WORD PROBLEM AND GENESIS OF A FREE GROUP FOR ENGLISH ALPHABET

Raj Kishor Bisht & H.S.Dhami
Dept. Of Mathematics,
University of Kumaun,
S.S.J. Campus Almora,
Almora (Uttaranchal) 263601
INDIA

ABSTRACT

Withan aim to find applications of the elements of the fundamental groups in computerizing the group operations, an attempt has been made in the present paper to discuss the word problem in the form of finding the generators of the English alphabet. The generating set has been utilized in the genesis of free group.

1.INTRODUCTION

Group presentation form an indispensable and integral part of the many applications of group theory. The basic idea embodied is to form a group by giving a set of generators for the groups and certain equation or relations satisfied by the generators. In the theory of group presentations the role that is played in analytic geometry by a coordinate system, is played by free groups.

The word problem in groups involves the case in which a group G is generated by a finite no. of elements $a_1, a_2, \ldots\ldots\ldots\ldots a_r$ and these satisfy a finite number of relations. In the group G is it possible to decide whether or not two different word w_1 and w_2 represent the same element of G or equivalently whether or not $w = w_1 w_2^{-1}$ is the identity.

One of the most interesting questions about a group is whether its word

problem can be solved or not. The word problem in the braid group is of particular interest to topologist, algebraists and geometers. An algorithm for solving the word problem in braid groups has been investigated by Garber et al [1] . A new approach to the word and conjugacy problems in the braid groups has been put forward by Birman et al [3]. Zadrozny et al [6] have presented different aspects of mathematical linguistics in their book. Wang[2] in 1995 studied the distributional word problem for finitely presented groups.

Different chapters on models of word learning, association between verbs and the semantic categories of their arguments etc., have been compiled in the book edited by Brent [4].

2.GENERATORS OF ENGLISH ALPHABET

Let

$$A = \{a, b, c, d \ldots\ldots\ldots\ldots\ldots\ldots z\} \qquad \ldots..(2.1)$$

be any set. We can think of A as an alphabet and a, b, c etc. are letters in the alphabet. Any symbol of the form a^n with $n \in N$, $\forall\ a \in A$ is a syllable and a finite string w of syllables written in juxtaposition shall be a word.

let the elements of A satisfy the ensuing relations.

$$a = a^1, b = a^2, c = a^3, d = a^4, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots z = a^{26}$$

and $a^0 = a^{27} = *$ (identity) ……(2.2)

For the above relation we shall now define a new type of juxtaposition say "juxtaposition modulo 27". Since all the words are generated by A and satisfy the relations (2.2) therefore all the words will be of the form.

a^α, where $0 \le \alpha \le 26$ …….(2.3)

Let $w_1 = a^{\alpha_1}$ and $w_2 = a^{\alpha_2}$ ...…..(2.4)

Then juxtaposition modulo 27 can be defined as

$$w_1 \#_{27} w_2 = a^{\alpha_1} \#_{27} a^{\alpha_2} = a^{(\alpha_1 +_{27} \alpha_2)} \quad \text{.....(2.5)}$$

Now we can prove the following theorem.

The set $G = \{a, b, c, \ldots\ldots\ldots\ldots z\}$ shall be a group, composition being juxtaposition reduced modulo 27.

Proof: We have by def. (2.5)

$$a^{\alpha_1} \#_{27} a^{\alpha_2} = a^{(\alpha_1 +_{27} \alpha_2)} = a^{\alpha_3} \text{ where } o \le \alpha_3 \le 26$$

$$\forall \alpha = a^{\alpha_1} \text{ and } \beta = a^{\alpha_2} \in G$$

Thus G is closed .

Now let $\alpha = a^{\alpha_1}$, $\beta = a^{\alpha_2}$, $\gamma = a^{\alpha_3}$ be any arbitrary elements of G,

then $\alpha \#_{27} (\beta \#_{27} \gamma) = a^{\alpha_1} \#_{27} a^{(\alpha_2 +_{27} \alpha_3)}$

= least non negative exponential of $'a'$ when $\alpha_1 + (\alpha_2 + \alpha_3)$ is divided by 27.

= least non negative exponential of a when $(\alpha_1 + \alpha_2) + \alpha_3$ is divided by 27.

$= a^{(\alpha_1 +_{27} \alpha_2)} \#_{27} a^{\alpha_3}$

$= (\alpha \#_{27} \beta) \#_{27} \gamma$

This proves associativity.

We have $* \in G$.

Also if $\alpha = a^{\alpha_1} \in G$, then we have

$$* \#_{27} a^{\alpha_1} = a^0 \#_{27} a^{\alpha_1} = a^{(0 +_{27} \alpha_1)} = a^{\alpha_1}$$

Therefore $*$ is the identity element.

For existence of inverse, let us suppose that

$$\alpha = a^{\alpha_1} \in G$$

then $a^{\alpha_1} \#_{27} a^{(27 - \alpha_1)} = a^0 = *$

Therefore $a^{(27-\alpha_1)}$ is the reverse of a^{α_1}.

Hence G is a group.

Nowwe can define the relation of congruence modulo 27 in the set of words generated by A, the element of which satisfy the set of relations given by (2.2).

Let $w_1 = a^{\alpha_1}$, and $w_2 = a^{\alpha_2}$

Then w_1 will be congruent to w_2 if $(\alpha_1 - \alpha_2)$ is divisible by 27. Symbolically we can write it as

$$w_1 \equiv w_2 \pmod{27} \qquad \ldots\ldots\ldots\ldots(2.5)$$

Now we can prove the following theorem

The congruence modulo 27 is an equivalence relation in the set of words as well as this relation has 27 equivalence classes.

Proof: Let W be the set of all words, described as above, then we can prove the equivalence relation in the following manner-

Let $w_1 = a^{\alpha_1}$ be any word

then $\alpha_1 - \alpha_1 = 0$ and 27 divides 0

This implies that $w_1 \equiv w_1 \pmod{27}$

Therefore the relation is reflexive.

Now let us suppose that $w_1 = a^{\alpha_1}$, $w_2 = a^{\alpha_2}$

Such that $w_1 \equiv w_2 \pmod{27}$

which implies that 27 divides $(\alpha_2 - \alpha_1)$

Therefore $w_2 \equiv w_1 \pmod{27}$

Hence the relation is symmetric.

Let $w_1 = a^{\alpha_1}, w_2 = a^{\alpha_2}, w_3 = a^{\alpha_3} \in W$

such that $w_1 \equiv w_2 \pmod{27}$ and $w_2 \equiv w_3 \pmod{27}$;

then $[(\alpha_1 - \alpha_2) + (\alpha_2 - \alpha_3)]$ is divisible by 27

which implies that $(\alpha_1 - \alpha_3)$ is divisible by 27

Hence $w_1 \equiv w_3 \pmod{27}$

Therefore the relation is transitive.

Hence this is an equivalence relation.

Consequently it will partition W into disjoint equivalence classes.

If $w = a^{\alpha_1} \in w$ then the equivalence class $\bar{w}$ or $[w]$ can be given as

$\{ w : w = a^{\alpha} \in W$ and 27 divides $(\alpha - \alpha_1) \}$ ……..(2.6)

for $w = a$ i.e. $\alpha = 1$

$\bar{a} = \{ w : w = a^{\alpha} \in w$ and 27 divides $(\alpha - 1) \}$

=set of all the words which are reduced in 'a'

Similarly for w= a^2 or b i.e. $\alpha = 2$

$\bar{b} = \{ w : w = a^{\alpha} \in w$ and 27 divides $(\alpha - 2) \}$

= set of all the words which are reduced in 'b'

In this way we have 27 equivalence classes.

$[*], [a], [b], \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots [z]$

The set of all 27 equivalence classes is an abelian group of order 27 with respect to juxtaposition of equivalence classes.

We observe that all possible words of English language can be reduced to the form of a single letter of the alphabet as it is evident from the appended computer programme. The conversion may have interesting applications in cryptology as discussed in the workshop on Algebraic methods in Cryptography [5].

The reverse process of finding the original words from the reduced letters has been tested under certain conditions as is evident from the second part of the programme. We propose to deal with this part (in case when no constraints are there) in our subsequent studies.

3. GENESIS OF FREE GROUP FOR ENGLISH ALPHABET

Making use of the result (2.2) all the words of the English Language can be assumed to have following generating set

$$X = \{a, b, c, d, \ldots\ldots\ldots\ldots\ldots\ldots\ldots m\} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (3.1)$$

The remaining 13 letters can be taken as the inverse of these elements, that is

$$z = a^{-1}, \quad y = b^{-1}, \quad x = c^{-1}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots n = m^{-1}$$

Then the set of all reduced words formed from our alphabet X will be a free group f (x) generated by X.

Now we discuss another group G_1 generated by X satisfying ensuing relations.

If we assume X has single element 'a' such that $a^3 = 1$. Then every element of G_1 shall have the from a^{α} where $0 \leq \alpha \leq 2$.

This relation evinces that each element of G_1 can be reduced to one of the three basic expressions

$$1, \quad a, \quad a^2 \qquad \ldots\ldots\ldots\ldots\ldots\ldots (3.3)$$

Now we can justify this relation by further reducing these expressions, so that one can use this relation in a better manner.

Let we define a matrix of order 1 x 1 which can be written for 'a'

$$A = [\omega]$$

which satisfies the equation $A^3 = 1$ and the three expressions given by (3.3) generate 3 different matrices. These matrices will form a matrix group of order 3. This shows $|G_1| = 3$ and that G_1 is isomorphic to

< ω[ ] >

Now the group table of G_1 can be easily set up.

If we assume **X** has two elements a, b which satisfy the following relations.

$$a^2 = 1, \quad b^2 = 1, \quad ab = ba \qquad \text{.....(3.4)}$$

Therefore every element in G_1 which is of the form $a^{\alpha_1} b^{\beta_1} a^{\alpha_2} b^{\beta_2} \ldots\ldots\ldots\ldots\ldots\ldots a^{\alpha_k} b^{\beta_k}$, where $K \in N$ and $0 \le \alpha_i \le 1, \quad 0 \le \beta_i \le 1$

can be reduced to one of the four basic forms

$$1, a, b, ab \qquad \text{.....(3.6)}$$

For justification, let we take two matrices of order 2×2

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

which can be written for a and b and satisfy the equations.

$$A^2 = 1, \qquad B^2 = 1, \qquad AB = BA$$

The four basic expressions (3.6) give four different matrices. These matrices will form a group of order 4. This shows that $|G_1| = 4$ and that G_1 is isomorphic to

$$< \quad \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad >$$

The group table of G_1 can be easily set up.

If we assume that **X** has three elements a, b, c which satisfy the following relations.

$a^2=1,\quad b^4=1,\quad c^2=1,\quad bab=a,\quad bcb=c,\quad ac=ca$ …….(3.7)

Then every element in G_1 which is of the form

$a^{\alpha_1}b^{\beta_1}c^{\gamma_1}a^{\alpha_2}b^{\beta_2}c^{\gamma_2}\ldots\ldots\ldots\ldots\ldots a^{\alpha_k}b^{\beta_k}c^{\gamma_k}$ where $K\in N$ and $0\le\alpha_i\le 1,\quad 0\le\beta_i\le 3,\quad 0\le\gamma_i\le 1$ can be reduced to one of the 16 basic forms.

$1, a, b, c, b^2, b^3, ab, ba, bc, cb, ac, ab^2, cb^2, abc, acb, ab^2c$ .....(3.8)

For justifiacation let we take three matrices of order 3X 3

$$A=\begin{bmatrix}0&0&-i\\0&1&0\\i&0&0\end{bmatrix},\quad B=\begin{bmatrix}0&0&i\\0&-1&0\\i&0&0\end{bmatrix},\quad C=\begin{bmatrix}0&0&-i\\0&-1&0\\i&0&0\end{bmatrix}$$

Which can be written fora, b and c and satisfy the equations.

$A^2=1\quad B^4=1\quad C^2=1\quad BAB=A\ ,BCB=C,\quad AC=CA$ ……..(3.9)

The 16 basic expressions (3.8) give 16 different matrices. These matrices will from a group of order 16. This shows that $|G_1|=16$ and that G_1 is isomorphic to

$$<\begin{bmatrix}0&0&-i\\0&1&0\\i&0&0\end{bmatrix},\quad\begin{bmatrix}0&0&i\\0&-1&0\\i&0&0\end{bmatrix},\quad\begin{bmatrix}0&0&-i\\0&-1&0\\i&0&0\end{bmatrix}>$$

The group table of G_1 can be easily set up.

We observe that the group structure of G_1 is uniquely determined by the given generators and relations. We can generalize this up to 13 generators propounded by (3.1).

We propose to deal with these in our subsequent studies. Here we can

prove the following theorem of free group for three generators.

Th. If G_1 is any group generated by a set $S = \{a, b, c\}$ then G_1 is a homomorphism image of $f(x)$ and hence isomorphism to a quotient of $f(x)$.

Proof: Define $\phi : f(x) \to G_1$

By setting $\phi(*) = 1$ and

$$\phi\left(x_1^{\varepsilon_1} x_2^{\varepsilon_2} x_3^{\varepsilon_3}\right) = x_1^{\varepsilon_1} x_2^{\varepsilon_2} x_3^{\varepsilon_3}$$

it can easily prove that

$$\phi(w_1 \# w_2) = \phi(w_1) \# \phi(w_2)$$

Let $w_1 = a^2bc$ and $w_2 = abc^2$

$$\phi(w_1 \# w_2) = \phi(a^2bcabc^2)$$

$$= bcab$$

$$= \phi(a^2bc) \# \phi(abc^2)$$

hence ϕ is a well defined homomorphism. clearly ϕ is surjective because X generators G_1. hence $G_1 \cong f(x)/\ker\phi$ by first isomorphism theorem.

REFERENCES

[1] D. Garber, S. Kaplan & M. Teicher (2002), A new algorithm for solving the word problem in Braid groups, Ramat-Gan, Israel 52900.

[2] J. Wang (1995) Average -case completeness of a word problem for groups. In proceeding of the 27th Annual Symposium on Theory of Computing, ACM Press, 325-334.

[3] Joan S. Birman, Ki Hyoung ko and Sang Jin Lee (1998) A new approach to the word and conjugacy problems in the braid groups, Advances in Mathematics , 139, 322-353.

[4] Michael R. Brent (ed.) (1997) Computational approaches to language acquisition, The MIT Press ,Massachusetts Institute of Technology. Cabbridge,Massachusetts 02142

[5] Phong Nguyen (2001) The two faces of lattices in cryptology,Lecture delivered in the workshop in Algebraic methods in cryptography of the Graduate college,"Method of Mathematics and Engineering for secure data transmission and information transfer, 8.11-9.11.2001. Ruhr-Universität Bochum.

[6] W.Zadrozny, A.Manaster Ramer and M.Andrew Moshier (Eds.) (1993),"Mathematics of Language ", Annals of Mathematics and Artificial Intelligence,8, (1-2).

COMPUTER PROGRAMMES

Programme 1 :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<fstream.h>
char alpha[28] ={'*','a','b','c','d','e','f','g',
          'h','i','j','k','l','m','n',
          'o','p','q','r','s','t','u','v',
          'w','x','y','z','\0'};
int number(char a)
{
int r;
int i=0;
while(alpha[i]!='\0')
{
if(alpha[i]==a)
break;
i++;
}
return i;
}
void factor3(int i)
{
ofstream fp;
fp.open("raju.doc");
char cha[4];
cout<<endl;
for(int p=1;p<i;p++)
for(int q=1;q<i;q++)
for(int r=1;r<i;r++)
if((p+q+r)==i)
```

```
{
cout<<"("<<p<<","<<q<<","<<r<<")"<<alpha[p]<<alpha[q]<<alpha
[r]<<"\t";
cha[0]=alpha[p];
cha[1]=alpha[q];
cha[2]=alpha[r];
cha[3]='\0';
fp<<cha<<endl;
}
fp.close();
}
void main()
{
char name[100];
int l,a=0,b;
clrscr();
cout<<endl<<"Enter the word :-";
cin.getline(name,100);
l=strlen(name);
cout<<endl<<"The structure is :-";
for(int i=0;i<l;i++)
cout<<number(name[i])<<" ";
for(i=0;i<l;i++)
a=a+number(name[i]);
cout<<endl<<"The Total is :-"<<a;
b=a%27;
cout<<endl<<"The equivalance class is :-"<<alpha[b];
getch();
cout<<endl<<"The form of 3";
factor3(a);
getch();
}
```

Programme 2 :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<math.h>
char alpha[29] ={'*','a','b','c','d','e','f','g',
          'h','i','j','k','l','m','n',
          'o','p','q','r','s','t','u','v',
          'w','x','y','z','*','\0'};
int number(char a)
{
int r;
int i=0;
while(alpha[i]!='\0')
{
```

```
if(alpha[i]==a)
break;
i++;
}
return i;
}
void main()
{
char name[100],nn[100],eq;
int l,a=0,b=0,k,raju[100],m;
clrscr();
cout<<endl<<"enter no  :-";
for(int i=1; ;i++)
{
cin>>k;
if(k==0)
break;
raju[i]=k;
b+=raju[i];
}
l=i;
cout<<endl<<"Enter the equivalnace class:-";
cin>>eq;
for(i=1;i<27;i++)
if(alpha[i]==eq)
{
a=i;
break;
}
if(i==27)
a=0;
m=b/27;
for(i=0;i<m;i++)
a+=27;
if(b>a) a+=27;
raju[0]=abs(a-b);
//cout<<endl<<a<<"\t"<<b<<"\t"<<raju[0];
cout<<endl<<"The name is :-";
for(i=0;i<l;i++)
cout<<alpha[raju[i]%28];
getch();
}
/*
m=b/27;
for(i=0;i<m;i++)
a+=27;
raju[0]=abs(a-b);
cout<<endl<<a<<"\t"<<b<<"\t"<<raju[0];
cout<<endl<<"The name is :-";
for(i=0;i<l;i++)
```

```
cout<<alpha[raju[i]%28];
getch();
}
  */
```

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<fstream.h>
char alpha[28] ={'*','a','b','c','d','e','f','g',
          'h','i','j','k','l','m','n',
          'o','p','q','r','s','t','u','v',
          'w','x','y','z','\0'};
int number(char a)
{
int r;
int i=0;
while(alpha[i]!='\0')
{
if(alpha[i]==a)
break;
i++;
}
return i;
}
void factor3(int i)
{
ofstream fp;
fp.open("raju.doc");
char cha[4];
cout<<endl;
for(int p=1;p<i;p++)
for(int q=1;q<i;q++)
for(int r=1;r<i;r++)
if((p+q+r)==i)
{
cout<<"("<<p<<","<<q<<","<<r<<")"<<alpha[p]<<alpha[q]<<alpha
[r]<<"\t";
cha[0]=alpha[p];
cha[1]=alpha[q];
cha[2]=alpha[r];
cha[3]='\0';
fp<<cha<<endl;
}
fp.close();
```

```
}
void main()
{
char name[100];
int l,a=0,b;
clrscr();
cout<<endl<<"Enter the word :-";
cin.getline(name,100);
l=strlen(name);
cout<<endl<<"The structure is :-";
for(int i=0;i<l;i++)
cout<<number(name[i])<<" ";
for(i=0;i<l;i++)
a=a+number(name[i]);
cout<<endl<<"The Total is :-"<<a;
b=a%27;
cout<<endl<<"The equivalance class is :-"<<alpha[b];
getch();
cout<<endl<<"The form of 3";
factor3(a);
getch();
}
```