THEORY OF COMPUTATION

UNIT - 4

K CHAIRMADURAI ME,.. AP
Department of Computer Science and Engineering
Adhiparasakthi Engineering College , Melmaruvathur

Unit IV Properties of Context Free languages

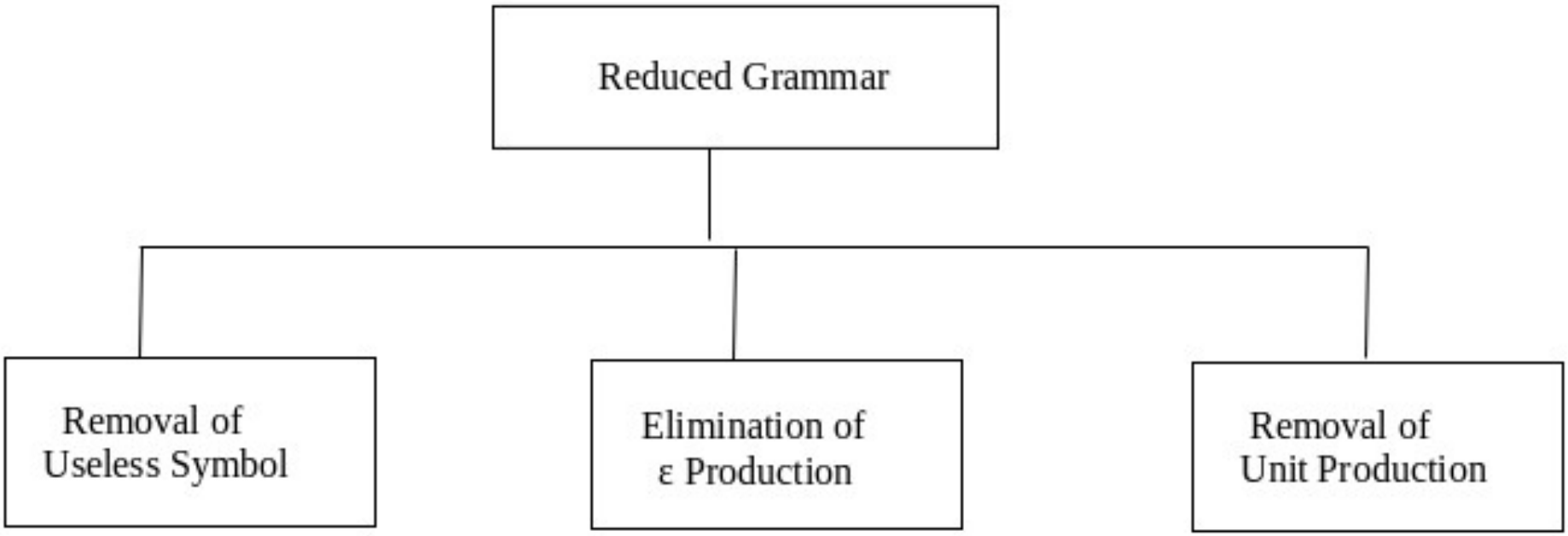
Syllabus:

- **Normal Forms of CFG**
- **Pumping Lemma for CFL**
- **Closure Properties of CFL**
- **Turing Machines**
- **Programming Techniques for TM**

Simplification of CFG

- Various languages can efficiently be represented by a context-free grammar.
- All the grammar are not always optimized that means **the grammar may consist of some extra symbols(non-terminal)**.
- Having extra symbols, **unnecessary increase the length of grammar**.
- **Simplification of grammar** means **reduction of grammar by removing useless symbols.**
- The properties of reduced grammar are given below:
  - **Each variable (i.e. non-terminal) and each terminal of G appears in the derivation of some word in L.**
  - **There should not be any production as X → Y where X and Y are non-terminal.**
  - **If ε is not in the language L then there need not to be the production X → ε.**

Simplification of CFG



Elimination of Useless Symbols

- Useful symbols are the variables and terminals that have **some production that leads from the starting symbol** to a string type $S \overset{*}{\Rightarrow} \omega$.
- Consider the derivation $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} \omega$ where $\omega \in T^*$
  - Here α,β are any terminals and X can be either terminal or variable.
  - If **X has no production**, (if X ϵ V) then X is a **useless symbol.**
  - If **X has some production** / if X leads to a string, with zero/more substitution, then it is a **useful symbol.**
- Thus, **useless symbols** are those variables/terminals that **do not appear in any derivation** of a terminal string from the start string.
- We need to determine if the symbol is useful by identifying if a symbol is **generating** and is **reachable.**

Generating symbols

- A symbol **X which is an element of V** [variable] is **generating symbol**

$$\text{if} \quad X \overset{*}{\underset{G}{\Rightarrow}} \omega \ , \text{where} \quad \omega \in T^*$$

- **Generating symbols** are variables that **generate a string containing terminals. (some terminal string must be derivable from X)**
- A **non generating symbol** is one that **cannot generate any terminal.**
- The **presence of production with non-generating symbol is useless**.
- **All the terminals symbols are generating symbols.**

Removal of Non Generating Symbols

Example

$S \rightarrow aA \mid bB \mid a \mid b$

$A \rightarrow Aa \mid a$

$B \rightarrow bB$

Here $B \rightarrow bB$ is a recursive grammar that its substitution to S results in endless derivation and doesnot give lead to any string of terminals.

$S \Rightarrow bB \Rightarrow bbB \Rightarrow bbbB \Rightarrow bbbbB \Rightarrow \ldots\ldots$

After eliminating the production for $B \rightarrow bB$, the resulting grammar includes.

$S \rightarrow aA \mid a \mid b \Rightarrow$ Eliminating the variable B

$A \rightarrow aA \mid a$

Reachable Symbols

- A **symbol is said to be reachable** if and only if it can be **reached from the start symbol.** $\text{If } S \overset{*}{\underset{G}{\Rightarrow}} \alpha, \text{ where } \alpha \text{ contains } X \quad [X \in V]$
- Then **X is said to be a reachable symbol.**
- **X must be occur in some string derivable from S.**
- If the **derivation from the start symbol** to a **string of terminals does not depend on a variable X** then **X is non-reachable symbol.**

Removal of non reachable symbols

Example

Let

$S \rightarrow aAa$

$A \rightarrow Sb \mid bCC$

$C \rightarrow abb$

$E \rightarrow aC$

- In the given grammar, S depends on A, A depends on S and C, E depends on C, and C does not depend on any variable.
- Here E is the only production which is not being depended by any other productions.

$E \rightarrow$ Non reachable symbol

After eliminating the non-reachable symbol, E we get the following grammars as,

$S \rightarrow aAa$

$A \rightarrow Sb \mid bCC$

$C \rightarrow abb$

Excercise - Eliminate useless Symbols

$S \rightarrow AB \mid a$
$A \rightarrow b$

Initial CFL grammar

Excercise - Eliminate useless Symbols

$S \rightarrow AB \mid a$
$A \rightarrow b$

Identify generating symbols

$S \rightarrow a$
$A \rightarrow b$

Remove non-generating

$S \rightarrow a$
$A \rightarrow b$

Identify reachable symbols

$S \rightarrow a$

Eliminate non-reachable

Home Work

Eliminate the non generating symbols from the given grammar.

$S \rightarrow AB \mid CA$

$B \rightarrow BC \mid AB$

$C \rightarrow aB \mid b$

$A \rightarrow a$

$S \rightarrow CA$

$A \rightarrow a$

$C \rightarrow b$

Find and eliminate the non generating symbols from

$S \rightarrow 0A0$

$A \rightarrow S1|1CC|D0A$

$C \rightarrow 011|DD$

$D \rightarrow 0D0$

$E \rightarrow 0C$

$S \rightarrow 0A0$

$A \rightarrow S1|1CC$

$C \rightarrow 011|$

$E \rightarrow 0C$

Eliminate the non reachable symbols from the given grammar.

$S \rightarrow 0B0|BC,$

$A \rightarrow 0C|BCC,$

$C \rightarrow 0,$

$B \rightarrow 1CC,$

$D \rightarrow E,$

$E \rightarrow 1$

$S \rightarrow 0B0 \mid BC$

$C \rightarrow 0$

$B \rightarrow 1CC$

Eliminate the non - reachable symbols from the grammar,

$S \rightarrow aAa$

$A \rightarrow bBB$

$B \rightarrow ab$

$C \rightarrow aB$

$S \rightarrow aAa,$

$A \rightarrow bBB,$

$B \rightarrow ab$

Elimination of ε Production

- The productions of type **S → ε** are called **ε productions**.
- Null productions are those productions that are of the form: **X → ε**

- **Finding and handling ε productions**

  - Find the **nullable variables**, which are of the form X → ε
  - The **nullable variable used in the productions is removed.**
  - **Remove the nullable production**.

Elimination of ε Production

Example

$$S \rightarrow aS \mid \varepsilon$$

Here $S \rightarrow aS$ and $S \rightarrow \varepsilon$ are the productions

$S \rightarrow aS$ doesn't contain null production.

But $S \rightarrow \varepsilon$ is an epsilon production.

S is said to be nullable variable.

By eliminating the nullable variable, the production becomes,

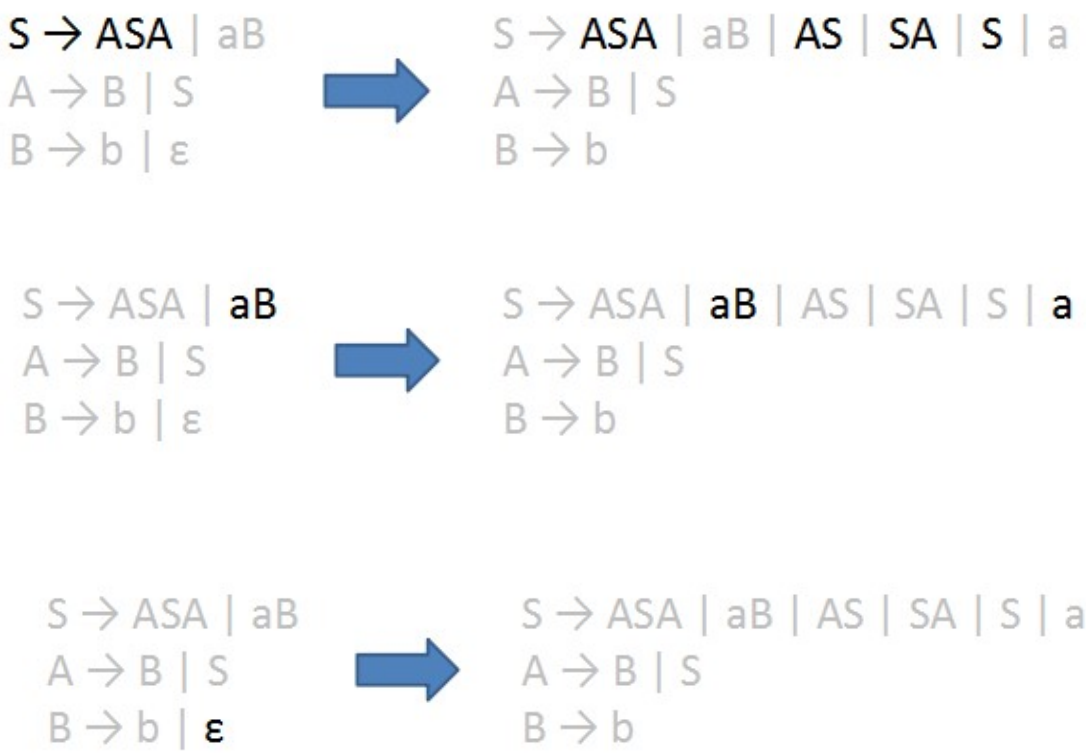
$S \rightarrow aS$ and on applying $S \rightarrow \varepsilon$ on $S \rightarrow aS$, then S becomes $S \rightarrow a$.

Therefore, $S \rightarrow aS|a$ is the final production.

Example - Elimination of ε Production

S → ASA | aB
A → B | S
B → b | ε

Nullable: {A, B}



Home Work

Remove ε productions from the grammar,

$S \rightarrow ABA$

$A \rightarrow \varepsilon$

$B \rightarrow \varepsilon$

Find the nullable symbols from the following grammar

$S \rightarrow aS \mid AB,$

$A \rightarrow \varepsilon$

$B \rightarrow \varepsilon$

$S \rightarrow aS \mid a$

Remove $\in$ - productions from the below given grammar.

$S \rightarrow AB$

$A \rightarrow aAA \mid \varepsilon$

$B \rightarrow bBB \mid \varepsilon$

$S \rightarrow AB \mid A \mid B$

$A \rightarrow aAA \mid aA \mid a$

$B \rightarrow bBB \mid bB \mid b$

Consider the grammar

$S \rightarrow ABC$

$A \rightarrow BC \mid a$

$B \rightarrow bAC \mid \varepsilon$

$C \rightarrow cAB \mid \varepsilon$

$S \rightarrow ABC \mid AB \mid BC \mid AC \mid A \mid B \mid C$

$A \rightarrow BC \mid B \mid C \mid a$

$B \rightarrow bAC \mid bA \mid bC \mid b$

$C \rightarrow cAB \mid cA \mid cB \mid C$

Identify the nullable symbols and eliminate them.

Elimination of Unit Productions

- The unit productions are those productions of the form **X → Y** where **X and Y are variables** of the grammar.
- The **left hand side** and **right hand side** contains **only one variable each side.**

Example

$S \rightarrow A \mid bb$

$A \rightarrow B \mid b$

$B \rightarrow S \mid a$

Here

$S \rightarrow A$,

$A \rightarrow B$,

$B \rightarrow S$ are all the unit productions.

Elimination of Unit Productions

Finding and eliminating unit productions

- Identify the non-unit productions in the grammar. Let it be $Y \rightarrow \alpha$, where α is a terminal.
- Select the unit production, $X \rightarrow Y$ where X and Y are variables.
- Add the production $X \rightarrow \alpha$ to the grammar since $X \rightarrow Y$ and $Y \rightarrow \alpha$.
- Remove the unit production, $X \rightarrow Y$ from the grammar.

Elimination of Unit Productions

Eliminate the unit production from the grammar,

S →ABA | BA | AA | AB | A | B

A → 0A | 0

B → 1B | 1

Solution

The unit productions are

S → A

S → B

The non unit productions are

S →ABA | BA | AA | AB

A → 0A | 0

B → 1B | 1

Since S → A, we shall use A's productions A → 0A | 0 to S.

Similarly, S → B, we shall use B's productions B → 1B | 1to S.

Thus the grammar becomes

S →ABA | BA | AA | AB | 0A | 0 | 1B | 1,

A → 0A | 0

B → 1B | 1

Home work

Eliminate the unit productions for following grammars

S → 0A | 1B | C
A → 0S | 00
B → 1 | A
C → 01

S -> S + T/ T
T -> T * F/ F
F -> (S)/a

S -> Aa/B/c
B -> A/bb
A -> a/bc/B

Home Work

Simplify the following grammar

$S \rightarrow ASB \mid \varepsilon$

$A \rightarrow aAS \mid a$

$B \rightarrow SbS \mid A \mid bb$

Answers

$S \rightarrow ASB \mid AB$

$A \rightarrow aAS \mid aA \mid a$

$B \rightarrow SbS \mid Sb \mid bS \mid b \mid bb \mid aAS \mid aA \mid a$

Simplify the following grammar

$S \rightarrow aAa \mid bBb \mid BB$

$A \rightarrow C$

$B \rightarrow S \mid A$

$C \rightarrow S \mid \varepsilon$

$S \rightarrow aAa \mid aa \mid bBb \mid bb \mid BB$

$A \rightarrow aAa \mid aa \mid bBb \mid bb \mid BB$

$B \rightarrow aAa \mid aa \mid bBb \mid bb \mid BB$

Elimination of null production

The epsilon production in the grammar is S → ε and the variable S has its impact on A and B productions.

The nullable variable are {S, A, B}

After removing S → ε, we get the final productions as

S → ASB | AB [since S → ε, exclude S]

A → aAS | aA | a

B → SbS | Sb | bS | b | A | bb

Elimination of unit productions

The unit production in the above grammar is B → A.

Hence eliminate the unit production in B and substitute the corresponding production of A in B. We get,

S → ASB | AB

A → aAS | aA | a

B → SbS | Sb | bS | b | bb | aAS | aA | a

Since the grammar doesn't contain any useless symbols, the above production becomes the final one.

Elimination of ε - productions

The nullable symbol is C since C → ε.

After removing C → ε, the grammar becomes,

S → aAa | bBb | aa | bb | BB | B [Since C → S | ε]

A → C

B → S | A

C → S

Elimination of unit productions

The unit productions in the above grammar are

S → B

A → C

B → S

B → A

C → S

The grammar without the above unit productions are

S → aAa | aa | bBb | bb | BB

A → aAa | aa | bBb | bb | BB [since A → C → S]

B → aAa | aa | bBb | bb | BB [since B → A → C → S and B→S]

C → aAa | aa | bBb | bb | BB [since C → S]

Elimination of non - generating symbols

From the above grammar

C is a non generating symbol, since it is has no derivation with respect to the start symbol. Hence C → aAa | aa | bBb | bb | BB is removed.

NORMAL FORMS FOR CFG

- Let the grammar **G = (V, T, P, S) be a context free grammar**. **If the production rules in G satisfy some restrictions,** then they are said to be in **normal form.**
- Normalization is performed in order to **standardize the grammar.**
- **By reducing** the grammar, the grammar **gets minimized** but does **not gets standardized**.
- This is because the RHS of productions **have no specific format.**
- In order to **standardize the grammar, normalization** is performed using normal forms.
- There are two types of normal forms
  - **Chomsky normal form (CNF)**
  - **Greibach normal form (GNF)**

CHOMSKY NORMAL FORM (CNF)

- A context free grammar is said to be in **chomsky normal form** (CNF) if all its productions are of the form-

$$A \rightarrow BC \quad \text{or} \quad A \rightarrow a$$

where **A, B, C are non-terminals** and **a is a terminal.**

- To be in CNF, all the productions must derive either t**wo non-terminals** or a **single terminal**.
  - **CNF restricts the number of symbols on the right side of a production to be two.**
  - **The two symbols must be non-terminals or a single terminal.**

CHOMSKY NORMAL FORM (CNF)

- A context free grammar, G = (V, T, P, S) is said to be in CNF if each production in G is of the form,

  $X \rightarrow YZ;$ $X \rightarrow \alpha$, where X, Y, Z $\in$ V and $\alpha \in$ T.

- Every CFG without null productions can be easily converted into its equivalent CNF.
- Algorithm to convert a CFG to CNF
  - Eliminate the **useless symbols**, **unit** and **null productions** from the grammar
  - Each variable should be having a derivation of length 2 or more, having only variable as of the form, $A \rightarrow \alpha$ where $|\alpha| \geq 2, \alpha \in V$.
  - The production have **three more than three variables** as derivation, **must be broken** down into a cascade of productions with derivations containing at most 2 variables.

Steps to be followed to standardize the grammar using CNF

- **Step 1:**
  - Reduce the grammar completely by-
    - Eliminating ∈ productions
    - Eliminating unit productions
    - Eliminating useless productions
- **Step 2:**
  - Replace each production of the form **$A \rightarrow B_1B_2B_3....B_n$** where n > 2 with **$A \rightarrow B_1C$** where **$C \rightarrow B_2B_3....B_n$**.
  - Repeat this step for all the productions having more than two variables on RHS.
- **Step 3:**
  - Replace each production of the form **$A \rightarrow aB$ with $A \rightarrow XB$ and $X \rightarrow a$.**
  - Repeat this step for all the productions having the form **$A \rightarrow aB$.**

PROBLEM 1

Convert the given grammar to CNF

S → aAD

A → aB / bAB

B → b

D → d

S → aAD
A → aB / bAB
B → b
D → d

- **Step 1:**

  The given grammar is already completely reduced.

- **Step 2:**
  - The productions already in chomsky normal form are-

    B → b ……..(1)

    D → d ……..(2)
  - These productions will remain as they are.
  - The productions not in chomsky normal form are-

    S → aAD ……..(3)

    A → aB / bAB ……..(4)
  - We will convert these productions in chomsky normal form.

- **Step 3:**
  - Replace the terminal symbols a and b by **new variables C_a and C_b.**
  - This is done by introducing the following two new productions in the grammar-

    $C_a \rightarrow a$ ……..(5)

    $C_b \rightarrow b$ ……..(6)
  - Now, the productions (3) and (4) modifies to-

    $S \rightarrow C_aAD$ ……..(7)

    $A \rightarrow C_aB / C_bAB$ ……..(8)

- Step 4:
  - Replace AD and AB by **new variables D1 and D2** respectively.
  - This is done by introducing the following two new productions in the grammar-

    $D_1 \rightarrow AD$ .....(9)

    $D_2 \rightarrow AB$ .....(10)
  - Now, the productions (7) and (8) modifies to-

    $S \rightarrow C_aD_1$ .....(11)

    $A \rightarrow C_aB / C_bD_2$ .....(12)

- Step 5:
  - From (1), (2), (5), (6), (9), (10), (11) and (12), **The resultant grammar is-**

    $S \rightarrow C_aD_1$

    $A \rightarrow C_aB / C_bD_2$

    $B \rightarrow b$

    $D \rightarrow d$

    $C_a \rightarrow a$

    $C_b \rightarrow b$

    $D_1 \rightarrow AD$

    $D_2 \rightarrow AB$
  - **This grammar is in chomsky normal form.**

PROBLEM 2

Convert the given grammar to CNF

S → 1A / 0B
A → 1AA / 0S / 0
B → 0BB / 1S / 1

- **Step 1:**
  - The given grammar is already completely reduced.
- **Step 2:**
  - The productions already in chomsky normal form are-

    A → 0 ……..(1)

    B → 1 ……..(2)
  - These productions will remain as they are.
  - The productions not in chomsky normal form are-

    S → 1A / 0B ……..(3)

    A → 1AA / 0S ……..(4)

    B → 0BB / 1S ……..(5)
  - We will convert these productions in chomsky normal form.

- **Step 3:**
  - Replace the terminal symbols 0 and 1 by **new variables C and D**.
  - This is done by introducing the following two new productions in the grammar-

    C → 0 ……..(6)

    D → 1 ……..(7)
  - Now, the productions (3), (4) and (5) modifies to-

    S → DA / CB ……..(8)

    A → DAA / CS ……..(9)

    B → CBB / DS ……..(10)

- **Step-04:**
  - Out of (8), (9) and (10), the productions already in Chomsky Normal Form are-

    S → DA / CB ……..(11)

    A → CS ……..(12)

    B → DS ……..(13)
  - These productions will remain as they are.
  - The productions not in chomsky normal form are-

    A → DAA ……..(14)

    B → CBB ……..(15)
  - We will convert these productions in Chomsky Normal Form.

- **Step-05:**
  - Replace AA and BB by **new variables E and F** respectively.
  - This is done by introducing the following two new productions in the grammar-

    E → AA ……..(16)

    F → BB ……..(17)
  - Now, the productions (14) and (15) modifies to-

    A → DE ……..(18)

    B → CF ……..(19)

- **Step 6:**
- From (1), (2), (6), (7), (11), (12), (13), (16), (17), (18) and (19), the resultant grammar is-

  S → DA / CB

  A → CS / DE / 0

  B → DS / CF / 1

  C → 0

  D → 1

  E → AA

  F → BB

- This grammar is in chomsky normal form.

HOME WORK

Find the CNF for the CFG, $S \rightarrow aA|bB$, $A \rightarrow aA$, $B \rightarrow bB \mid b$.

Convert the grammar given to CNF.

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

Convert the grammar $S \rightarrow AB \mid aB$, $A \rightarrow aab \mid \varepsilon$, $B \rightarrow bb|A$ into CNF.

Simplify the following grammar to its equivalent CNF. $P \Rightarrow \{ S \rightarrow AB \mid CA$, $A \rightarrow a$, $B \rightarrow BC \mid AB$, $C \rightarrow aB \mid b \}$

HOME WORK

Find the CNF equivalent to

$S \rightarrow aAD$

$A \rightarrow aB \mid bAB$

$B \rightarrow b$

$D \rightarrow d$

Convert the grammar given below to CNF

$S \rightarrow ABA$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid \varepsilon$

Convert the grammar given to CNF

$S \rightarrow 0A0 \mid 1B1 \mid BB$

$A \rightarrow C$

$B \rightarrow S \mid A$

$C \rightarrow S \mid \varepsilon$

Convert the grammar into CNF.

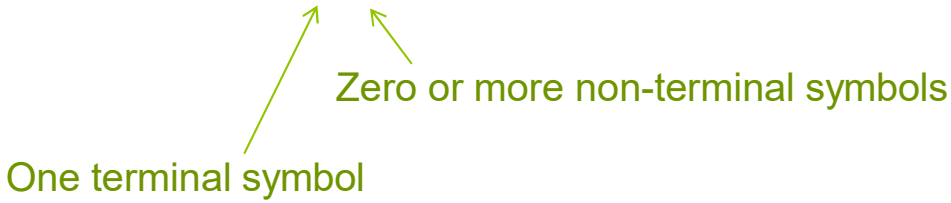
$A \rightarrow bAB \mid \varepsilon$

$B \rightarrow BAa \mid \varepsilon$

Greibach Normal Form (GNF)

- A CFG is in Greibach Normal Form (GNF), if the right-hand side of each rule has **one terminal followed by zero or more non-terminals**:

$$A \rightarrow a\ B, \text{ where } a \in T \text{ and } B \in V^*.$$



- For converting the given grammar to GNF, we use 2 lemmas.

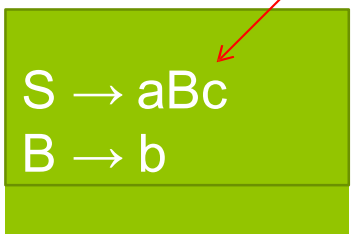
Example of a grammar in GNF

$$S \rightarrow aB \mid bA$$
$$A \rightarrow a \mid aS \mid bAA$$
$$B \rightarrow b \mid bS \mid aBB$$

Every right-hand side consists of exactly one terminal followed by zero or more non- terminals.

Example of a grammar not in GNF

Terminal at end is not allowed



Not in Greibach Normal Form

Lemma 1: Substitution Rule

Let G = {V, T, P, S} be a CFG. Let $A \rightarrow \alpha B \gamma$ be a production in P and B is

$$B \rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid \beta_4$$

The equivalent grammar can be obtained by substituting B in A.

Then the resulting grammar is

$$A \rightarrow \alpha\beta_1\gamma \mid \alpha\beta_2\gamma \mid \alpha\beta_3\gamma \mid \alpha\beta_4\gamma$$

Lemma 2: Elimination of Left recursion

Grammar of the form $A \rightarrow A\alpha \mid \beta$ is called left **recursive grammar.** To eliminate left recursion, rewrite the grammar as

$A \rightarrow \beta Z$

$Z \rightarrow \alpha Z \mid \varepsilon$

If we eliminate ε production, then we get

$A \rightarrow \beta Z \mid \beta$

$Z \rightarrow \alpha Z \mid \alpha$

We can generalize this grammar. If there is a CFG as

$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid \ldots . A\alpha_n \mid \beta_1 \mid \beta_2 \mid \beta_3 \ldots\ldots \beta_n$

The equivalent grammar without left recursion is

$A \rightarrow \beta_1 Z \mid \beta_2 Z \mid \beta_3 Z \mid \ldots \mid \beta_1 \mid \beta_2 \mid \beta_3 \quad \ldots\ldots$

$Z \rightarrow \alpha_1 Z \mid \alpha_2 Z \mid \alpha_3 Z \mid \ldots\ldots \mid \alpha_1 \mid \alpha_2 \mid \alpha_3$

Procedure for converting to GNF

1. Simplify the CFG (*i.e., eliminate null production, unit production and useless symbols*) and convert to Chomsky Normal Form (CNF).
2. Convert the rules to ascending order.
   Rename the Non Terminals as $A_1, A_2 \ldots\ldots$ with $S = A_1$.
3. For each production of the form $\boldsymbol{A_i \rightarrow A_j\alpha}$, apply the following:
   1. if $j>i$ – Leave the production as it is.
   2. if $j=i$ – Apply elimination of left recursion rule.
   3. if $j<i$ – Apply substitution rule.
4. For each production of the form $A_i \rightarrow A_j\alpha$, where $j>i$, apply the substitution lemma if A_j is in GNF, to bring A_i to GNF.

PROBLEM 1

Convert the following CFG to GNF

$S \rightarrow AA|a$

$A \rightarrow SS|b$

Solution:

Step 1: Simplify the CFG and convert to CNF

– Given grammar is in CNF form.

Step 2: Rename the variables as $S=A_1$, $A=A_2$

$A_1 \rightarrow A_2A_2 \mid a$ ….(1)

$A_2 \rightarrow A_1A_1 \mid b$ ….(2)

PROBLEM 1

- Step 3.1: In (1) $j>i$ so leave the production as it is.
- Step 3.2: In (2) $j<i$ so apply substitution rule.
  - $A_2 \rightarrow A_2A_2A_1 \mid aA_1 \mid b$ ….(new 2)
- Step 3.3: In (new 2) $j=i$ so apply elimination of **left recursion rule**.

$A_2 \rightarrow aA_1\mathbf{A_3} \mid b\mathbf{A_3} \mid aA_1 \mid b$ ….(new 2)

$\mathbf{A_3} \rightarrow A_2A_1\mathbf{A_3} \mid A_2A_1$ ….(3)

Step 3.4: In (3) $j<i$ so apply **substitution rule**.

$A_3 \rightarrow aA_1A_3A_1A_3 \mid bA_3A_1A_3 \mid aA_1A_1A_3 \mid bA_1A_3 \mid$
$aA_1A_3A_1 \mid bA_3A_1 \mid aA_1A_1 \mid bA_1$ ….(new 3)

PROBLEM 1

Step 4: Now (1) is in $A_i \rightarrow A_j\alpha$ where $j<i$ so we apply substitution rule to convert it to GNF.

$A_1 \rightarrow aA_1A_2 | bA_2 | aA_1A_3A_2 | bA_3A_2 | a$ …(new 1)

Final production rule:

$A_1 \rightarrow aA_1A_2 | bA_2 | aA_1A_3A_2 | bA_3A_2 | a$

$A_2 \rightarrow aA_1A_3 | bA_3 | aA_1 | b$

$A_3 \rightarrow aA_1A_3A_1A_3 | bA_3A_1A_3 | aA_1A_1A_3 | bA_1A_3 |$

$aA_1A_3A_1 | bA_3A_1 | aA_1A_1 | bA_1$

Now the given CFG is converted to GNF.

Home Work

Convert the following CFG to GNF

S → XA|BB
B → b|SB
X → b
A → a

Convert the following CFG to GNF

S → CA
A → a
C → aB|b

Convert the following CFG to GNF

S → AB
A → BS|b
B → SA|a

Convert the following CFG to GNF

S → ABA
A → aA|ε
B → bB|ε

Convert the following CFG to CNF

S → AB
C → AB|b
B → CA
A → a

Convert the following CFG to CNF

S → BC|a
B → AC|b
C → a
A → b

Turing Machine

- Turing machines, first invented , described by **Alan Turing in 1936-37.**
- Turing's 'Automatic machines'
- Named 'Turing machines' by Alonzo Church in a review of Turing's paper (Church 1937)
- Simple abstract computational device intended to help investigate the extent and limitations of what can be computed.
- Foundational models of computability
- Accept Recursive Enumerable Languages (generated by Type-0 Grammar).
- Universal Turing Machine, can take any other Turing Machine, expressed as a number, and execute it. This is the version that is most analogous to the modern idea of a computer.

Turing Machine - Definition

A Turing Machine is formally defined as:

$M = (Q, \sum, \Gamma, \delta, q_0, B, F)$ where

Q - Finite Set of states

Γ – Finite set of allowable tape symbols

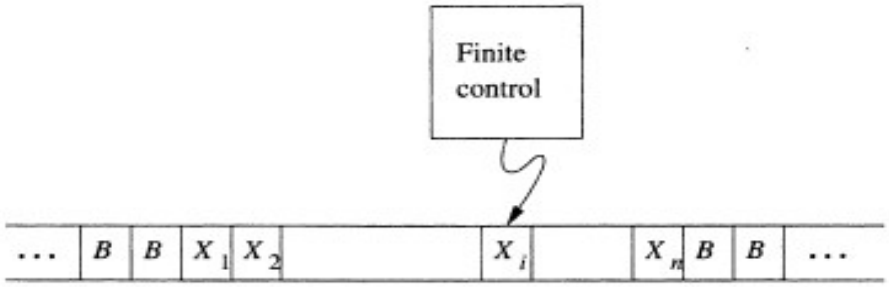
∑ - Set of input symbols, Subset of Γ excluding B

δ - Transition function which maps $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$.

q_0 in Q – Start state

B – Blank Symbol

F – Set of Final states, $F \subseteq Q$



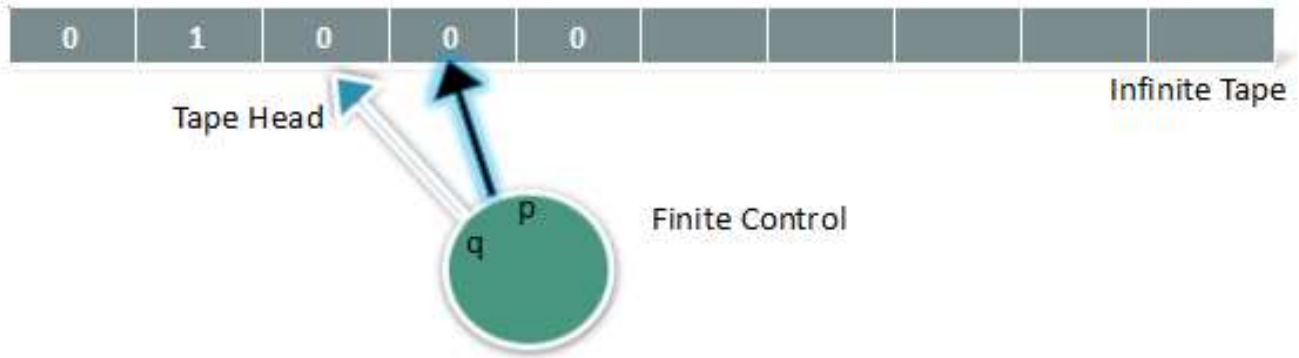
Model of TM

A **Turing Machine** (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given. ...

After reading an input symbol, it is replaced with another symbol,

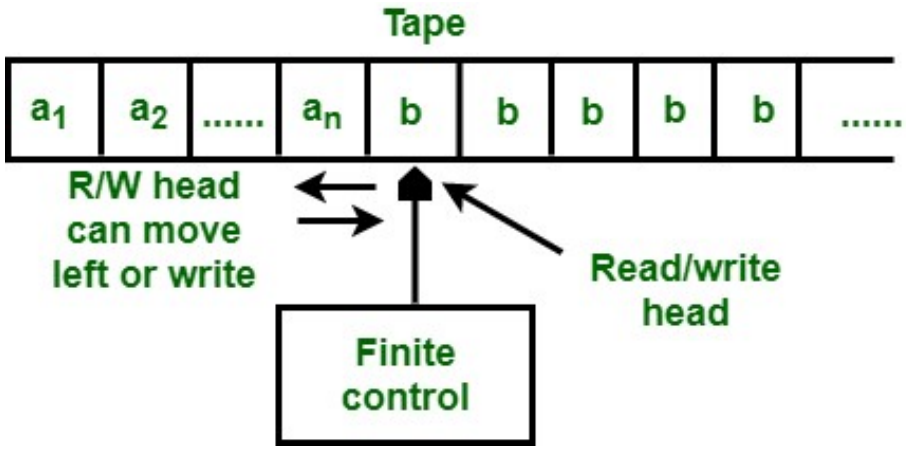
its internal state is changed,

and the tape head moves from one cell to the other either in right or to the left.



Model of TM

- The turning machine can be modelled with the help of the following representation.
- 1. The input tape is having an infinite number of cells, each cell containing one input symbol and thus the input string can be placed on tape. The empty tape is filled by blank characters.
- 2. The finite control and the tape head which is responsible for reading the current input symbol. The tape head can move to left to right.

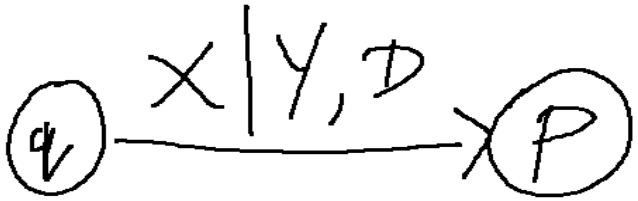


Transition Function : $\delta(q, X)=(p,Y,D)$

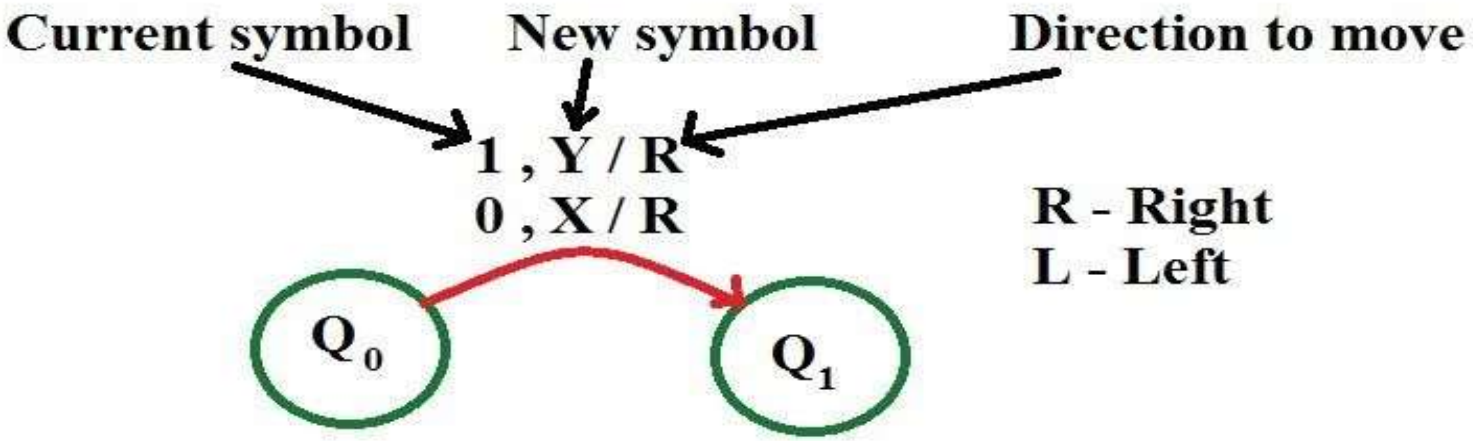
- It tells us how the machine gets one step to the next.

$$\delta(q, X)=(p,Y,D)$$

- When the machine is in a certain state q ϵ Q and the head is currently scanning X in the tape, then the machine
  1. Replaces the symbol **X by Y** on the tape
  2. Goes to state p, and
  3. The tape head moves one cell to the left or right based on D.



Transition Diagram



If current symbol is 0, replace it by X and move right

OR

If current symbol is 1, replace it by Y and move right

Instantaneous description

- The ID of a TM capture what is going out at any moment i.e. it contains all the information to exactly capture the "current state of the computations".
- It contains the following:
  - The content of the tape
  - The location of the tape head
  - The machine's internal state

$$X_1X_2\ldots\ldots\ldots\ldots X_{i-1}qX_iX_{i+1}\ldots\ldots\ldots X_n$$

Instantaneous description

$$X_1X_2\ldots\ldots\ldots\ldots X_{i-1}\mathbf{q}X_iX_{i+1}\ldots\ldots\ldots X_n$$

$$\delta(q, X_i)=(p,Y,L)$$

$$X_1X_2\ldots\ldots\ldots\ldots \mathbf{p}X_{i-1}\mathbf{Y}X_{i+1}\ldots\ldots\ldots X_n$$

$$X_1X_2\ldots\ldots\ldots\ldots X_{i-1}\mathbf{q}X_iX_{i+1}\ldots\ldots\ldots X_n$$

$$\delta(q, X_i)=(p,Y,R)$$

$$X_1X_2\ldots\ldots\ldots\ldots X_{i-1}\mathbf{Y}\mathbf{p}X_{i+1}\ldots\ldots\ldots X_n$$

Instantaneous description

Suppose $\delta(q, X_i) = (p, Y, L)$; i.e., the next move is leftward. Then

$$X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \underset{M}{\vdash} X_1 X_2 \cdots X_{i-2} p X_{i-1} Y X_{i+1} \cdots X_n$$

, suppose $\delta(q, X_i) = (p, Y, R)$; i.e., the next move is rightward. Then

$$X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \underset{M}{\vdash} X_1 X_2 \cdots X_{i-1} Y p X_{i+1} \cdots X_n$$

Language of TM

$$L(M) = \{w \mid q_0 w \xrightarrow{*} \alpha_1 p \alpha_2,\ p \in F \text{ and } \alpha_1, \alpha_2 \text{ in } \Gamma\}$$

- Let the M = <Q, Γ, b, Σ, δ, q_o, F>
- Then **L(M)** is the set of strings w in ∑* such that

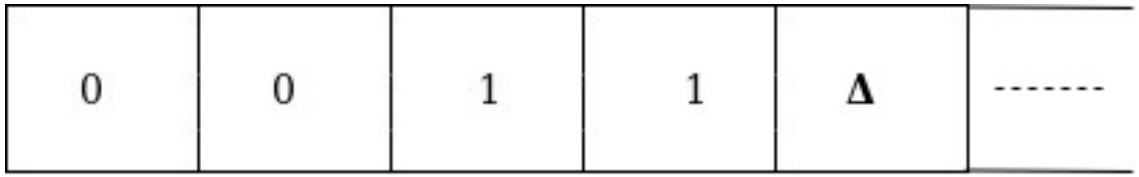
$$q_o\, w \vdash^{*} \alpha p \beta$$

For some state **p in F** and any **tape strings α and β**

Problem 1

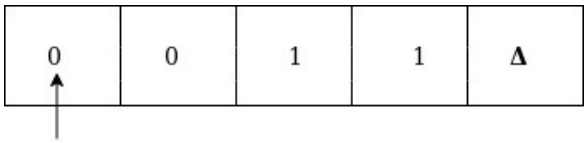
Construct TM for the language L ={0^n1^n} where n>=1.

- Now we will see how this turing machine work for 0011.
- The simulation for 0011 can be shown as below:

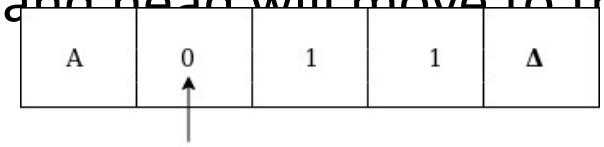


- Now, we will see how this turing machine will works for 0011.

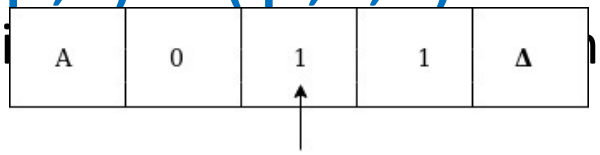
  Initially, **state is q0 and head points to 0** as:



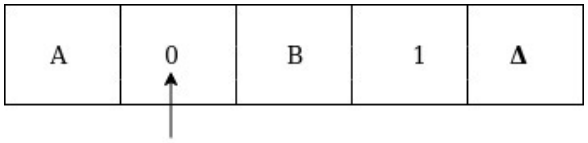
- The move will be **δ(q0, 0) = δ(q1, A, R)** which means it will go to state q1, replaced 0 by A and head will move to the right as:



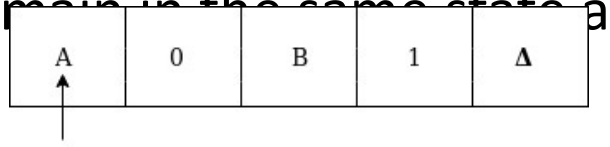
- The move will be **δ(q1, 0) = δ(q1, 0, R)** which means it will not change any symbol, remain i ove to the right as:



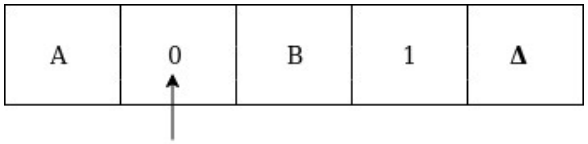
- The move will be **δ(q1, 1) = δ(q2, B, L)** which means it will go to state q2, replaced 1 by B and head will move to left as:



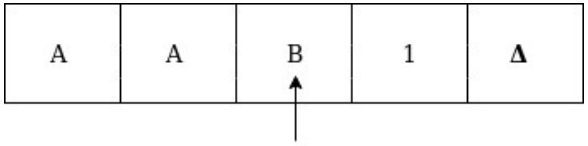
- Now move will be **δ(q2, 0) = δ(q2, 0, L)** which means it will not change any symbol, remain in the same state and move to left as:



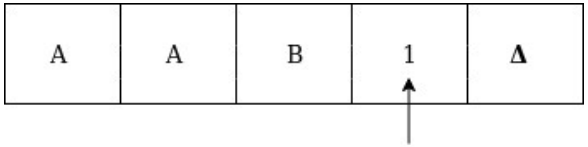
- The move will be **δ(q2, A) = δ(q0, A, R)**, it means will go to state q0, replaced A by A and head will move to the right as:



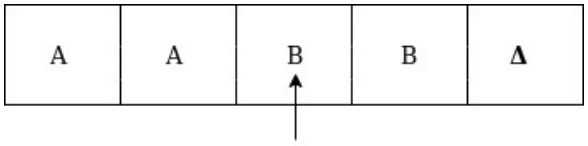
- The move will be **δ(q0, 0) = δ(q1, A, R)** which means it will go to state q1, replaced 0 by A, and head will move to right as:



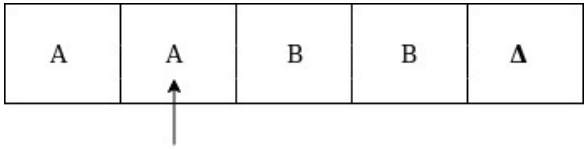
- The move will be **δ(q1, B) = δ(q1, B, R)** which means it will not change any symbol, remain in the same state and move to right as:



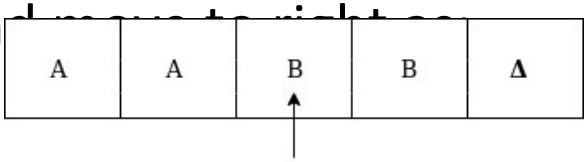
- The move will be **δ(q1, 1) = δ(q2, B, L)** which means it will go to state q2, replaced 1 by B and head will move to left as:



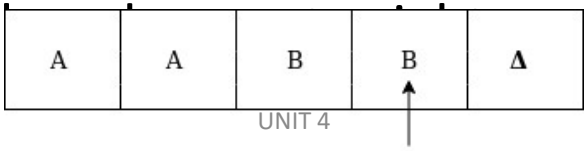
- The move **δ(q2, B) = (q2, B, L)** which means it will not change any symbol, remain in the same state and move to left as:



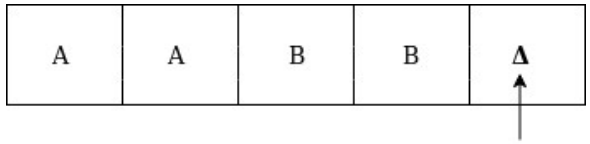
- Now immediately before B is A that means all the 0s are marked by A. So we will move right to ensure that no 1 is present. The move will be **δ(q2, A) = (q0, A, R)** which means it will go to state q0, will not change any symbol, an



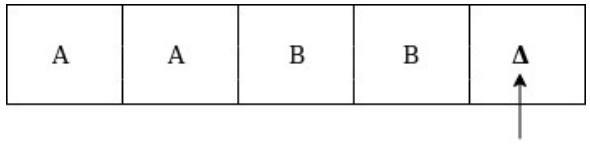
- The move **δ(q0, B) = (q3, B, R)** which means it will go to state q3, will not change any symbo

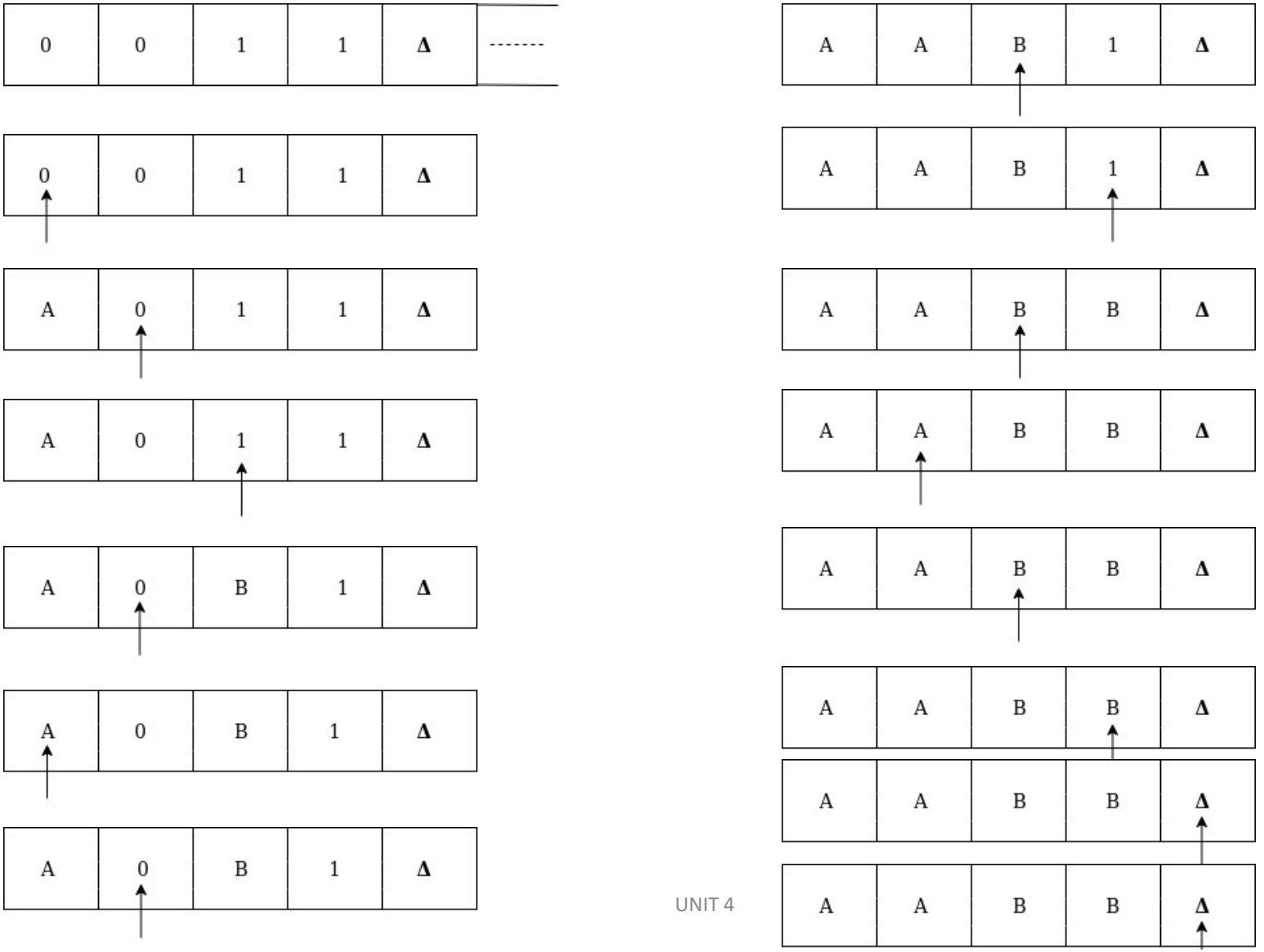


- The move **δ(q3, B) = (q3, B, R)** which means it will not change any symbol, remain in the same state and move to right as:

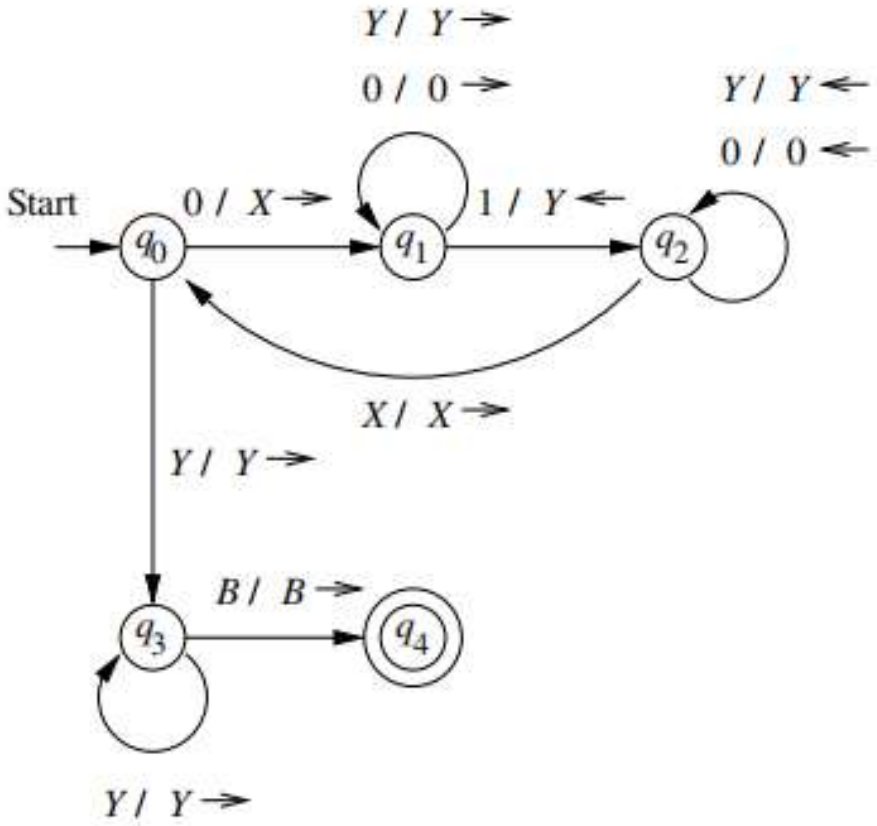


- The move **δ(q3, Δ) = (q4, Δ, R)** which means it will go to state q4 which is the HALT state and HALT state is always an accept state for any TM.




0 0 1 1 Δ -------
0 0 1 1 Δ
A 0 1 1 Δ
A 0 1 1 Δ
A 0 B 1 Δ
A 0 B 1 Δ
A 0 B 1 Δ
A A B 1 Δ
A A B 1 Δ
A A B B Δ
A A B B Δ
A A B B Δ
A A B B Δ
A A B B Δ

- The same TM can be represented by **Transition Diagram**:



Transition diagram for a TM that accepts strings of the form 0^n1^n

$q_0$0011 -- δ(q0, 0) = δ(q1, X, R)

Xq_1 0011 -- δ(q1, 0) = δ(q1, 0, R)

X0$q_1$11 -- δ(q1, 1) = δ(q2, Y, L)

X$q_2$0Y1 -- δ(q2, 0) = δ(q2, 0, L)

q_2x0Y1 -- δ(q2, X) = δ(q0, X, R)

X$q_0$0Y1 -- δ(q0, 0) = δ(q1, X, R)

XXq_1Y1 -- δ(q1, Y) = δ(q1, Y, R)

XXY$q_1$1 -- δ(q1, 1) = δ(q2, Y, L)

XXq_2YY -- δ(q2, Y) = (q2, Y, L)

Xq_2XYY -- δ(q2, X) = (q0, X, R)

XXq_0YY -- δ(q0, Y) = (q3, Y, R)

XXYq_3Y -- δ(q3, Y) = (q3, Y, R)

XXYYq_3B -- δ(q3, B) = (q4, B, R)

XXYYBq_4 --- string reached final state

Instantaneous description

Here is an example of an accepting computation by M. Its input is 0011. Initially, M is in state q_0, scanning the first 0, i.e., M's initial ID is $q_0$0011. The entire sequence of moves of M is:

$$q_0 0011 \vdash X q_1 011 \vdash X 0 q_1 11 \vdash X q_2 0 Y 1 \vdash q_2 X 0 Y 1 \vdash$$
$$X q_0 0 Y 1 \vdash X X q_1 Y 1 \vdash X X Y q_1 1 \vdash X X q_2 Y Y \vdash X q_2 X Y Y \vdash$$
$$X X q_0 Y Y \vdash X X Y q_3 Y \vdash X X Y Y q_3 B \vdash X X Y Y B q_4 B$$

For another example, consider what M does on the input 0010, which is not in the language accepted.

$$q_0 0010 \vdash X q_1 010 \vdash X 0 q_1 10 \vdash X q_2 0 Y 0 \vdash q_2 X 0 Y 0 \vdash$$
$$X q_0 0 Y 0 \vdash X X q_1 Y 0 \vdash X X Y q_1 0 \vdash X X Y 0 q_1 B$$

Transition Table

TM M is

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

where δ is given by the table in Fig. 8.9.

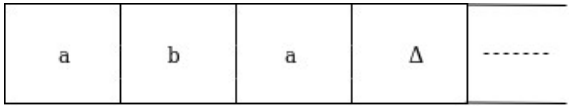
State	Symbol 0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_4, B, R)
q_4	–	–	–	–	–

Figure 8.9: A Turing machine to accept $\{0^n 1^n \mid n \geq 1\}$

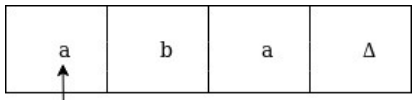
Problem 2

- **Construct a turing machine which accepts the language of “aba” over ∑ = {a, b}.**

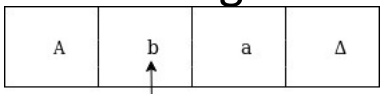
- We will assume that on input tape the string 'aba' is placed like this:



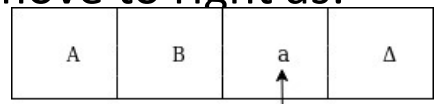
- The tape head will read out the sequence up to the Δ characters. If the tape head is readout '**aba**' string then TM will halt after reading Δ.
- Now, we will see how this turing machine will work for **aba**. Initially, state is q0 and head points to a as:



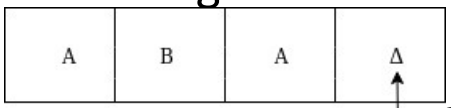
- The move will be **δ(q0, a) = δ(q1, A, R)** which means it will go to state q1, replaced a by A and head will move to right as:



- The move will be **δ(q1, b) = δ(q2, B, R)** which means it will go to state q2, replaced b by B and head will move to right as:



- The move will be **δ(q2, a) = δ(q3, A, R)** which means it will go to state q3, replaced a by A and head will move to right as:

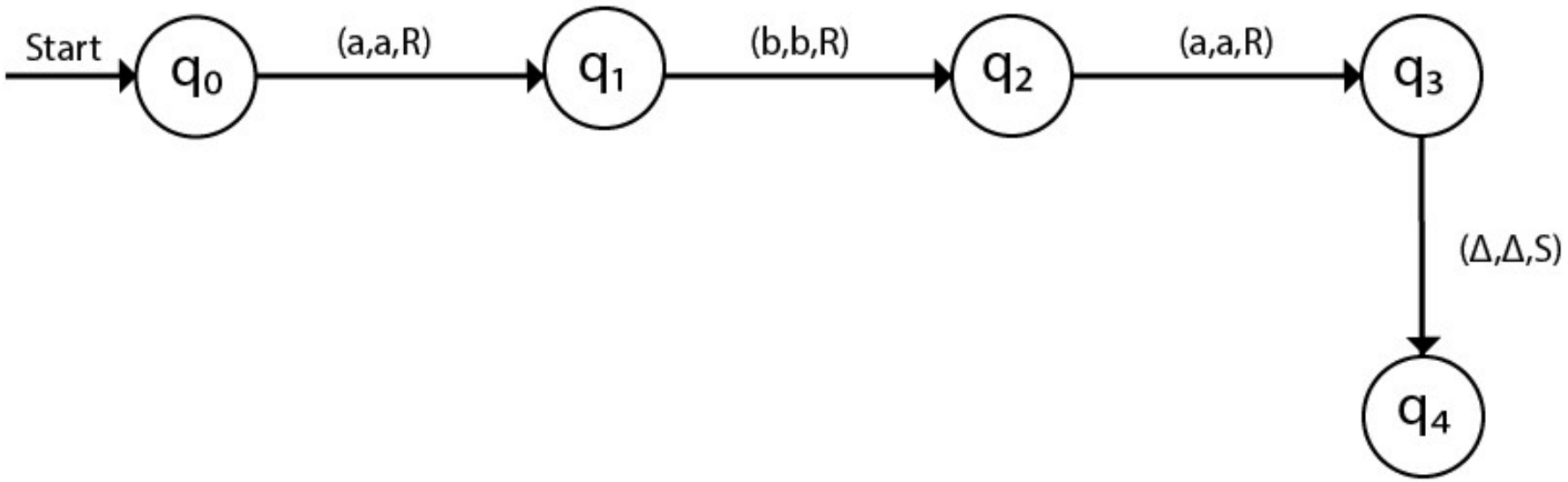


- The move **δ(q3, Δ) = (q4, Δ, S)** which means it will go to state q4 which is the HALT state and HALT state is always an accept state for any TM.

The same TM can be represented by Transition Table:

States	a	b	Δ
q0	(q1, A, R)	–	–
q1	–	(q2, B, R)	–
q2	(q3, A, R)	–	–
q3	–	–	(q4, Δ, S)
q4	–	–	–

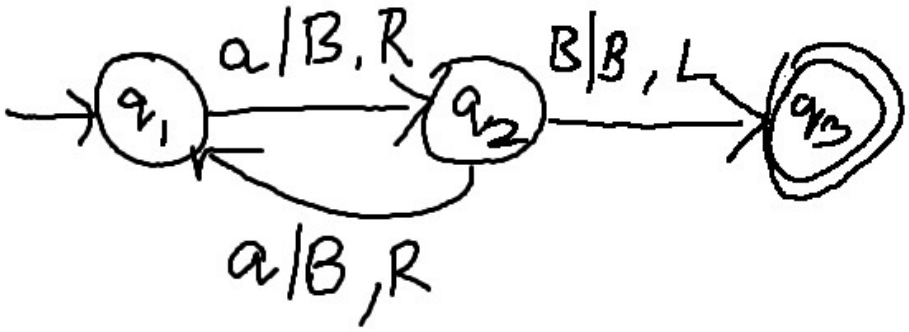
The same TM can be represented by Transition Diagram:



Problem 3

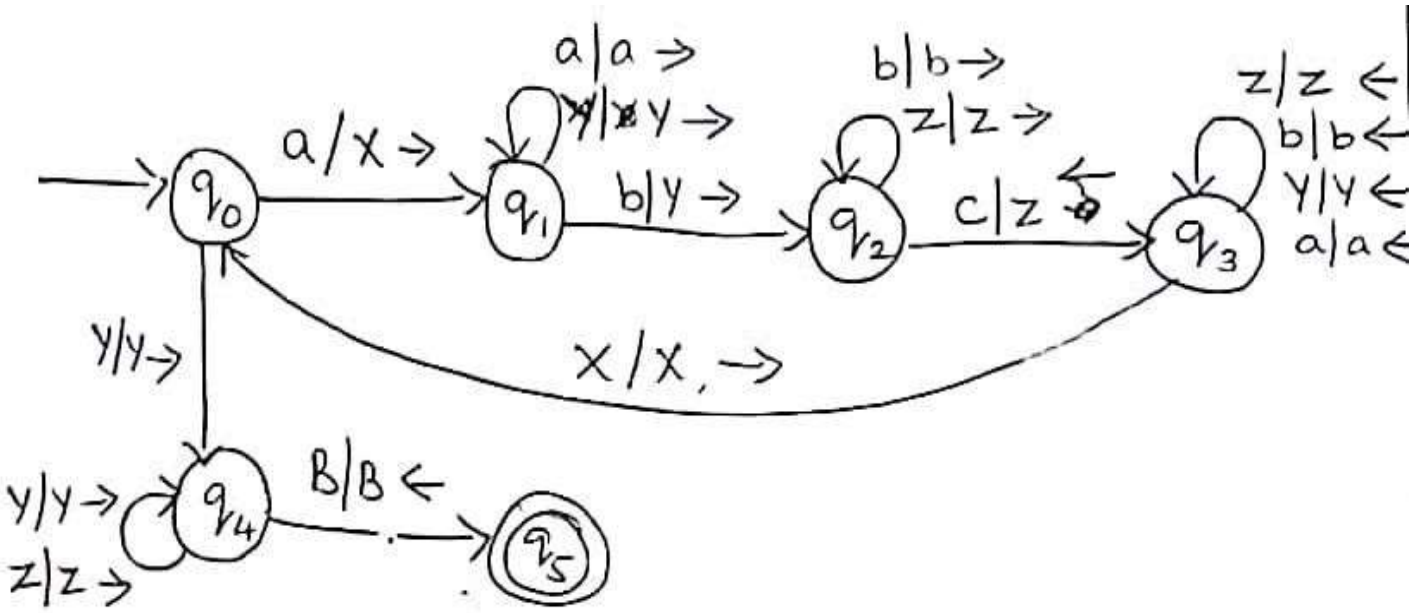
- **Design a Turing Machine that accepts the language**

  L={odd number of a's }



Problem 4

- **Design a Turing Machine that accepts the language $L=\{a^n b^n c^n \mid n \geq 1\}$**



aabbcc Xabbcc
XaYbcc XaYbZc
XXYbZc XXYYZc
XXYYZZ

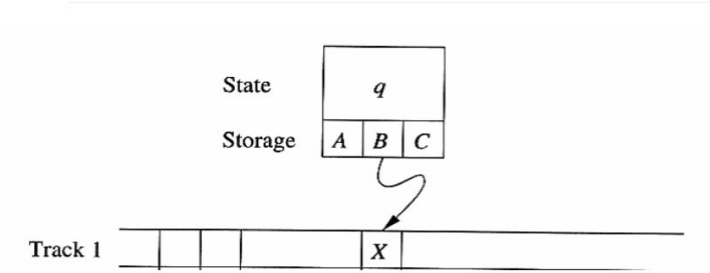
Home Work

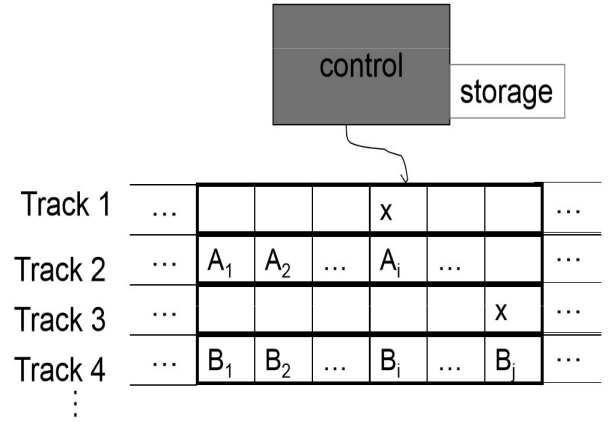
- Design a Turing Machine that accepts the language $L=\{a^n b^n c^n \mid n \geq 1\}$
- Design a Turing Machine that accepts the language L : The set of strings with an equal number of 0's and 1's.
- Construct a TM for the language $L = \{0^n 1^n 2^n\}$ where $n \geq 1$
- Design a Turing Machine that accepts the language

  $L=\{ww^R \mid w \text{ is any string of 0's and 1's}\}$

Programming Techniques for TM

1. Storage in the Finite Control (or) State
2. Multiple Tracks
3. Subroutines





Types of Turing Machine

1. Multi Head Turing Machine
2. Multi Tape Turing Machine

Halting Problem in Turing Machine