Mealy Machine and Moore Machine:

Mealy machine: It is a finite state machine where the output is determined by both the current state and the current input.

Mathematically represented as:

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

Where,

- Q : A finite set of states.
- Σ : Input alphabets.
- Δ : Output alphabets.
- $\delta : Q \times \Sigma \rightarrow Q$: State transition function.
- $\lambda : Q \times \Sigma \rightarrow \Delta$: Output function.
- q_0 : Initial state.

Moore machine: It is a type of finite state machine where the output is determined by the current state only, not the input.

Mathematically reprented as:

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

Where,

everything is same except,

$\lambda : Q \rightarrow \Delta$: Output function.

Key differences:

Feature	mealy	moore
Output depends on	Current State and input	Current state only
Output timing	Changes immediately	Delayed by one state
Representation	more compact	Larger state space.

Conversion Rules:

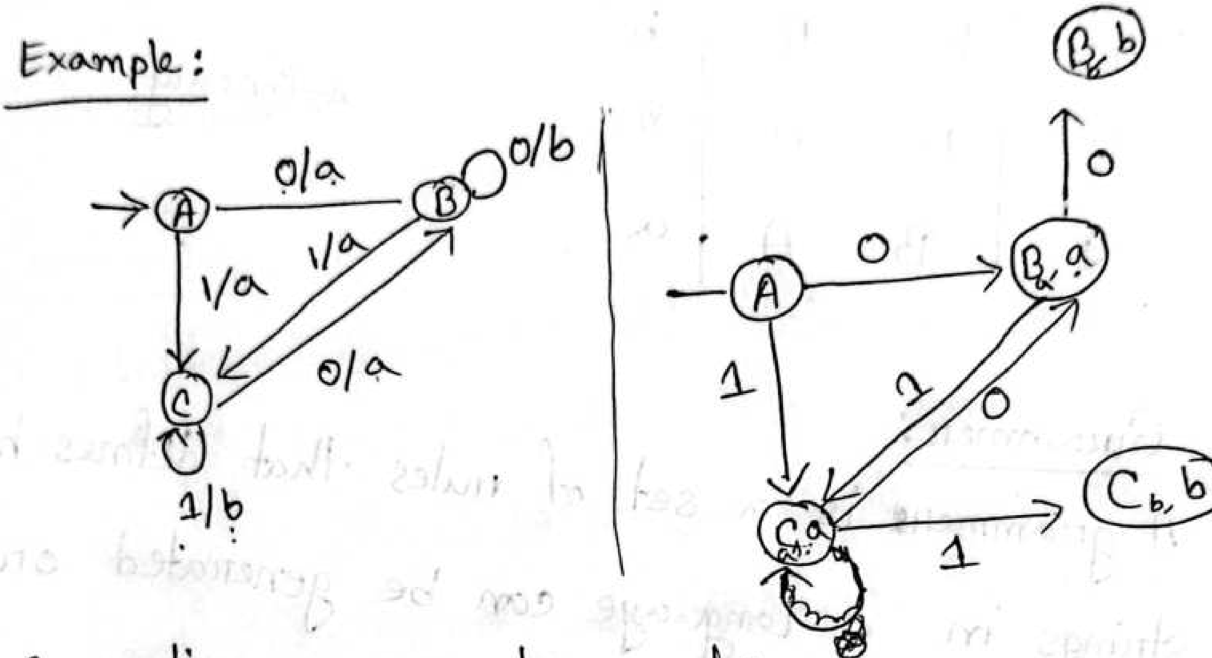
Mealy to Moore,

1. For each input output pair in the mealy machine create a unique state in the moore machine.
2. Assign the output of each mealy transition to the state created for that transition in the moore machine.

3. Ensure the transition in the moore machine correspond to the transition in the mealy machine.

4. Adjust the diagram where to represent the moore machine where the output is explicitely linked to the state.

Example:



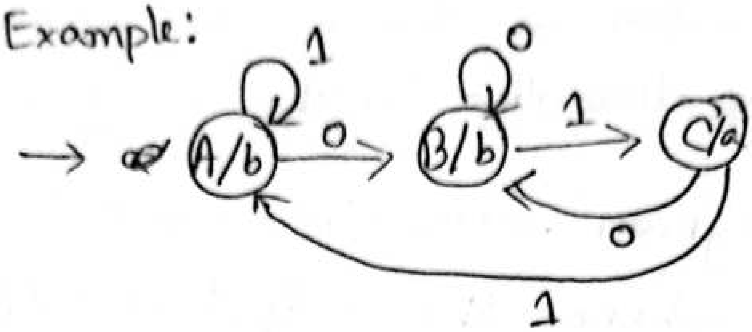
Converting moore to mealy:

1. For each state in the moore machine, assign its to all transition originating from that state in the mealy machine.

2. Maintain the same transition as the mooremachine

3. Add the the output to each transition in the mealy machine.

Example:



State	0	1	Output
→A	B	A	b
B	B	C	b
C	B	A	a

← mealy

Grammer:

A grammers is a set of rules that defines how strings in a language can be generated or derived.

Components:

$G = (T, N, S, P)$

1. T : Terminals ⇒ Symbols that appear in the final string (e.g. a, b).
2. ~~Non~~ N/V : Non-terminals → Variables used to derive strings. (e.g. S, A, B).

3. S: Start Symbol → A special non-terminal from which the derivation begins.

4. Productions P: Rules that form $A \rightarrow \beta$, where $A \in N$ and $\beta \in (N+T)^*$

Trapple: A constraint or property that limits or modifies how grammar rules are applied.

Derivation:

1. Leftmost derivation: At each step, replace the left most non-terminal in the string.
2. Right most derivation: At each step, replace the ~~lef~~ right most non-terminal in the string.
3. Left most Bottom-up: Derivation proceeds upwards starting with the leftmost terminal.
4. Right most Bottom up: Derivation proceeds upwards, starting with the rightmost element/terminal.

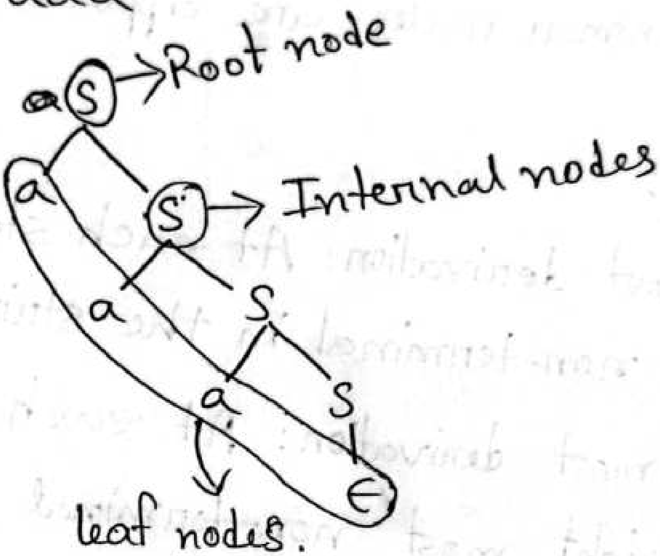
Derivation tree:

A parse tree is a tree representation of the derivation of a string according to a grammer.

Example.

$S \rightarrow aS \mid \epsilon$

string aaa



Classification of Grammer:

Chomski Hierarchy,

1. Type 0 (Unrestricted grammer)
   - Productions have no restrictions
   - Form $\alpha \rightarrow \beta$, where $\alpha, \beta \in (N+T)^*$ and $\alpha \neq \epsilon$
   - Language: Recursive enumerable language.
   - Automata: Turing Machine.

2. Type 1 (Context Sensitive Grammer):

- Length of L.H.S $\leq$ Length of R.H.S in the production Rules.
- $S \to \epsilon$ works only if S is start symbol and no $S \to aS$ found.
- Language: conatext free languge.
- Automata: Linear Bounded Automata

3. Type 2 (Context Free ~~Language~~ Grammer):

- Production rules are $\alpha \to \beta$ where, α is only one non terminal.
- Language: Context free language.
- Automata: Push Down Automata.

4. Type 3 (Regular Grammer):

- Production are of the form,

$$X \to Y \; ; \; Y \in VT^* + T^* \text{ (Left Linear)}$$

$$\in T^*V + T^* \text{ (Right Linear)}$$

$V \to$ nonterminal, $T \to$ Terminal.

language: Regular Language.
Automata: Finite Automaton

Context Free Grammer:

A Grammer where all production are of the form $A \to \beta$ where A is a single non-terminal, β is a string of terminals and/or non terminals.

Example: $S \to AB$
$A \to aA | \epsilon$
$B \to bB | \epsilon$

Context Sensitive Grammer:

A grammer where the length of left hand side of a production is less than or equal to the length of right hand side.

Example $AB \to ABB$

Hierarchy:

Type 3 $\subseteq$ Type 2 $\subseteq$ Type 1 $\subseteq$ Type 0

Regular Grammer:

A regular grammer is a type of formal grammer in the chomski hierarchy that generates regular languges. It is used to describe languages that can be recognised by finite automaton.

Type 1: Production rules form:

$A \rightarrow aB \mid a$ (Right linear grammer)

Type 2: Production rules form:

$A \rightarrow Ba \mid a$ (Left linear grammer)

Pushdown Automaton:

A pushdown automata is a type of computational model used to recognize context free languages. It extends the capabilities of a finite automaton by including a stack as an auxiliary memory structure.

Formal Definition:

A PDA is a 7-tuple:

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

Where,

Q : finite set of states.

Σ : The input alphabet.

Γ : the stack alphabet.

δ : the transition function:

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*} \text{ for (N-PDA)}$$
$$\rightarrow Q \times \Gamma^* \text{ for (D-PDA)}$$

q_0 : the initial state ($q_0 \in Q$)

z_0 : the initial stack symbol ($z_0 \in \Gamma$)

F : a set of accepting state ($F \subseteq Q$)

Types of PDA:

1: Deterministic PDA.

- For Every state, input and stack condition - there is at most one transition.
- Recognizes a subset of CFL.

2. Non- Deterministic PDA:

- For a given state, input and stack combination, multiple transition may possible.
- Recognizes all context free languages.

Operations:

1. Input Reading
2. Stack operations (Push, Pop, Skip)
3. Acceptence,

   PDA accepts input string if,

   i) It reaches accepting state, or,

   ii) The stack is empty.

Example:

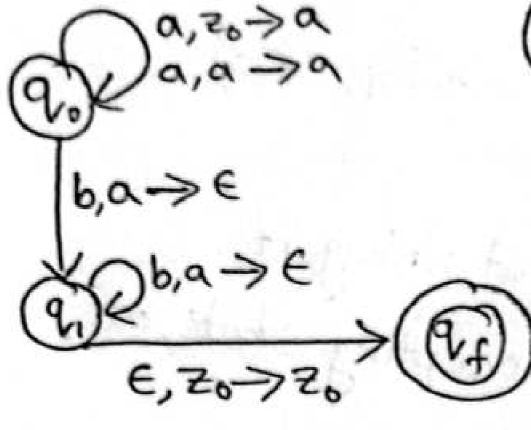
PDA for $L = \{ a^n b^n \mid n \geq 0 \}$:

1. Push a onto the stack for each a read.
2. Pop a from the stack for each b read.
3. Accept if the stack is empty when input ends.

Transitions:

$$\delta(q_0, a, Z_0) = (q_0, AZ_0)$$
$$\delta(q_0, a, A) = (q_0, AA)$$
$$\delta(q_0, b, A) = (q_1, \epsilon)$$
$$\delta(q_1, b, A) = (q_1, \epsilon)$$
$$\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$$

Pictorial Representation:



(Note: The Z_0 was initially pushed in-to the stack).

CFG to CNF (Chomski Normal Form):

Example:

$S \rightarrow ASA \mid aB$
$A \rightarrow B \mid S$
$B \rightarrow b \mid \epsilon$

Step 1:

$S_0 \rightarrow S$ (new)
$S \rightarrow ASA \mid aB$
$A \rightarrow B \mid S$
$B \rightarrow b \mid \epsilon$

Step 2: Remove null production,

Removing $B \rightarrow \epsilon$,

$S_0 \rightarrow S$ $\qquad B \rightarrow b$
$S \rightarrow ASA \mid aB \mid a$
$A \rightarrow B \mid S \mid \epsilon$

Removing $A \to \epsilon$,

$S_0 \to S$
$S \to ASA \mid aB \mid a \mid SA \mid AS \mid S$
$A \to B \mid S$
$B \to b$

Step 3: (Remove unit production)

$S \to ASA \mid aB \mid a \mid SA \mid AS$
$A \to b \mid ASA \mid aB \mid a \mid SA \mid AS$
$B \to b$

Step 4:

$S \to AX \mid aB \mid a \mid SA \mid AS$
$A \to b \mid AX \mid aB \mid a \mid SA \mid AS$
$X \to SA$
$B \to b$

Step 5:

$S \to AX \mid YB \mid a \mid SA \mid AS$
$A \to b \mid AX \mid \cancel{a}YB \mid a \mid SA \mid AS$
$X \to SA$
$Y \to a$
$B \to b$

Personal

CNF to GNF (Greibach Normal Form):

Example:

$S \rightarrow CA \mid BB$

$B \rightarrow b \mid SB$

$C \rightarrow b$

$A \rightarrow a$

step 1:

$S \rightarrow bA \mid bB \mid SBB$ (Left recursion)

$C \rightarrow b$

$B \rightarrow b \mid \cancel{bAB \mid bBB \mid SBBB}\ SB$

$A \rightarrow a$

Step 2:

$S \rightarrow bAZ \mid bBZ$

$Z \rightarrow BBZ \mid \epsilon$

$\cancel{B \rightarrow b \mid bAB \mid bBB \mid bAZBBB}$

$B \rightarrow \cancel{bBZBBB}\ b \mid SB$

$A \rightarrow a$

$C \rightarrow b$

~~left~~ recursion removal,

$\therefore A \rightarrow A\alpha \mid \beta$

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' \mid \epsilon$

Step 3:

$S \rightarrow bAZ \mid bBZ$ (removed unnecessary C)

$Z \rightarrow BBZ \mid \epsilon$

$B \rightarrow b \mid bAZB \mid bBZB$

$A \rightarrow a$

Step 4:

$S \rightarrow bAZ \mid bBZ$

$Z \rightarrow bBZ \mid bAZBBZ \mid bBZBBZ \mid \epsilon$

$B \rightarrow b \mid bAZB \mid bBZB$

$A \rightarrow a$

Step 5: (Removing $Z \rightarrow \epsilon$)

$S \rightarrow bAZ \mid \cancel{B}bBZ \mid \cancel{A}bA \mid bB$

$Z \rightarrow bBZ \mid bAZBBZ \mid bBZBBZ \mid bB \mid bABBZ \mid bAZBB \mid$
$bABB \mid bBBBZ \mid bBZBB \mid bZBB$

$B \rightarrow b \mid bAZB \mid bBZB \mid bAB \mid bBB$

$A \rightarrow a$

Personal

PDA to CFG

1. $S \rightarrow [q_0 Z P]$ for each P, where $P \in Q$

2. If $\delta(q\, x\, A) = (P, B_1 B_2 B_3 \dots B_m)$

then,

$$[q\, A\, q_{m+1}] \rightarrow x\, [q\, B_1\, q_{\square}]\, [q_{\square}\, B_2\, q_{\Delta}] \dots [q_{\circ}\, B_m\, q_{m+1}]$$

3. if $\delta(q\, x\, A) = (P, \epsilon)$ then,

$[q\, A\, P] \rightarrow x$ where, $x \in$ Terminals

$A, B \in \Gamma$

Example:

$L = \{ a^n b^n \mid n \geq 0 \}$

$Q = \{q_0, q_1\}$

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

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

PDA

$\delta(q_0, a, Z_0) \rightarrow (q_0, a Z_0)$

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

$\delta(q_0\, b, a) \rightarrow (q_1\, \epsilon)$

$\delta(q_1, b, a) \rightarrow (q_1\, \epsilon)$

$\delta(q_1\, \epsilon\, Z_0) \rightarrow (q_1, Z_0)$

Personal

CFG

$S \rightarrow [q_0 z_0 q_0] \mid [q_0 z_0 q_1]$

(1) $[q_0 z_0 q_0] \rightarrow a[q_0 a q_0][q_0 z_0 q_0]$ ×

$[q_0 z_0 q_0] \rightarrow a[q_0 a q_1][q_1 z_0 q_0]$ ×

$[q_0 z_0 q_1] \rightarrow a[q_0 a q_0][q_0 z_0 q_1]$ ×

$[q_0 z_0 q_1] \rightarrow a[q_0 a q_1][q_1 z_0 q_1]$

(2) $[q_0 a q_0] \rightarrow a[q_0 a q_0][q_0 a q_0]$ ×

$[q_0 a q_0] \rightarrow a[q_0 a q_1][q_1 a q_0]$ ×

$[q_0 a q_1] \rightarrow a[q_0 a q_0][q_0 a q_1]$ ×

$[q_0 a q_1] \rightarrow a[q_0 a q_1][q_1 a q_1]$

(3) $[q_0 a q_1] \rightarrow b$

(4) $[q_1 a q_1] \rightarrow b$

(5) $[q_1 z_0 q_1] \rightarrow \epsilon$

Let,

$[q_0 z_0 q_1] = D$

$[q_0 a q_1] = C$

$[q_1 z_0 q_1] = A$

$[q_1 a q_1] = B$

Personal

∴ Final CFG

$S \to D$

$D \to aCA$

$C \to aCB$

$C \to b$

$B \to b$

$A \to \epsilon$

CFG to PDA:

No GNF $\Gamma = (V \cup T)$

Example:

$S \to aSb \mid ab$

$$\left.\begin{array}{l} 1.\ \delta(q, \epsilon, S) = (q, aSb) \\ 2.\ \delta(q, \epsilon, S) = (q, ab) \end{array}\right\} \text{for variable}$$

$$\left.\begin{array}{l} 3.\ \delta(q, a, a) = (q, \epsilon) \\ 4.\ \delta(q, b, b) = (q, \epsilon) \end{array}\right\} \text{for terminal}$$

GNF $\Gamma' = (V)$

Example:

$$S \to 0BB$$
$$B \to 0S \mid 1S \mid 0$$

$$\delta(q, 0, S) = (q, BB)$$
$$\delta(q, 0, B) = (q, S)$$
$$\delta(q, 1 B) = (q, S)$$
$$\delta(q, 0, B) = (q, \epsilon)$$