Generators, Relations, and the Free Group

BSM Topology, Spring 2002

Eger David

Abstract

Groups are sometimes defined in terms of generators and relations. By generators, we mean a list of symbols $g_1, g_2, \cdots g_n$, along, implicitly, with another set of symbols $g_1^{-1}, g_2^{-1}, \cdots g_n^{-1}$. From these symols we can form finite strings which we call words. We assume that any string of the form $g_i g_i^{-1}$ or $g_i^{-1} g_i$ is equivalent to the empty word, and that any relation provided is equivalent to the empty word, which we denote 1, as it will be our group identity. We say that a word w_1 is equivalent to a word w_2 if w_1 can be transformed into w_2 in a finite number of insertions or deletions of syllables equivalent to the empty word. When we define a group by generators and relations, the elements of the group are the equivalence classes of words, and group multiplication is string concatenation. That this definition provides a group structure is trivial.

Examples of generator-relation presentations of groups:

$\left\langle g_1, g_2 : g_1 g_2 g_1^{-1} g_2^{-1} = 1 \right\rangle$ is a presentation of the group $\mathbb{Z} \times \mathbb{Z}$

$\langle g_1 : g_1 g_1 g_1 = 1 \rangle$ is a presentation of $\mathbb{Z}_3$.

The general word problem, that is, showing if two words are equivalent given an arbitrary presentation, is known to be unsolvable. However, in this paper, we show that in the very special case of the Free Group on n elements, defined in terms of generators and relations, showing whether two words are equivalent is trivial. We conclude the paper with the result that almost every Free Group is not Abelian.

For a group with the presentation

$$\langle g_1, g_2, \cdots, g_k : r_1 = 1, r_2 = 1, \cdots, r_l = 1\rangle$$

The two types of allowed transformatins of one word into an equivalent word are (1) the insertion or deletion of a relation r_i and (2) the insertion or deletion of a basic pair, that is, a syllable of the form $f_i f_i^{-1}$ or $f_j^{-1} f_j$. I will use the words reduction and deletion interchangeably, and I will use the adjective basic to refer to the insertion or deletion of basic pairs.

Let us say that a word is in reduced form if there are no occurences of $g_i g_i^{-1}$, $g_i^{-1} g_i$, or r_j in the word. In general, a word may have more than one reduced form, depending upon how we reduce it. For example, if our group is

$$\left\langle g_1, g_2 : g_1 g_2 g_1^{-1} g_2^{-1} = 1\right\rangle$$

Then we can reduce the word $g_1^{-1} g_1 g_2 g_1^{-1} g_2^{-1}$ to either $\cancel{g_1^{-1} g_1} g_2 g_1^{-1} g_2^{-1} = g_2 g_1 g_2^{-1}$ or $\cancel{g_1^{-1} g_1 g_2 g_1^{-1}} g_2^{-1} = g_2^{-1}$

Determining whether two words are equivalent is known to be unsolvale. However, in the very special case of the Free Group F_n, which we define here as having n generators and no relations, the only allowed transformations are basic transformations. I claim that any sequence of basic reductions which produces a word in reduced form will produce the same word; Therefore, we can define a function $\mathrm{Red}(w) : \Sigma^* \to \Sigma^*$ so that $\mathrm{Red}(w) \sim w$ and $\mathrm{Red}(w)$ is *the* reduced form of w. In other words, each equivalence class of words of the free group can be associated with the unique reduced word in that equivalence class.

Let us define the *left-most reduction* of a word $w = a_1 a_2 a_3 \cdots a_m$ as the reduction which is achieved by the following algorithm:

```
LeftMost(w)
    for(i=1; i < length(w); i=i+1) {
          if w(i)w(i+1) is of the form f_i f_i^{-1} or f_j^{-1} f_j
               delete w({i,i+1}); let i = 0
    }
```

In our proofs below, it is important to distinguish two notations for the word being reduced. The first notation refers to the initial composition of the word, denoted $a_1 a_2 a_3 \cdots a_m$. Therefore, if w begins as $a_1 a_2 a_3 a_4$, then after the first reduction, it may appear as $a_1 a_4$. On the other hand, $w(i)$ refers to the letter at position i, at the present moment in the algorithm. So after the first reduction, $w(2) = a_4$. For rigor, we might define $w_t(i)$, where t is how many reductions have been applied to w so far; however, if there is no case of confusion, we will retain the simpler notation above.

Since at each iteration, either i increases, or the length of w decreases by 2 and i is set to 1, this algorithm will terminate within m^2 steps. Further, after the last reduction, the algorithm ensures that the word that remains is in reduced form. At each step, we only use the allowed reduction rules, so $\texttt{LeftMost}(w) \sim w$.

Theorem If R is any sequence of basic reductions $R_1, R_2 \cdots R_r$ of a word w leaving a reduced word $R(w)$, then $R(w) = \texttt{LeftMost}(w)$

proof: Let us assume that for all words w of length l less than k, that the proposition is true. We claim, then, that for any $R = R_1, R_2 \cdots R_r$, that $R(w_k) = \texttt{LeftMost}(w_k)$. If w_k is already in reduced form, then certainly $R(w_k) = \texttt{LeftMost}(w_k) = w_k$. Otherwise, let $a_f a_{f+1}$ be the first pair in w_k deleted by R, resulting in the word $R_1(w_k)$. I claim that

$$\texttt{LeftMost}(w_k) = \texttt{LeftMost}(R_1(w_k)) = R_r(R_{r-1}(\cdots R_2(R_1(w_k)))) = R(w_k)$$

The second equality is true by the inductive hypothesis, the third by tautology. The only thing left to prove then, is that $\texttt{LeftMost}(w_k) = \texttt{LeftMost}(R_1(w_k))$.

Now $\texttt{LeftMost}(w_k)$ is a reduction of w_k consisting of a sequence of basic reductions $L_1, L_2, \cdots L_m$.

Case I: If there is an L_i which deletes $a_f a_{f+1}$ from w_k, then it doesn't matter if we "lift" L_i to the front of the list, reordering the basic reductions $L' = L_i, L_1, L_2, \cdots, L_{i-1}, L_{i+1}, \cdots L_m$. The only reasons that one reduction L_i would not be able to happen earlier than another reduction L_j are if (1) L_i and L_j delete the same element a_k, or (2) L_i depends on L_j in the sense that, if L_i deletes the letters $a_m a_n$, then L_j deletes the letters $a_r a_s$ where $m < r < s < n$. The first case is not a problem since $L_1 \cdots L_m$ each delete different letters. The second case is also not a problem since there are no letters between a_f and a_{f+1}.

Now, L' results in the same reduced word as $\texttt{LeftMost}(w_k)$ since they each consist of the same steps, that is, there is an ${L_j}' \in L'$ that deletes $a_p a_q$ from w if and only if there is an $L_k \in \texttt{LeftMost}(w_k)$ that deletes $a_p a_q$ from w. So, since ${L_1}'(w_k) = R_1(w_k) = w_{k-2}$, where $\texttt{length}(w_{k-2}) < k$ then

$$\begin{aligned}\texttt{LeftMost}(w_k) &= L'(w_k) = {L_m}'({L_{m-1}}'(\cdots {L_2}'(w_{k-2}))) \\ &= \texttt{LeftMost}(w_{k-2}) = R_r(R_{r-1}(\cdots R_2(w_{k-2}))) = R(w_k)\end{aligned}$$

Case II: There is no L_i that deletes $a_f a_{f+1}$ from w_k.

At some point, at least one of $a_f a_{f+1}$ is deleted by $\texttt{LeftMost}(w_k)$ because $\texttt{LeftMost}(w_k)$ leaves a reduced word. Therefore, the only other option is that there is an L_i which deletes the pair $a_d a_f$. However, I claim that this too, is not a problem. At time i, w looks something like $w(1)w(2)\cdots a_d a_f a_{f+1}\cdots w(l)$. Both $a_d a_f$ and $a_f a_{f+1}$ are basic pairs, which means that $a_d = a_{f+1} = a_f^{-1}$. So, deleting $a_d a_f$ or deleting $a_f a_{f+1}$ both leave $w(1)w(2)\cdots a_f^{-1}\cdots w(l)$. So without the loss of generality, we can assume that $\texttt{LeftMost}(w_k)$ does indeed delete $a_f a_{f+1}$, leaving us in the previous case. $\square$

Corollary In $F_n = \langle f_1, f_2, \ldots f_n :\rangle$, if w_1 and w_2 are reduced words with different spellings, then $w_1 \not\sim w_2$. (And therefore as group elements, $w_1 \neq w_2$)

proof: I will show that for each word w and each basic insertion or deletion p, we have $\mathrm{Red}(p(w)) = \mathrm{Red}(w)$. In the case of a deletion p_d, then $\mathrm{Red}(p_d(w))$ and $\mathrm{Red}(w)$ are two reductions of w. The case of insertion p_i is handled with the fact that p_i has an inverse $p_i^{-1} = p_{d_i}$, and since $\mathrm{Red}(w) = \mathrm{Red}(p_{d_i}(p_i(w)))$ and $\mathrm{Red}(p_i(w))$ are two reductions of $p_i(w)$. In each case, by the previous theorem equality holds.

So, if R is any sequence of basic manipulations of w_1, then by induction $\mathrm{Red}(R(w_1)) = \mathrm{Red}(w_1)$. Then if w_1 and w_2 are reduced words and there is some sequence of basic manipulations so that $R(w_1) = w_2$, then it follows that w_1 and w_2 must have the same spelling. $\square$

Corollary $F_n = \langle f_1, f_2, \ldots f_n :\rangle$ is not Abelian when $n > 1$.

proof: $f_1 f_2$ and $f_2 f_1$ are reduced words with different spellings, and therefore distinct. $\square$