WORD GROUPS

SIMON NORTON
Cambridge, England
simon@dpmms.cam.ac.uk

When I read Daniel Austin's article "Can All Words Be Expressed as Sumagrams?" in the May 2004 Word Ways, I thought to myself that his results could have been obtained without needing the ability to rearrange the letters of a word. Let me discuss this in terms of the mathematical underpinning of his article.

This is what is called a free abelian group, where the second word derives from the name of the Norwegian mathematician Abel. The elements of this group are sequences of (upper case) letters and antiletters. We denote the antiletter corresponding to the letter *A*, say, by *A'*. Such sequences are called words. Let us use lower case letters to denote words; in particular, we reserve the letter *i* for the "empty word" which has no letters.

Words behave as follows: (a) two words are equal if one can be obtained from the other by rearranging the letters and antiletters, together with the creation or annihilation of letter/antiletter pairs (for example, *AEBCE' = CDAD'B*); (b) the product *a.b* of two words *a* and *b* is obtained by juxtaposing them (for example, *A.A'B = AA'B = B);* (c) the inverse *a'* of a word *a* is obtained by replacing each letter by the corresponding antiletter and vice versa.

Words obey the group laws *a(bc) = ab(c), ai = ia = a,* and *aa' = a'a = i* and the abelian group law *ab = ba.* The significance of the word "free" is that only those equalities hold which are implied by the abelian group laws.

Given a set *S* of words, we may define the group <*S*> of words generated by them as the set of words obtainable by repeatedly taking the products and inverses of words in *S* (for example, *AC'* is in <*AB,BC*> because *AC' = AB.(BC)'*. Austin's main result can be expressed as follows:

Theorem The set of number names generates all the letters contained in such names.

However, we can state a much stronger theorem in the context of what are called free groups. Like a free abelian group a free group consists of words, but these behave as follows: (a) when deciding when two words are equal, we are not allowed to rearrange their letters and antiletters and we may only create or annihilate letter/antiletter pairs if they occupy adjacent positions (for example, *ACC'B = ABD'D* but *ACBC'* is not equal to *CAC'B*); (b) the product of two words is again obtained by juxtaposing them; (c) to obtain the inverse of a word we must not only replace each letter by the corresponding antiletter and vice versa but reverse their order (for example, the inverse of *AB'* is *BA'*, not *A'B*).

It can be seen that with these definitions words obey the group laws but not the abelian group law. The group is called free because only those equalities that are consequences of the group laws hold. We can now state the following result:

Theorem In the free group on the alphabet, he set of number names generates all the letters contained in such names.

This is a stronger theorem because the product of a sequence of words is far less likely to be a given letter in the free group than in the free abelian group. However, the proof is much the same as that given by Austin:

EIGHTY and EIGHT generate Y
adding SIX and SIXTY generates T
adding TEN generates EN
adding SIXTEEN generates E and N
adding TWENTY generates W
adding ONE generates O
adding NINE generates I

We cannot generate U at this stage, but otherwise our arguments are exactly the same as Austin's. Just as he did, we can continue the argument so that:

adding ONE NONILLION generates LL
adding ONE TRILLION generates R
adding FORTY generates F
adding FOUR generates U

after which adding THREE, FIVE, SEVEN, ELEVEN, ONE MILLION, ONE BILLION, ONE QUINTILLION, ONE SEPTILLION, ONE OCTILLION, ONE DECILLION, ONE THOUSAND and ZERO give H (and G), V, S (and X), L, M, B, Q, P, C, D, A and Z, respectively.

Note that each time we are performing an operation that can be called left or right division; the left quotient of ab by a is $a'ab = b$, and similarly the right quotient of ab by b is a. In many cases that's all we need to do, but there are some cases in which we also need to do left or right multiplications (one such case will turn up later).

However, this generating set is somewhat inelegant in that we have used 25 number names to generate 24 letters, effectively wasting one of these generators. This is expressed mathematically by saying that there is a relation between the generators we have chosen; having used ONE NONILLION to generate LL, we then used ELEVEN to generate L. Can we avoid this waste?

We ask the following question: what is the most economical set of generators satisfying the following three properties: (a) they generate all the letters they contain, (b) the number of different letters they contain is equal to the number of generators, and (c) adding one number name at a time will enable us to extend the set of letters generated until they cover all the letters that occur in number names? By "economical" we mean that the number of generators is as small as possible, the numbers are as low as possible (for example, <1,2,5> is inferior to <1,3,4>), and the number of elementary operations (replacing a pair of generators $\langle a,b\rangle$ by $\langle a,ba\rangle$ or $\langle a,ab\rangle$ or $\langle a,a'b\rangle$ or $\langle a,ba'\rangle$) is as small as possible. The most economical set I have been able to obtain is <ONE, THREE, FOUR, FIVE, EIGHT, NINE, TEN, ELEVEN, FOURTEEN, EIGHTEEN, THIRTY, FORTY, ONE DECILLION, ONE DUODECILLION, ONE TREDECILLION> which generates <C,D,E, F,G,H,I,K,L,N,O,R,T,V,Y> as follows:

EIGHT and EIGHTEEN generate EIGHT and EEN
adding FOUR and FOURTEEN generates T, FOUR and EIGH
adding TEN generates EN, E, N and IGH
adding ONE generates O

adding NINE generates I and GH
adding ONE DECILLION and ONE TREDECILLION generates R, DECILL and FOU
adding THREE generates H and G
adding THIRTY generates Y
adding FORTY generates F and U
adding FIVE generates V
adding ELEVEN generates L and DEC
adding ONE DUODECILLION generates DUO, D and C.

By ordering our operations appropriately we can start with our set of number names and get the letters with just 49 elementary operations.

Let me now move to a different generating set, namely the names of the chemical elements. These too can generate all the letters contained in them, and the most economical generating set I have been able to obtain (the atomic numbers are to be kept as low as possible) is <B,Fe,Br,Y,Rh, In,Sn,Ba,Er,Tb,Yb,Ra,Th> which generates <A,B,D,E,H,I,M,N,O,R,T,U,Y> as follows:

Er and Tb generate ERBIUM and T
adding Yb generates Y
adding Y generates RIUM
adding Th generates HO
adding Sn generates IN
adding In generates DIUM
adding Rh generates R, IUM, D and ERB
adding Ra generates A
adding Ba generates B and E
adding B generates ORON
adding Fe generates (ORON)(IRON)' = OI', and therefore OI'.IN = ON, and therefore
(by removing ON and then R from ORON and IRON) O, I and hence H, N and UM
adding Br generates M and U

This required 35 elementary operations. If we replace B and Fe by I and Nb, we get a set which can be reduced to the same set of 13 letters by divisions only. However, as the atomic numbers of I and Nb are greater than those of B and Fe, this set is less economical, as well as requiring 36 elementary operations.

To extend this to the rest of the alphabet (all but Q and J):

H generates G
He generates L
C generates C
O generates X
F generates F
Na generates S
S generates P (British spelling)
V generates V
Ni generates K
Zn generates Z
Lw generates W.