = (q_2, \varepsilon)$

Scanned with OKEN Scanner

Step3:- Below shows the transition diagram.



Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Transition Diagram.

(6) Design PDA for the language Balanced Parenthesis

(Ans) Step1:- Usually there are 3 symbols available in the Parenthesis viz,

( ) → Left & Right Parenthesis.

{ } → Left & Right Curley Braces.

[ ] → Left & Right Square Braces.

Step2:- Below shows the transition function.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

<i> $\delta(q_0, (, Z_0) = (q_0, (Z_0)$

<ii> $\delta(q_0, \{, Z_0) = (q_0, \{Z_0)$

<iii> $\delta(q_0, [, Z_0) = (q_0, [Z_0)$

<iv> $\delta(q_0, (, () = (q_0, ((Z_0)$

<v> $\delta(q_0, \{, \{) = (q_0, \{\{Z_0)$

<vi> $\delta(q_0, [, [) = (q_0, [[Z_0)$

<vii> $\delta(q_0, (, [) = (q_0, ([Z_0)$

<viii> $\delta(q_0, [, () = (q_0, [(Z_0)$

<ix> $\delta(q_0, \{, () = (q_0, \{(Z_0)$

<x> $\delta(q_0, (, \{) = (q_0, (\{Z_0)$

<xi> $\delta(q_0, [, \{) = (q_0, [\{Z_0)$

<xii> $\delta(q_0, \{, [) = (q_0, \{[Z_0)$

Scanned with OKEN Scanner

(xiii) $\delta(q_0, \}, \{) = (q_1, \varepsilon)$

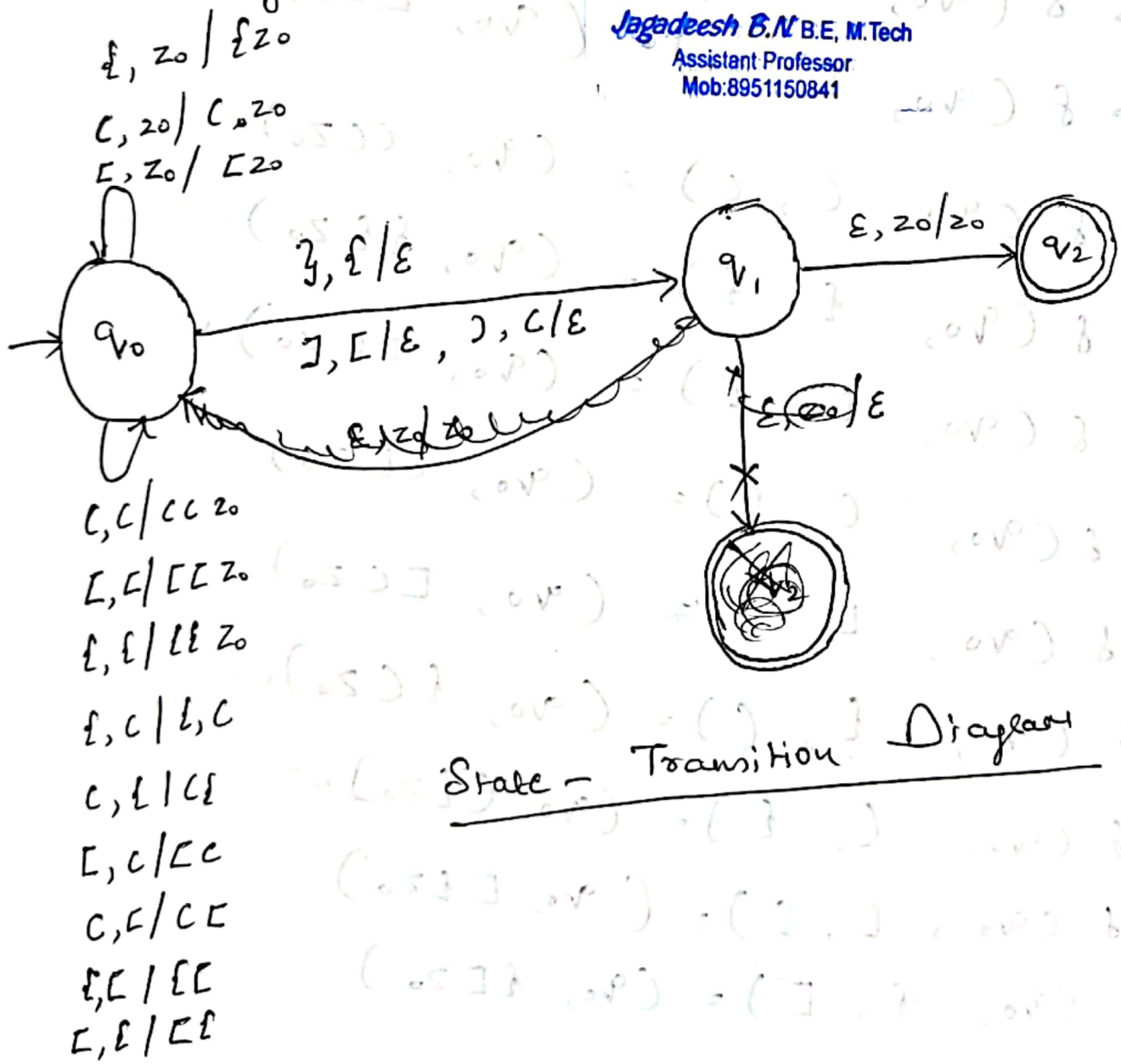
(xiv) $\delta(q_0,), ( ) = (q_1, \varepsilon)$

(xv) $\delta(q_0,], [) = (q_1, \varepsilon)$

(xvi) $\delta(q_1, \varepsilon, z_0) = (q_2, \varepsilon)$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Step 3:- Below shows the transition diagram.



State - Transition Diagram

(7) Design PDA for the language $L = \{ n_a(w) > n_b(w)$ where $w \in (a+b)^* \}$

(Ans) Step 1:- Let $L = \{ aab, baaba, ababaa \ldots \}$

Step 2:- * Here if the input symbol is a & stack topmost element is b

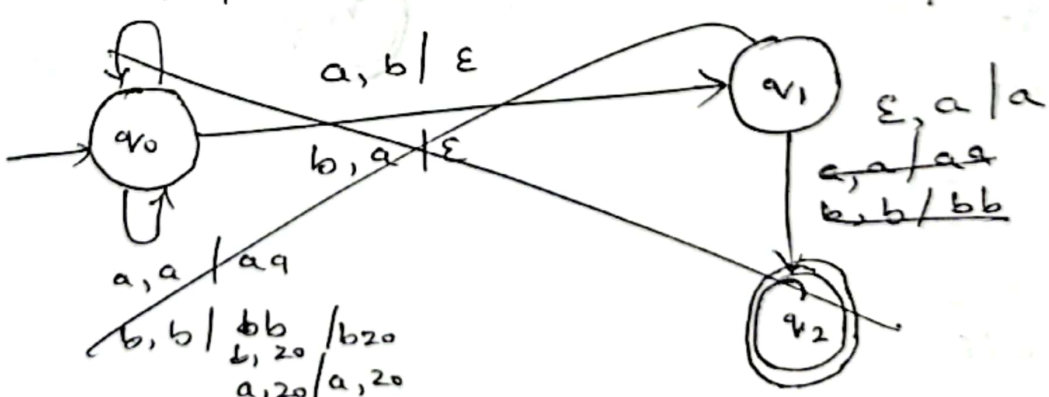
(or) if the input symbol is b & stack topmost element is a, perform pop operation.

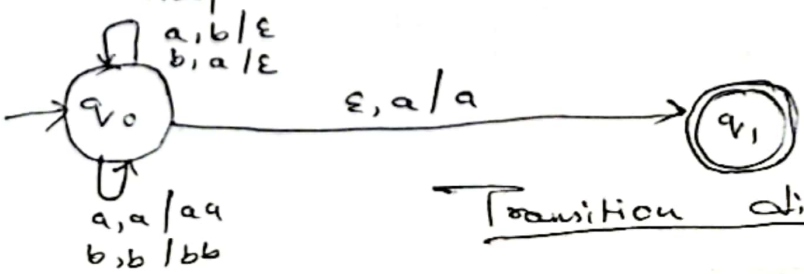
* Secondly if the i/p symbol is a & stack topmost element is also a

(or) if the i/p symbol is b & stack topmost symbol is also b, then push the element on to the stack.

Step 3:- Below shows the transition function.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841





Transition diagram

Scanned with OKEN Scanner

(8) Design PDA for the language $L = \{ n_a(w) < n_b(w) \mid w \in (a+b)^* \}$

(Ans) Step 1:- Consider a language

$L = \{ abb, babba, \ldots \}$

Step 2:-

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

$\delta(q_0, b, b) = (q_0, bb)$

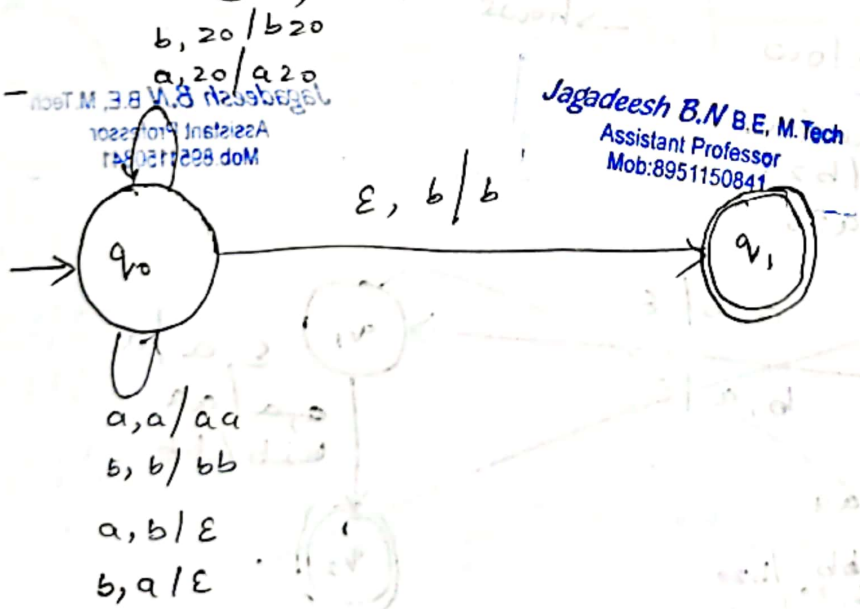
$\delta(q_0, a, b) = (q_0, \varepsilon)$

$\delta(q_0, b, a) = (q_0, \varepsilon)$

$\delta(q_0, \varepsilon, b) = (q_1, b)$

ab
aab

Step 3:-



Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Transition diagram

Scanned with OKEN Scanner

Syntax Analysis Phase of Compilers : Part 2

Bottom-up Parsing

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

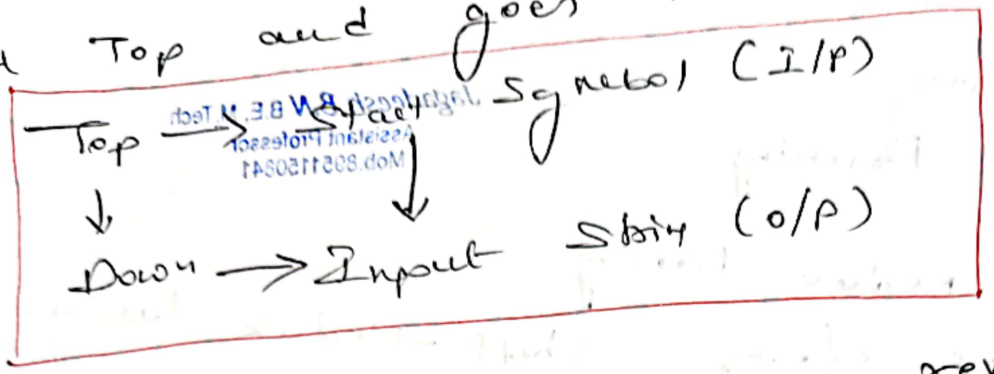
(i) Reductions
(ii) Handle Pruning
(iii) Shift-Reduce Parsing
(iv) Conflicts during Shift-Reduce Parsing.

What is Bottom-Up Parsing?

* Parser is also called as Syntax Analyzer.
* Parser accepts the tokens as input and checks for the correctness of the token.
* If the token syntax checked by the parser is correct, then parser generates Parse tree.
* Suppose if the token syntax checked by the parser is incorrect, then parser generates error message and reports it to the user.
* Parsing Techniques are of 2 types viz
  Top-Down Parsing technique.
  Bottom-Up Parsing technique.

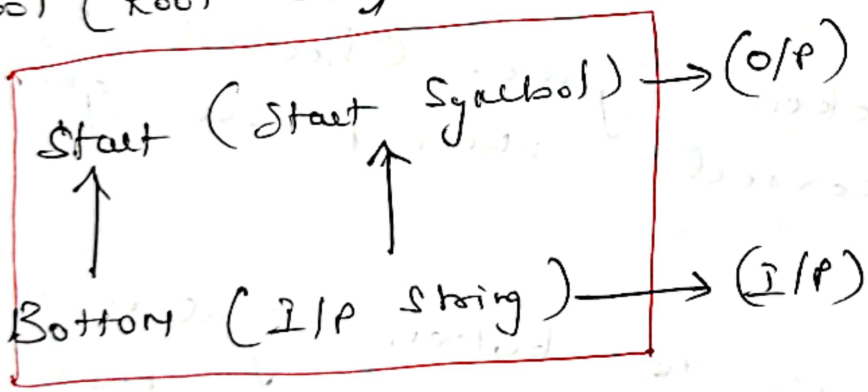
Scanned with OKEN Scanner

* In Top-Down parsing, the parser will construct parse tree begining from Top and goes towards down.



* In Bottom-up parsing is reverse to the Top-Down parsing, parsing will be generated from the Bottom (leaf node/ Input String) and goes towards start Symbol (Root node)

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841



Scanned with OKEN Scanner

(i) Reductions

(1) $S \rightarrow aABe$

$A \rightarrow Abc \mid b$

$B \rightarrow d$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Input String :- "abbcde".

(Ans) Step1:- * Here we have to take a substring, and then check whether that substring matches to the production at the right side.

* If the substring matches, then substitute that substring with the corresponding left side of the production (Non-Terminal)

Step2:-

String :- abbcde → Input

(Called as Handle, and this process is called as Reduction)

abbcde (A→b)

aAbcde (A→Abc)

aAde (B→d)

aABe (S→aABe)

S → output

This process is said to be Bottom-Up Parsing

Scanned with OKEN Scanner

Step 3:- In the above Step 2, the one inside the bracket is said to be a right Most derivation in reverse order.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

S

$S \to aABe$

$S \to aAde$

$S \to aAbcde$

$S \to aabcde$

} Right-Most Derivation in Reverse Order (Bottom-to-top)

(2) $E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid id$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Input string :- id*id

(Ans) Step 1 :-

id*id (F→id) → Input

F*id (T→F)

~~T*id (E→T)~~

T*id (F→id)

T*F (T→T*F)

T (E→T)

E → Output

Scanned with OKEN Scanner

(ii) Handle and Handle Pruning:-

* Handle means a substring which matches to the right side of the production, if it matches, then it is replaced with a corresponding left-side of the production.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Example:- $S \rightarrow aABe$

$A \rightarrow Abc \mid b$

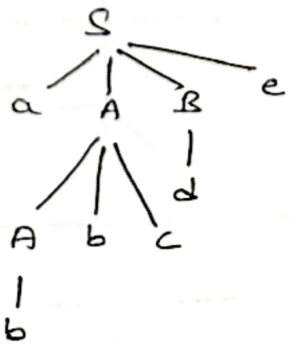
$B \rightarrow d$

String:- abbcde

Step 1:- (Ans)

Right Sentential Form	Handle	Reducing Production
a b b c d e	b	$A \rightarrow b$
a A b c d e	A b c	$A \rightarrow Abc$
a A d e	B d	$B \rightarrow d$
a A B e	a A B e	$S \rightarrow aABe$
S		

Step 2:- Bottom-up Parser generates Right most derivation in reverse order.



RMD Parse tree

Scanned with OKEN Scanner

RMD

$S \to aABe$
$S \to aAde$
$S \to aAbcde$
$S \to abbcde$

Reverse of Reduction production in the previous table column generates RMD.

* Handle Pruning Means produces the Right Most Derivation in Reverse Order.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

* "Handle" Means its a Substring which matches with the right side of the production.

Example :

$E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid id$

String : id*id

(Ans)

Right ~~Sential~~ Sentential Form	Handle	Reducing Production
id*id	id	$F \to id$
F*id	F	$T \to F$
T*id	id	$F \to id$
T*F	T*F	$T \to T*F$
T	T	$E \to T$
E		
Handle Pruning		

Obtaining Right Most Derivation in Reverse Order is said to be

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

* The process of obtaining Right Most derivation in reverse order is called <u>Handle Pruning.</u>

* E

E -> T

E -> T * F

E -> T * id

E -> F * id

E -> id * id

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

Continuation of PDA problems.

(1) Design a PDA for the language $L = \{ a^n b^n \mid n \geq 0 \}$

(Ans) Step 1:- The language $L = \{ \varepsilon, ab, aabb, aaabbb \ldots \}$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Step 2:- Below shows the transition function for the PDA.

$$\delta(q_0, a, Z_0) = (q_1, aZ_0)$$

$$\delta(q_1, a, a) = (q_1, aa)$$

$$\delta(q_1, b, a) = (q_2, \varepsilon)$$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

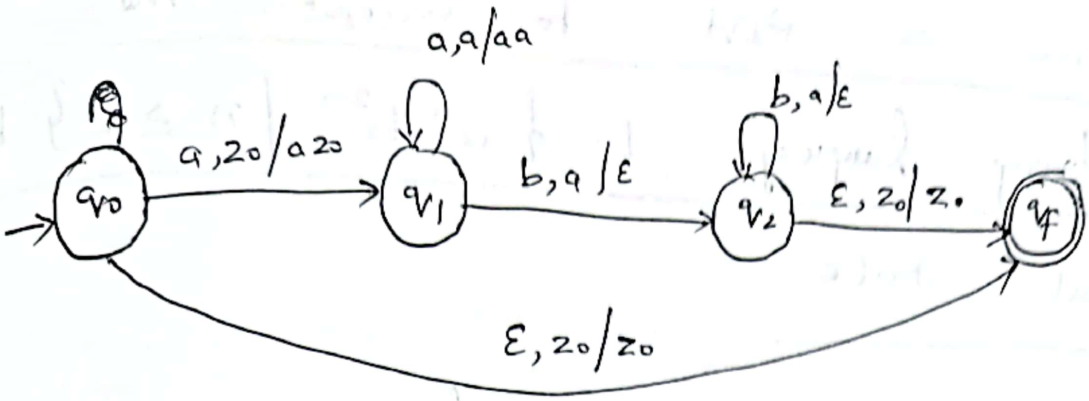
$$\delta(q_2, b, a) = (q_2, \varepsilon)$$

$$\delta(q_2, \varepsilon, Z_0) = (q_f, Z_0)$$

$$\delta(q_0, \varepsilon, Z_0) = (q_f, Z_0)$$

Step 3:- Below shows the Transition diagram for the above step 2.

Scanned with OKEN Scanner



Transition diagram.

Step 4:- Components of PDA i.e

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

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Where,

$Q = \{q_0, q_1, q_2, q_f\}$

$\Sigma = \{a, b\}$

$\delta =$ Transition function of Step2.

$q_0 = \{q_0\}$ start state / Initial state

$F = \{q_f\}$ is an accepting state / final state.

$Z_0 = \{Z_0\}$ Initial symbol on to the stack.

$\Gamma = \{a, z_0\}$

Step 5:- Instantaneous Description (Q, w, γ)

$(\cancel{q_0, w}, \; (q_0, aaabbb, Z_0) \vdash (q_1, aabbb, aZ_0)$

$\vdash (q_1, abbb, aaZ_0)$

$\vdash (q_1, bbb, aaaZ_0)$

$\vdash (q_2, bb, aaZ_0)$

$\vdash (q_2, b, aZ_0)$

$\vdash (q_f, \varepsilon, Z_0)$

Scanned with OKEN Scanner

(2) Give a PDA to accept the following language $L = \{ a^n b^{2n} \mid n \geq 1 \}$ by final state

(Ans) Step 1:- Let the language

$L = \{ abb, aabbbb, aaabbbbbb \ldots \}$

Step 2:- The transition function for the above PDA is written by,

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

$\delta(q_0, a, Z_0) = (q_1, aZ_0)$

$\delta(q_0, a, a) = (q_1, aa)$

$\delta(q_1, b, a) = (q_2, a)$

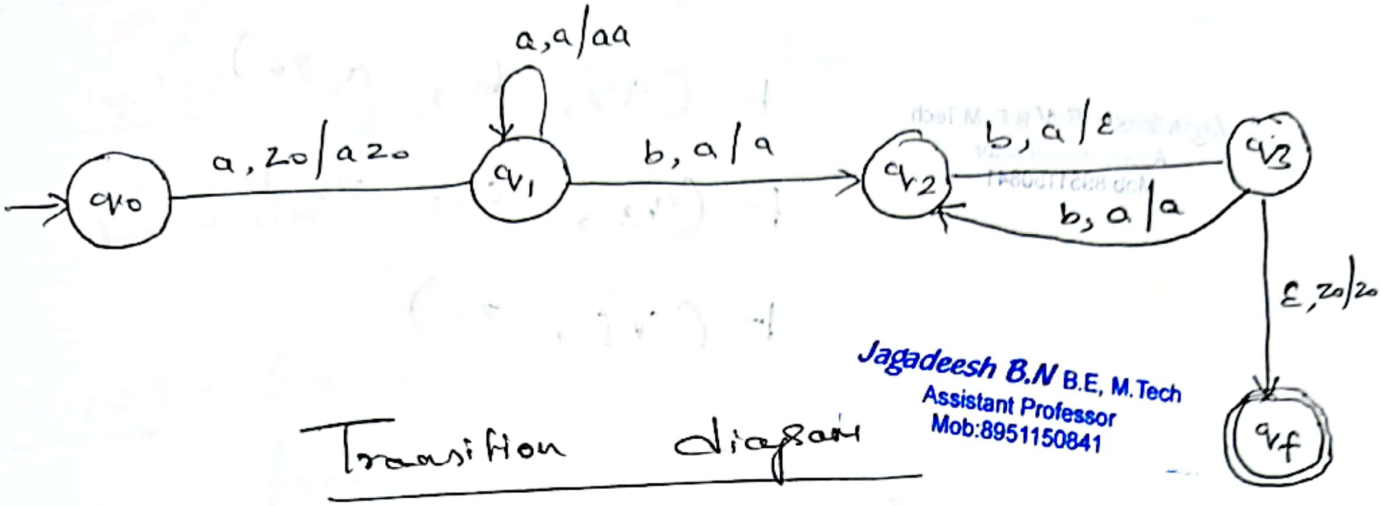
$\delta(q_2, b, a) = (q_3, \varepsilon)$

$\delta(q_3, b, a) = (q_2, a)$

$\delta(q_3, \varepsilon, Z_0) = (q_f, Z_0)$

Scanned with OKEN Scanner

Step3:- Below shows the transition diagram



Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Transition diagram

Step4:- Components of PDA

$A = (Q, \Sigma, \delta, q_0, F, Z_0, I)$

Where.

$Q = \{q_0, q_1, q_2, q_3, q_f\}$

$\Sigma = \{a, b\}$

$\delta = \{$ Transition Function of Step2 $\}$

$q_0 = \{q_0\}$ Start / Initial State.

$F = \{q_f\}$ Final / Accept State.

$Z_0 = \{Z_0\}$ Initial symbol on to the stack.

$I = \{a, Z_0\}$

Step5:- Instantaneous Description

Scanned with OKEN Scanner

$$\delta(q_0, abb, Z_0) \vdash (q_1, bb, aZ_0)$$
$$\vdash (q_2, bb, aZ_0)$$
$$\vdash (q_2, b, aZ_0)$$
$$\vdash (q_3, \varepsilon, Z_0)$$
$$\vdash (q_f, Z_0)$$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

(3) Construct PDA for the language

$L = \{ a^n b^m c^n \mid n \geq 1, m \geq 1 \}$

(Ans) Step 1:- Consider the language

$L = \{ aabbbcc \ldots \}$

Step 2:- Below shows the transition function.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

$\delta(q_0, a, z_0) = (q_1, a z_0)$

$\delta(q_1, a, a) = (q_1, aa)$

$\delta(q_1, b, a) = (q_2, a)$

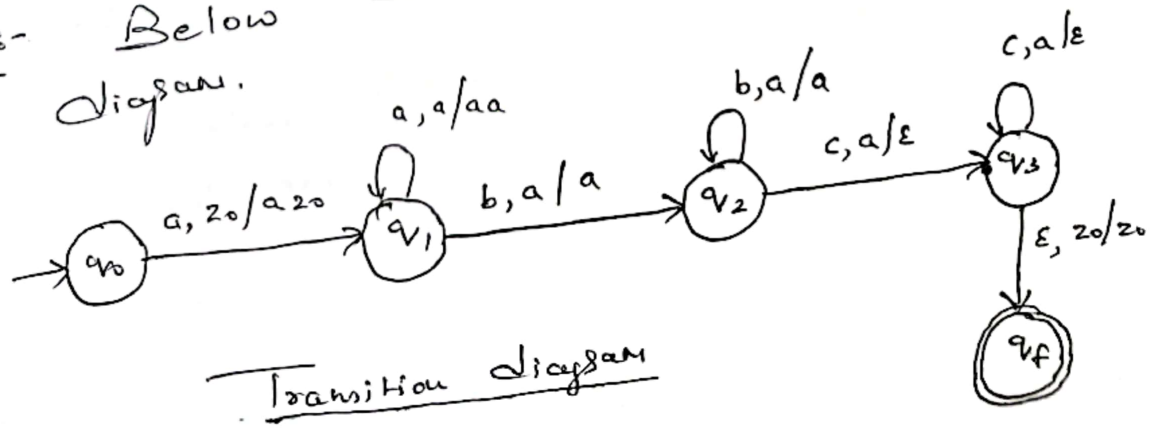
$\delta(q_2, b, a) = (q_2, a)$

$\delta(q_2, c, a) = (q_3, \varepsilon)$

$\delta(q_3, c, a) = (q_3, \varepsilon)$

$\delta(q_3, \varepsilon, z_0) = (q_f, z_0)$

Step 3:- Below shows the transition diagram.



Transition diagram

Scanned with OKEN Scanner

Step4:- Instantaneous Description.

~~$(q_0, aabbbcc) \vdash (q_1, abbb$~~

$(q_0, aabbbcc, z_0) \vdash (q_1, abbbcc, az_0)$

$\vdash (q_1, bbbcc, aaz_0)$

$\vdash (q_2, bbcc, aaz_0)$

$\vdash (q_2, bcc, aaz_0)$

$\vdash (q_2, cc, aaz_0)$

$\vdash (q_3, c, az_0)$

$\vdash (q_3, \varepsilon, z_0)$

$\vdash (q_f, z_0)$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

(4) Obtain PDA to accept the language $L(M) = \{ wcw^R \mid w \in (a+b)^* \}$ where w^R is reverse of w by a final state.

(Ans) Step 1:- Let Language $L = \{ abacaba, baacaab \ldots \}$

Step 2:- Obtain the transition function for the following :-

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

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

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, c, a) = (q_1, a)$

$\delta(q_0, c, b) = (q_1, b)$

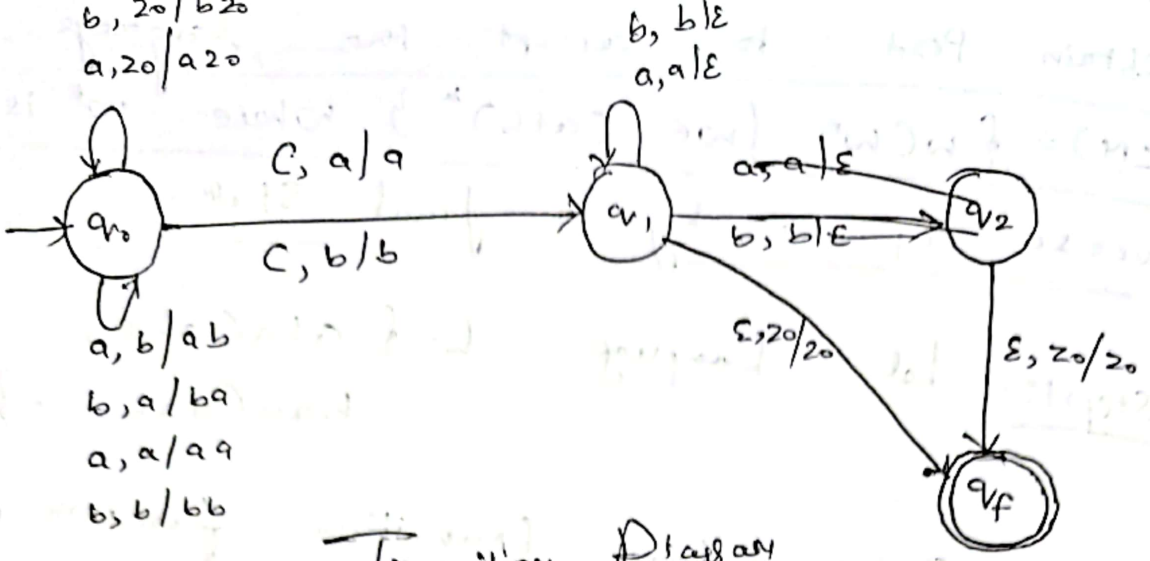
$\delta(q_1, a, a) = (q_f, \varepsilon)$

$\delta(q_1, b, b) = (q_f, \varepsilon)$

$\delta(q_f, \varepsilon, z_0) = (q_f, z_0)$

Step 3:- Below shows the transition diagram.

Scanned with OKEN Scanner



Tramition Diagram

Step1:- Instantaneous Description.

$\delta(q_0, aba(aba, Z_0) \vdash (q_0, ba(aba, aZ_0)$

$\vdash (q_0, a(aba, baZ_0)$

$\vdash (q_0, (aba, abaZ_0)$

$\vdash (q_1, aba, abaZ_0)$

$\vdash (q_1, ba, baZ_0)$

$\vdash (q_1, a, aZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

$\vdash (q_2, Z_0)$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

(5) Obtain a PDA for the language $L = \{ w \mid w \in (a+b)^* \text{ and } n_a(w) = n_b(w) \}$ by a final state

(Ans) Step1:- Let the language $L = \{aabb, abab, baba, bbaa \ldots\}$

Step2:- Transition Function for the above language L is,

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, z_0) = (q_0, bz_0)$

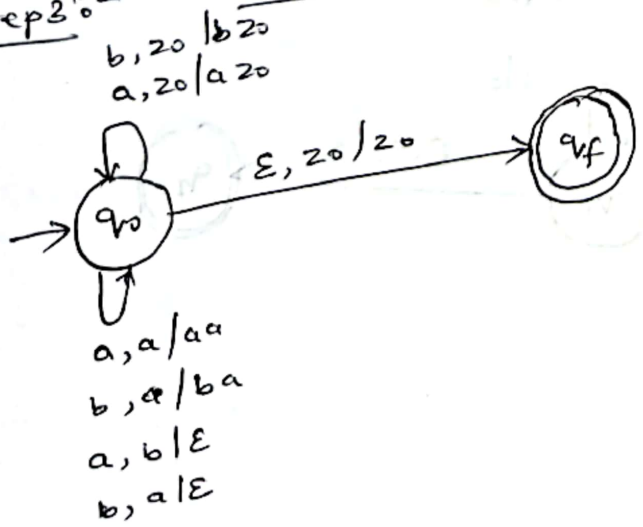
$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, a, b) = (q_0, \varepsilon)$

$\delta(q_0, b, a) = (q_0, \varepsilon)$

$\delta(q_0, \varepsilon, z_0) = (q_f, z_0)$

Step3:- Transition diagram



Scanned with OKEN Scanner

Another Solution

Obtain PDA for the language

$L = \{ a^n b^{2n} \mid n \geq 1 \}$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

(Ans) **Step 1:-** Consider a string for the language

$L = \{abb, aabbbb, aaabbbbbb \ldots\}$

Step 2:- Transition function

$\delta(q_0, a, z_0) = (q_0, aaz_0)$

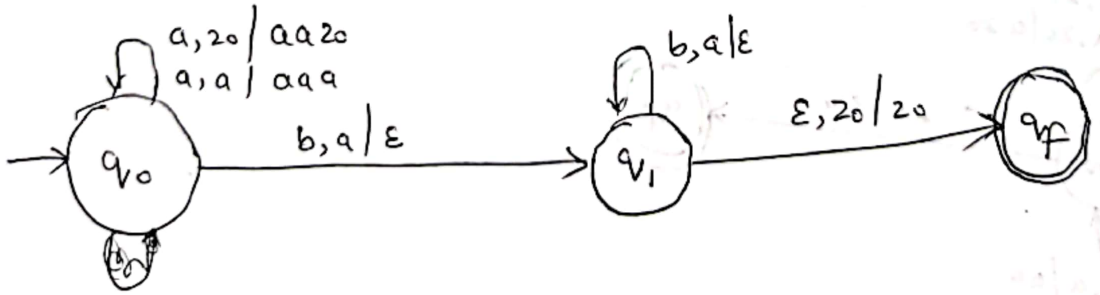
$\delta(q_0, a, a) = (q_0, aaa)$

$\delta(q_0, b, a) = (q_1, \varepsilon)$

$\delta(q_1, b, a) = (q_1, \varepsilon)$

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

Step 3:- Transition diagram



Scanned with OKEN Scanner

(6) Obtain PDA for the language $L = \{ ww^R \mid w \in (a+b)^* \}$ by a final state

(Ans) Step1:- Let us consider a string for the language $L = \{ abaaba, baaaab, babaabab \}$

Step2:- Transition Function

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

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

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

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, a, a) = (q_1, \varepsilon)$

$\delta(q_0, b, b) = (q_1, \varepsilon)$

$\delta(q_0, \varepsilon, z_0) = (q_f, z_0)$

$\delta(q_1, a, a) = (q_1, \varepsilon)$

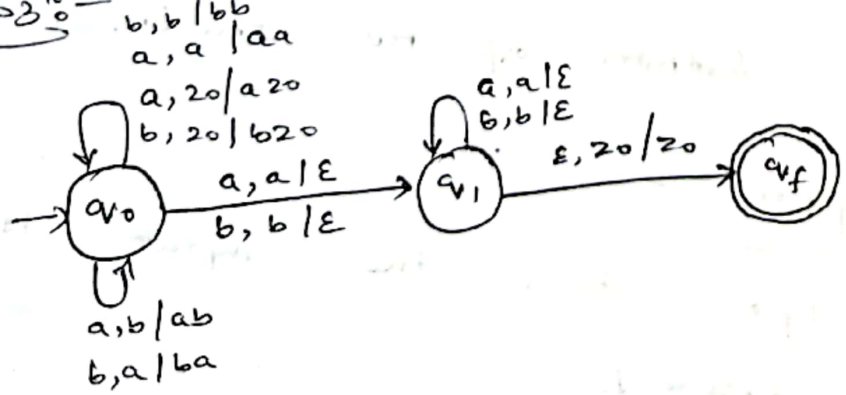
$\delta(q_1, b, b) = (q_1, \varepsilon)$

~~Step3:- Transition diagram~~

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, b) = (q_0, bb)$

Step3:- Transition diagram



Scanned with OKEN Scanner

Continuation of Bottom-Up Parsing

Shift Reduce Parsing

* Shift-Reduce Parser is a Bottom-Up parsing technique.
* Bottom-Up Parsing Means Constructs a parse tree beginning at the input string which is a leaf nodes and goes towards Up i.e the Root Node which is the start symbol of the grammar.
* In Shift Reduce parser we mainly use 2 Data-Structures viz
  (i) stack
  (ii) Input Buffer.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

(i) Stack :-
* Stack stores Symbols of the grammar.
* Initially the stack contains $, which specifies the bottom of the stack.

(ii) Input Buffer :-
* Input Buffer stores the input string that to be parsed.
* End of the input buffer in a string is indicated by a $.

Scanned with OKEN Scanner

* Here the Symbol $ specifies the bottom of the stack and end of the input string.

Actions of Shift-Reduce Parser

* There are 4 - Actions that can be performed on to the Shift-Reduce parser.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

(i) Shift
(ii) Reduce
(iii) Accept
(iv) Error

} 4 - Actions.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

(i) Shift :-

* Shift Means, it is a PUSH operations, which shifts the i/p symbol on to the top of the stack.
* Initially top of the stack contains $, on top of the $, the i/p symbol is pushed.

(ii) Reduce :-

* If top of the stack matches with the right side of the production, then it is reduced to the corresponding left ~~side~~ hand side Non-Terminal.

Scanned with OKEN Scanner

<iii> Accept:-

* After Parsing the entire input string, the top of the stack contains $ and start symbol S, then the corresponding input string is accepted by the parser.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

<iv> Error:-

* If there is an syntax errors in the grammar (or) if the input string does not belong to the language of the grammar, then the parser generates the error messages.

Scanned with OKEN Scanner

Example 1 :-

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

<D> $E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

String is generated is id * id

(Ans) Step1:- Divide into 3 columns

STACK Operation	INPUT BUFFER	ACTION.
$	id*id $	Shift
$id	*id $	Reduce by F->id
$F	*id $	Reduce T->F
$T	*id $	Shift
$T*	id $	Shift
$T*id	$	Reduce F->id
$T*F	$	Reduce T->T*F
$T	$	Reduce E->T
$E	$	Accepted by the grammar.

Conflicts

Shift-Reduce

Reduce Reduce Conflicts

Scanned with OKEN Scanner

Example 2:-

Consider the following grammar

$S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Parse the input string (a,(a,a)) using Shift Reduce Parser

(for) Step 1:- Divide into 3 columns.

Stack operation	Input Buffer	Actions
$	(a,(a,a))$	Shift
$(	a,(a,a))$	Shift a
$(a	,(a,a))$	Reduce S->a
$(S	,(a,a))$	Reduce L->S
$(L	,(a,a))$	Shift ,
$(L,	(a,a))$	Shift (
$(L,(	a,a))$	Shift a
$(L,(a	,a))$	Reduce S->a
$(L,(S	,a))$	Reduce L->S
$(L,(L	,a))$	Shift ,

Scanned with OKEN Scanner

$ (L, (L,	a))$	shift a
$ (L, (L, a	))$	Reduce S→a
$ (L, (L, S	))$	Reduce L→L,S
$ (L, (L	))$	shift)
$ (L, (L)	)$	Reduce S→(L)
$ (L, S	)$	Reduce L→L,S
$ (L	)$	shift)
$ (L)	$	~~shift~~ Reduce S→(L)
$ S	$	Accepted by the parser.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

(3) Consider the grammar

$S \to S+S$

$S \to S-S$

$S \to (S)$

$S \to a$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Input string : a - (a+a)

Stack operation	Input Buffer	Action
$	a - (a+a) $	Shift a
$ a	- (a+a) $	Reduce by S→a
$ S	- (a+a) $	Shift -
$ S -	(a+a) $	Shift (
$ S - (	a+a) $	Shift a
$ S - (a	+a) $	Reduce by S→a
$ S - (S	+a) $	Shift +
$ S - (S +	a) $	Shift a
$ S - (S + a	) $	Reduce by S→a
$ S - (S + S	) $	Reduce by S→S+S
$ S - (S	) $	Shift)
$ S - (S)	$	Reduce by S→(S)
$ S - S	$	Reduce S→S-S
$ S	$	Accepted by the String

Scanned with OKEN Scanner

(4) Consider the grammar.

$E \rightarrow 2E2$

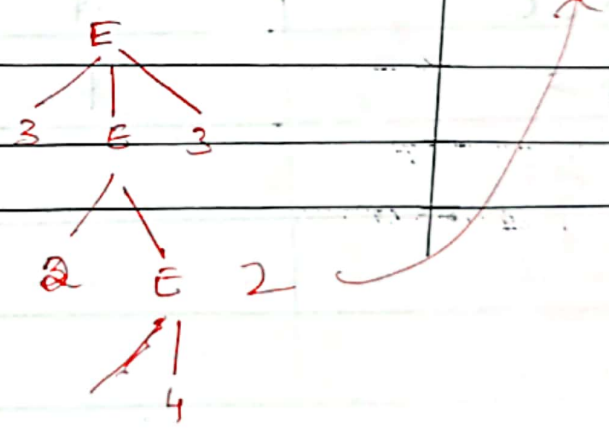
$E \rightarrow 3E3$

$E \rightarrow 4$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Input string 32423

Stack Operation	Input Buffer	Action
$	32423 $	Shift 3
$3	2423 $	Shift 2
$32	423 $	Shift 4
$324	23 $	Reduce by E→4
$32E	23 $	Shift 2
$32E2	3 $	Reduce E→2E2
$3E	3 $	Shift 3
$3E3	$	Reduce by E→3E3
$E	$	Accepted by the Parser.



Parse Tree

Scanned with OKEN Scanner

(4) Consider the grammar

$S \rightarrow CC$

$C \rightarrow cC \mid d$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Perform Shift Reduce Parsing for the input string w = cdcd

Stack Operation	Input Buffer	Action
$	cdcd$	Shift c
$c	dcd$	Shift d
$cd	cd$	Reduce C→d
$cC	cd$	Reduce C→cC
$C	cd$	shift c
$Cc	d$	shift d
$Ccd	$	Reduce C→d
$CcC	$	Reduce C→cC
$CC	$	Reduce S→CC
$S	$	Accepted by the parser

Scanned with OKEN Scanner

(5) Construct Shift Reduce parser for the given grammar and Explain the Conflicts in Shift-Reduce-Parsing

$S \rightarrow aABe$

$A \rightarrow Abc \mid b$

$B \rightarrow d$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

String "abbcde"

(Ans)

Stack Operation	Input Buffer	Action
$	abbcde $	Shift a
$a	bbcde $	Shift b
$ab	bcde $	Reduce A->b
$aA	bcde $	Shift b
$aAb	cde $	Shift c
$aAbc	de $	Reduce A->Abc
$aA	de $	Shift d
$aAd	e $	Reduce B->d
$aAB	e $	Shift e
$aABe	$	Reduce S->aABe
$S	$	Accepted by the parser.

Scanned with OKEN Scanner

(6) Consider the following grammar

$S \rightarrow TL$

$T \rightarrow int \mid float$

$L \rightarrow L, id \mid id$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Parse the input string int id, id; using Shift-Reduce parser.

(7) Consider string "10201", design Shift-Reduce parser for the following grammar,

$S \rightarrow 0S0 \mid 1S1 \mid 2$

Scanned with OKEN Scanner

Introduction to LR Parsing

* LR means L stands for scanning the input symbol from left-to-Right and R stands for obtaining RMD in reverse order.

* LR parser is an efficient bottom-up parsing technique that can be used to parse large classes of Context free grammar called LR(0) parsing.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

* L → stands for scanning symbols from left-to-Right

R → stands for RMD in reverse.

0 → stands for no of i/p symbols of lookahead.

* Types of parsing methods are

(i) SLR

(ii) CLR

(iii) LALR

Scanned with OKEN Scanner

SLR

* SLR stands for Simple LR.

* SLR is the smallest class of grammar having few number of states.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

* LR(0) and SLR is similar, the only difference is that, in LR(0) parsing, there is a chance of Shift-Reduce conflicts, to overcome from this drawback we go for SLR(1).

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

(1) Construct SLR parsing table for the following grammar.

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Also parse the input string id* id + id.

(Ans) Step 1:- Write the Augmented grammar i.e, for the above grammar.

$E' \rightarrow E$

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

→ ~~Augmented~~ Grammar.

$E' \rightarrow E$

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow id$

⇒ Augmented Grammar.

Scanned with OKEN Scanner

Step 2:- Calculate canonical collection of LR(0) items.

$I_0 \rightarrow$

0 $E' \rightarrow .E$

1 $E \rightarrow .E+T$

2 $E \rightarrow .T$

3 $T \rightarrow .T*F$

4 $T \rightarrow .F$

5 $F \rightarrow .(E)$

6 $F \rightarrow .id$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

goto (I_0, E)

I_1 : $E' \rightarrow E.$

$E \rightarrow E.+T$

goto (I_0, T)

I_2 : $E \rightarrow T.$

$T \rightarrow T.*F$

goto (I_0, F)

I_3 : $T \rightarrow F.$

goto (I_0, ( )

I_4 : $F \rightarrow (.E)$

$E \rightarrow .E+T$

$E \rightarrow .T$

$T \rightarrow .T*F$

$T \rightarrow .F$

$F \rightarrow .(E)$

$F \rightarrow .id$

goto (I_0, id)

I_5 : $F \rightarrow id.$

Scanned with OKEN Scanner

I_6: goto (I_1, +)

$E \to E + \cdot T$

$T \to \cdot T * F$

$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot id$

I_7: goto (I_2, *)

$T \to T * \cdot F$

$F \to \cdot (E)$

$F \to \cdot id$

I_8: goto (I_4, E)

$F \to (E \cdot)$

$E \to E \cdot + T$

goto (I_4, T)

I_2: $\{ E \to T \cdot$, $T \to T \cdot * F$

goto (I_4, F)

I_3 $\{ T \to F \cdot$

goto (I_4, (

I_4 $\{ F \to (\cdot E)$, $E \to \cdot E + T$, $E \to \cdot T$, $T \to \cdot T * F$, $T \to \cdot F$, $F \to \cdot (E)$ $\}$

goto (I_4, id)

I_5 $\{ F \to id \cdot$

goto (I_6, T)

I_9: $E \to E + T \cdot$

$T \to T \cdot * F$

goto (I_6, F)

I_3 $\{ T \to F \cdot$

goto (I_6, (

I_4 $\{ F \to (\cdot E)$

goto (I_6, id)

I_5 $\{ F \to id \cdot$

goto (I_7, F)

I_{10}: $T \to T * F \cdot$

goto (I_7, (

I_4 $\{ F \to (\cdot E)$

goto (I_7, id)

I_5 $\{ F \to id \cdot$

goto (I_8,))

I_{11}: $F \to (E) \cdot$

goto (I_8, +)

I_6 $\{ E \to E + \cdot T$, $T \to \cdot T * F$ $\}$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

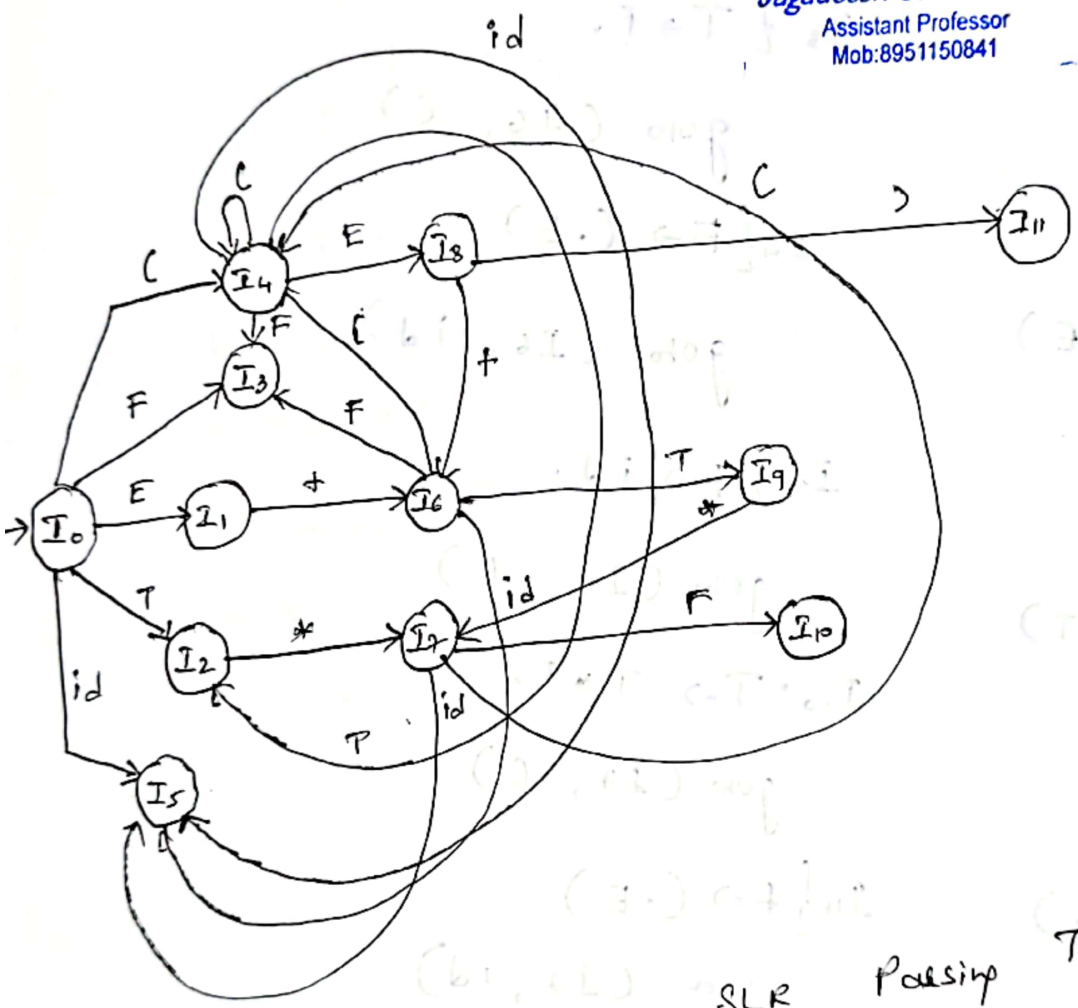
goto(I_9, *)

$$I_7 \begin{cases} T \to T * . F \\ F \to . (E) \\ F \to . id \end{cases}$$

goto

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Step 2:- Designing of Finite Automata



Step 3:- Construction of SLR Parsing Table, before to that we must find the Follow of the grammar, here to find Follow no need to Eliminate left Recursion / left Factoring.

Scanned with OKEN Scanner

0 E → E+T

1 E → T

2 T → T*F

3 T → F

4 F → (E)

5 F → id

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

FOLLOW (E) = { $, +,) }

FOLLOW (T) = { $, +,), * }

FOLLOW (F) = { $, +,), * }

	ACTION						GOTO		
	id	+	*	(	)	$	E	T	F
0	S_5			S_4			1	2	3
1		S_6				Accept			
2		r_2	S_7		r_2	r_2			
3		r_4	r_4		r_4	r_4			
4	S_5			S_4			8	2	3
5		r_6	r_6		r_6	r_6			
6	S_5			S_4				9	3
7	S_5			S_4					10
8		S_6			S_{11}				
9		r_1	S_7		r_1	r_1			
10		r_3	r_3		r_3	r_3			
11		r_5	r_5		r_5	r_5			

Scanned with OKEN Scanner

Step 4:- Parse the input strings

Stack operations	Input Buffer	Action
$0	id * id + id $	Shift 5
$0id5	* id + id $	reduce F->id
$0F3	* id + id $	reduce T->F
$0T2	* id + id $	shift 7
$0T2*7	id + id $	Shift 5
$0T2*7id5	+ id $	reduce F->id
$0T2*7F10	+ id $	reduce T->T*F
$0T2	+ id $	reduce E->T
$0E1	+ id $	Shift 6
$0E1+6	id $	Shift 5
$0E1+6id5	$	reduce F->id
$0E1+6F3	$	reduce T->F
$0E1+6T9	$	reduce E->E+T
$0E1	$	Accept

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

(2) Consider the following grammar

$S \rightarrow As \mid b$

$A \rightarrow SA \mid a$

Construct SLR parse table for the grammar. Show the Actions of the parser for the input string "abab".

(Ans) Step 1:- Augmented Grammar.

$S' \rightarrow S$

$S \rightarrow AS$

$S \rightarrow b$

$A \rightarrow SA$

$A \rightarrow a$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Step 2:- Calculate Canonical Collection of LR(0) items.

I_0

0 $S' \rightarrow .S$

1 $S \rightarrow .AS$

2 $S \rightarrow .b$

3 $A \rightarrow .SA$

4 $A \rightarrow .a$

Scanned with OKEN Scanner

goto (I_0, S)

$S' \to S.$
$A \to S.A$
$A \to .SA$
$A \to .a$
$S \to .AS$
$S \to .b$

I_1

goto (I_0, A)

$S \to A.S$
$S \to .AS$
$S \to .b$
$A \to .SA$
$A \to .a$

I_2

goto (I_0, b)

$S \to b.$

I_3

goto (I_0, a)

$A \to a.$

I_4

~~goto (I_1, A)~~

~~$A \to .SA$~~
~~$A \to .a$~~
~~$S \to .AS$~~
~~$S \to .b$~~

~~$A \to SA.$~~
~~$S \to A.S$~~
~~$S \to .AS$~~
~~$S \to .b$~~

goto (I_1, S)

$A \to S.A$
$A \to .SA$
$A \to .a$
$S \to .AS$
$S \to .b$

I_6

goto (I_1, a)

$A \to a.$

I_4

goto (I_1, b)

$S \to b.$

I_3

goto (I_2, S)

$S \to AS.$
$A \to S.A$
$A \to .SA$
$A \to .a$
$S \to .AS$
$S \to .b$

I_7

goto (I_1, A)

$A \to SA.$
$S \to A.S$
$S \to .AS$
$S \to .b$

I_5

~~$A \to .AS$~~
~~A~~

$A \to .SA$
$A \to .a$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

~~goto (I_1, a)~~

~~$A \to a.$ — I_6~~

~~goto (I_1, b)~~

~~$S \to b.$ — I_4~~

~~goto (I_2, S)~~

$S \to AS.$

goto (I_2, A)

$S \to A.S$
$S \to .AS$
$S \to .b$
$A \to .SA$
$A \to .a$
— I_2

goto (I_2, b)

$S \to b.$ — I_3

goto (I_2, a)

$A \to a.$ — I_4

goto (I_2, S)

$S \to AS.$
$A \to S.A$
$A \to .SA$
$A \to .a$
$S \to .AS$
— I_7

goto (I_5, b)

$S \to b.$ — I_3

goto (I_5, a)

$A \to a.$ — I_4

goto (I_6, A)

$A \to SA.$
$S \to A.S$
$S \to .AS$
$S \to .b$
$A \to .SA$
$A \to .a$
— I_5

goto (I_6, S)

$A \to S.A$
$A \to .SA$
$A \to .a$
$S \to .AS$
$S \to .b$
— I_6

goto (I_6, a)

$A \to a.$ — I_4

goto (I_6, b)

$S \to b.$ — I_3

goto (I_5, A)

$S \to A.S$
$S \to .AS$
$S \to .b$
$A \to .SA$
$A \to .a$
— I_2

goto (I_7, A)

$A \to SA.$
$S \to A.S$
$S \to .AS$
$S \to .b$
$A \to .SA$
$A \to .a$
— I_5

goto (I_7, S)

~~$A \to S.A$~~
~~$A \to .SA$~~
~~$A \to .a$~~

$A \to S.A$
$A \to .SA$
$A \to .a$
$S \to .AS$
$S \to .b$
— I_6

~~goto (I_7, A)~~

~~$A \to SA.$~~
~~$S \to A.S$~~
~~$S \to .AS$~~
~~$S \to .b$~~
~~$A \to .SA$~~
~~$A \to .b$~~
~~— I_5~~

goto (I_7, a)

$A \to a.$ — I_4

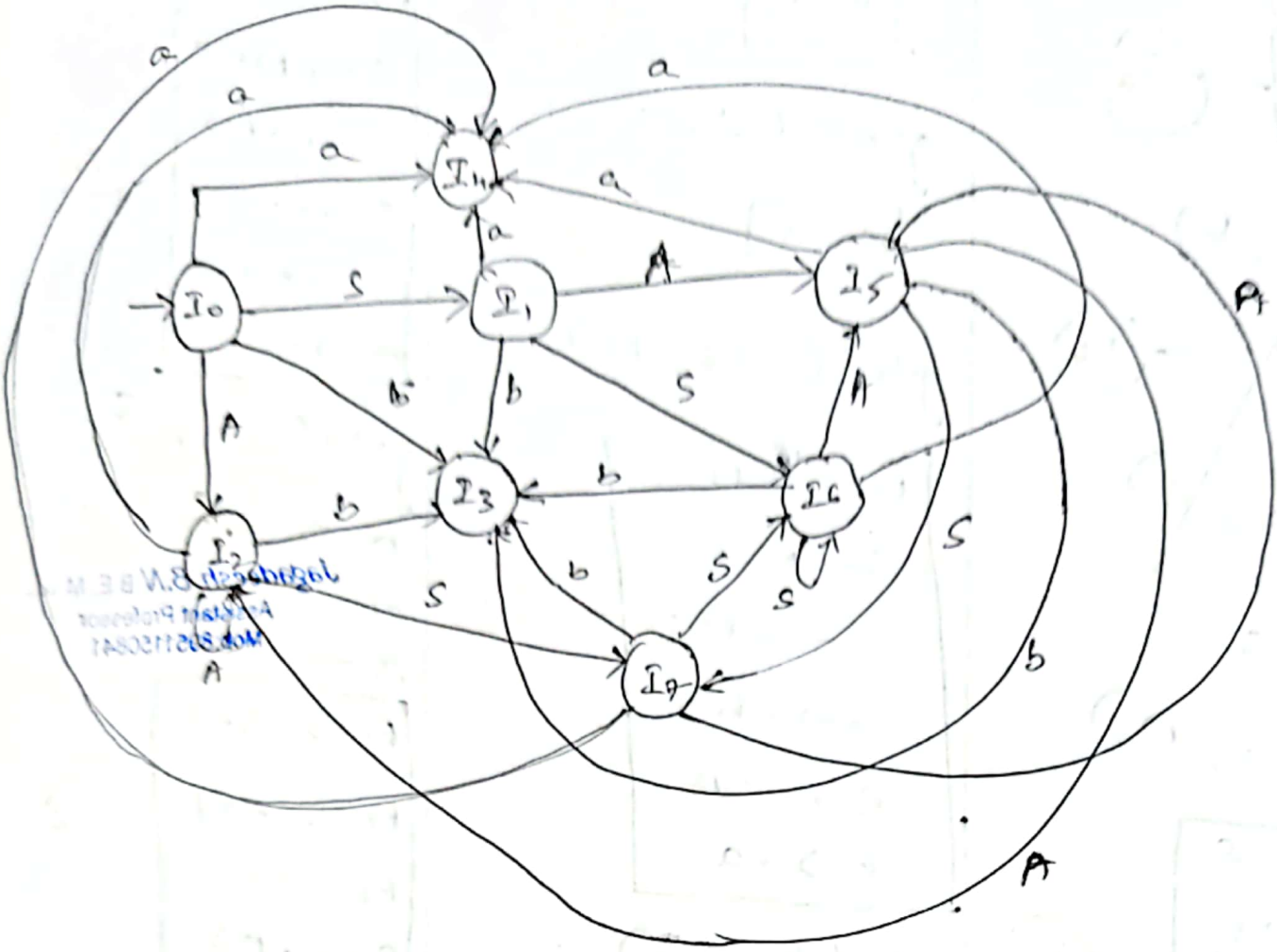
goto (I_7, b)

$S \to b.$ — I_3

Jagadeesh, B.N B.E, M.Tech
Assistant Professor
Mob 8951195841

Scanned with OKEN Scanner

Step 2:- Designing of the finite Automata.



Step 3:- Construction of SLR parsing table, before construction of SLR parsing table we must find the FIRST & FOLLOW of LHS of Non-terminals.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

~~Follow~~

FIRST(S) = {b, a}

FIRST(A) = {a, b}

FOLLOW(S) = {$, a, b}

FOLLOW(A) = {b, a}

Scanned with OKEN Scanner

	ACTION			GOTO	
	a	b	$	S	A
0	S4	S3		1	2
1	S4	S3	Accept	6	5
2	S4	S3		7	2
3	r2	r2	r2		
4	r4	r4			
5	S4/r3	S3/r3		7	2
6	S4	S3		6	5
7	S4/r1	S3/r1	r1	6	5

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

In the above parsing table, each and every cell must contain only one entry, but if we consider state (5, a) = S4/r3 and also in state (5, b) = (S3/r3) and (7, b) = (S3/r1) hence (7, a) = (S4/r1) and there is a shift-Reduce Conflicts.

Step4:- Let us check whether the string will be accepted by the parser/not.

Scanned with OKEN Scanner

STACK OPERATION	INPUT BUFFER	ACTION
$0	abab$	Shift 4
$0a4	bab$	reduce A->a
$0A2	bab$	Shift 3
$0A2b3	ab$	reduce S->b
$0A2S7	ab$	reduce S->AS
$0S1	ab$	shift 4
$0S1a4	b$	reduce A->a
$0S1A5	b$	reduce A->SA
$0A2	b$	Shift S3
$0A2b3	$	reduce S->b
$0A2S7	$	reduce S->AS

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

$0 S1	$	Accept

(3) Construct Canonical LR(0) items for the grammar,

$S \rightarrow L=R \mid R$

$L \rightarrow *R \mid id$

$R \rightarrow L$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Also Construct SLR parsing table and parse the Input String *id=id

Scanned with OKEN Scanner

More Powerful LR Parsers

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

CLR Parser

* CLR stands for Canonical LR parser where L means Scanning the i/p Symbol Left-to-Right and R means obtaining the RMD in reverse order.

* The difference ~~betw~~ between SLR and CLR parser is that in SLR we calculate LR(0) items and in CLR we calculate LR(1) items.

* LR(1) items means LR(0) + lookahead Symbol.

* Procedures is Similar to rest of SLR parser.

Scanned with OKEN Scanner

(3) Construct CLR parse table for the following grammar

$S \to CC$

$C \to cC \mid d$

and parse the input string "dd".

(Ans) Step1:-

$S' \to S$
$S \to CC$
$C \to cC$
$C \to d$

→ Augmented Grammar.

Step2:- Calculating LR(1) items.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

~~$S' \to .S, \$$~~
~~$S \to .CC$~~
~~$C \to .cC$~~
~~$C \to .d$~~

~~$S' \to .S, \$$~~
~~$S \to .CC,$~~

$S' \to .S, \$$ ⇒ first$(\varepsilon . \$) = \$$

$S \to .CC, \$$ ⇒ first$(C . \$) =$ first$(C) = \{c, d\} \Rightarrow c/d$

$C \to .cC, c/d$

$C \to .d, c/d$

I_0

Scanned with OKEN Scanner

goto (I_0, S)

$S' \to S. , \$$ — I_1

goto (I_0, C)

first $(\varepsilon . \$) = \$$

$S \to C . C , \$$

$C \to . cC , \$$

$C \to . d , \$$

— I_2

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

goto (I_2, c)

$C \to c . C , c/d$

$C \to . cC , c/d$

$C \to . d , c/d$

— I_3

goto (I_2, d)

$C \to d. , \$$ — I_7

goto (I_3, C)

$C \to cC. , c/d$ — I_8

goto (I_3, c)

$C \to c . C , c/d$

$C \to . cC , c/d$

$C \to . d , c/d$

— I_3

goto (I_3, d)

$C \to d. , c/d$ — I_4

goto (I_0, d)

$C \to d. , c/d$ — I_4

goto (I_2, C)

$S \to CC. , \$$ — I_5

goto (I_2, c)

$C \to c . C , \$$

$C \to . cC , \$$

$C \to . d , \$$

— I_6

Scanned with OKEN Scanner

goto (I6, c)

$C \to cC.$, $\$$ — I_9

goto (I6, c)

$C \to c.C$, $\$$

$C \to .cC$, $\$$

$C \to .d$, $\$$ — I_6

goto (I6, d)

$C \to d.$, $\$$ — I_7

0 $S' \to .S$

1 $S \to .CC$

2 $C \to .cC$

3 $C \to .d$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob 8951195841

Step 3:- Construction of CLR parse table

	ACTION			GOTO	
	c	d	$	S	C
0	S3	S4		1	2
1			ACCEPT		
2	S6	S7			5
3	S3	S4			8
4	r3	r3			
5			r1		
6	S6	S7			9
7			r3		
8	r2	r2			
9			r2		

Scanned with OKEN Scanner

Step4:- Parse the input string

STACK OPERATION	INPUT BUFFER	Action
$0	dd $	Shift 4
$0d4	d $	reduce r3 C->d
$0C2	d $	Shift 7
$0C2d7	$	reduce r3 C->d
$0C2C5	$	reduce r1 S->CC
$0S1	$	ACCEPT

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

Construct CLR parser for the grammar

$S \rightarrow L=R \mid R$

$L \rightarrow *R \mid id$

$R \rightarrow L$ and also parse the string id = id.

(Ans) Step 1:- Write the Augmented grammar.

$S' \rightarrow S$

$S \rightarrow L=R$

$S \rightarrow R$

$L \rightarrow *R$

$L \rightarrow id$

$R \rightarrow L$

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Step 2:- Calculate Canonical collection of LR(1) items.

I_0

$S' \rightarrow .S, \$$

$S \rightarrow .L=R, \$$

$S \rightarrow .R, \$$

$L \rightarrow .*R, =/\$$

$L \rightarrow .id, =/\$$

$R \rightarrow .L, \$$

Scanned with OKEN Scanner

goto (I_0, S)

$S' \to S.$, \$ — I_1

goto (I_0, L)

$S \to L. = R$, \$

$R \to L.$, \$ — I_2

goto (I_0, R)

$S \to R.$, \$ — I_3

goto (I_0, *)

$L \to *.R$, =/\$

$R \to .L$, =/\$

$L \to .*R$, =/\$

$L \to .id$, =/\$ — I_4

goto (I_0, id)

$L \to id.$, =/\$ — I_5

goto (I_2, =)

$S \to L = .R$, \$

$R \to .L$, \$

$L \to .*R$, \$

$L \to .id$, \$ — I_6

goto (I_4, R)

$L \to *R.$, =/\$ — I_7

goto (I_4, L)

$R \to L.$, =/\$ — I_8

goto (I_4, *)

$L \to *.R$, =/\$

$R \to .L$, =/\$

$L \to .*R$, =/\$

$L \to .id$, =/\$ — I_4

goto (I_4, id)

$L \to id.$, =/\$ — I_5

goto (I_6, R)

$S \to L = R.$, \$ — I_9

goto (I_6, L)

$R \to L.$, \$ — I_{10}

goto (I_6, *)

$L \to *.R$, \$

$R \to .L$, \$

$L \to .*R$, \$

$L \to .id$, \$ — I_{11}

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

goto (I_6 , id)

$L \to id \cdot , \$$ — I_{10}

goto (I_{11} , R)

$L \to * \cdot R \cdot , \$$ — I_{13}

goto (I_{11} , L)

$R \to L \cdot , \$$ — I_{10}

goto (I_{11} , *)

$L \to * \cdot R , \$$

$R \to \cdot L , \$$

$L \to \cdot * R , \$$

$L \to \cdot id , \$$ — I_N

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

goto (I_{11} , id)

$L \to id \cdot , \$$ — I_{12}

0 $S' \to \cdot S$

1 $S \to \cdot L = R$

2 $S \to \cdot R$

3 $L \to \cdot * R$

4 $L \to \cdot id$

5 $R \to \cdot L$

Scanned with OKEN Scanner

Step3:- Construction of CLR parsing table.

	ACTION				GOTO		
	id	=	*	$	S	L	R
0	S_5		S_4		1	2	3
1				Accept			
2		S_6		r_5			
3				r_2			
4	S_5		S_4			8	7
5		r_4		r_4			
6	S_{12}		S_{11}			10	9
7		r_3		r_3			
8		r_5		r_5			
9				r_1			
10				r_5			
11	S_{12}		S_{11}			10	13
12				r_4			
13				r_3			

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

Step 4:- Parse the input String

Stack Operation	Input Buffer	Action
$0	id = id $	shift 5
$0 id 5	= id $	reduce r4 L->id
$0 L 2	= id $	shift 6
$0 L 2 = 6	id $	Shift 12
$0 L 2 = 6 id 12	$	reduce r4 L->id
$0 L 2 = 6 L 10	$	reduce r5 R->L
$0 L 2 = 6 R 9	$	reduce r1 S->L=R
~~$0 L 2 = 6~~ $0 S 1	$	ACCEPT

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

Scanned with OKEN Scanner

* LALR stands for Look - Ahead, L Scan the i/p symbol from left to Right R Means obtain the RMD in reverse order.

Construct LALR parser for the grammar

$S \to CC$

$C \to aC \mid b$

parse the string bb

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

(Ans) Step 1 :- Write the Augmented grammar

0 $S' \to S$

1 $S \to CC$

2 $C \to aC$

3 $C \to b$

Scanned with OKEN Scanner

Step2 :- Calculate LR(1) items.

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob.8951150841

I_0:
$S' \to .S, \$$
$S \to .CC, \$$
$C \to .aC, a/b$
$C \to .b, a/b$

goto (I_0, S)
I_1:
$S' \to S., \$$

goto (I_0, C)
I_2:
$S \to C.C, \$$
$C \to .aC, \$$
$C \to .b, \$$

goto (I_0, a)
I_3:
$C \to a.C, a/b$
$C \to .aC, a/b$
$C \to .b, a/b$

goto (I_0, b)
I_4:
$C \to b., a/b$

goto (I_2, C)
I_5:
$S \to CC., \$$

goto (I_2, a)
I_6:
$C \to a.C, \$$
$C \to .aC, \$$
$C \to .b, \$$

goto (I_2, b)
I_7:
$C \to b., \$$

goto (I_3, C)
I_8:
$C \to aC., a/b$

goto (I_3, a)
I_3:
$C \to a.C, a/b$
$C \to .aC, a/b$
$C \to .b, a/b$

goto (I_3, b)
I_9:
$C \to b., \$$

goto (I_6, C)
I_9:
$C \to aC., \$$

goto (I_6, a)
I_6:
$C \to a.C, \$$
$C \to .aC, \$$
$C \to .b, \$$

goto (I_6, b)
I_7:
$C \to b., \$$

Scanned with OKEN Scanner

Next if we consider the state I_3 & I_6 productions are same, only its look-Ahead symbol is different.

I_{36}

$C \to a.C,\ a/b/\$$
$C \to .aC,\ a/b/\$$
$C \to .b,\ a/b/\$$

I_{36}

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

$C \to b.,\ a/b/\$$

I_{47}

$C \to aC.,\ a/b/\$$

I_{89}

Step3:- Construction of LALR parse table

	ACTION			GOTO	
	a	b	$	S	C
0	S_{36}	S_{47}		1	2
1			ACCEPT		
2	S_{36}	S_{47}			5
36	S_{36}	S_{47}			89
47	r_3	r_3	r_3		
5			r_1		
89	r_2	r_2	r_2		

Scanned with OKEN Scanner

Step4:- Let us see how to parse the input string.

STACK OPERATION	INPUT BUFFER	ACTION
$0	bb $	shift 47
$0b47	b $	reduce r3 C->b
$0C2	b $	shift 47
$0C2b47	$	reduce r3 C->b
$0C2C5	$	reduce r1 S->CC
$01	$	ACCEPT

Jagadeesh B.N B.E, M.Tech
Assistant Professor
Mob:8951150841

(2) Construct LALR parser for the grammar:

S -> L=R | R

L -> *R | id

R -> L and also parse the string id=id.

Scanned with OKEN Scanner