ON THE WORD PROBLEM ON GROUPS

AASHIR MELHOTRA AND ARPIT MITTAL

Abstract. In this paper, we provide a proof for the fact that the word problem for groups is undecidable. Every finitely generated group G has a presentation $P = \langle S, R \rangle$, where S is a finite set of generators, and R is a set of strings, with alphabet with S or the inverse of an element in S. These strings are deemed identical to the empty string. We provide the combinatorial group theory HNN (Higman-Neumann-Neumann). We also give examples of automatic groups and discuss some applications of complexity theory to this topic.

Contents

1. Group Theory Preliminaries

We assume introductory group theory at the level of a standard first text, for example, see [Art15]. We begin with the definition of a presentation.

Definition 1.1. Let G be a group. We say a subset S of G **generates** G if every element $g \in G$ is a finite product of elements in S or their inverses.

This is an attempt to use the concept of bases in linear algebra to group theory.

Example. If $G = \mathbb{Z}_2 \times \mathbb{Z}_4$, then $x = (1, 0)$ and $y = (0, 3)$ will work as generators.

Consider a set S containing elements $x_1, \ldots x_n$ and their corresponding inverses $x_1^{-1}, x_2^{-1}, \ldots x_n^{-1}$. Also consider all finite strings of elements in S (Call this set F_S^i). In a group, there are relators that identify certain words in a string of generators. For it to be a group, it must identify xx^{-1} with the empty string for all elements $x \in S$. But there could also be other relations, such as the finite order generators have in finite groups.
We define it more formally:

Definition 1.2. A **relator** in G is a finite word in S that we identify with the empty word. Given a set of relators R, we create an equivalence relation $\sim$ on the set of all finite string in S such $x \sim y$ when $xy^{-1} \in R$

Date: December 11, 2022.

Here, y^{-1} for a word $x_1x_2\dots x_n$ is $x_n^{-1}x_{n-1}^{-1}\dots x_1$.
Let's now define residually finite groups

Definition 1.3. A group G is **residually finite** if for every non-identity element g of G, there exists a finite group H_g and a homomorphism $\phi_g : G \to H_g$ such that $\phi_g(g) \neq 1$.
Equivalently, for every element g, there is a normal subgroup N_g of G such that G/N_g is finite (or the index $[G : N_g]$ is finite, and $g \notin N_g$

2. HNN Extensions

Let $\alpha_i : H_i \to K_i$ be a family of group isomorphism between subgroups H and K of a common group G. The following extension of a group is attributed to Graham Higman, Bernhard Neumann, and Hanna Neumann (the latter two are married!)

Definition 2.1. The **HNN-extension** of G with respect to α_i is the group G' with presentation $\langle S \cup \bigcup_{i\in I} p_i | R \cup \bigcup_{i\in I}\bigcup_{h\in H_i} p_i^{-1}h^{-1}p_i\alpha(h)\rangle$

Here, the p_i's are additional variables. The added relations mean that h and it's image and conjugate with respect to the new "stable" letters p_i.
We now state Britton's lemma and it's consequences. A proof of this lemma can be found in [Cra21].

Lemma 2.2 (Britton's Lemma)**.** *A **pinch** is a part of a word in S that if of the form $p_i^{-1}vp_i, v \in H_i$ or $p_ivp_i^{-1}, v \in K_i$, where p_i is a stable variable.*
***Britton's Lemma** states that if a word w in $S \cup p_i \cup p_i^{-1}$ is equivalent to 1 and contains $p_iorp_i^{-1}$ for some i, then that word must contain a pinch.*

This means, we can detect a pinch, reduce it, and repeat. If we reach 1, then we're done. If at any point we can't detect any pinches, we conclude that the word isn't equivalent to 1. Here are additional corollaries (from now, we denote G' as the HNN extension of G).

Proposition 2.3.
- *G is embedded in G'. This means there exists an injective homomorphism $\phi : G \to G'$*
- *A **good subgroup** is a subgroup A of G such that $\phi_i(A \cap H_i) = A \cap K_i$. For every group subgroup A, consider $\bar{A} \leq G'$ be the subgroup generated by elements of A and all stable letters p_i. Then $\bar{A} = A'$, and $A' \cap G = A$*
- *In the case where $i = \{1\}$ and $H = K$ (so ϕ is the identity map), we have $p^{-1}gp = g$ for $g \in G \iff g \in H$.*

3. Modular Machines

Although they may seem as meaningless abstraction at first, *modular machines* are one of the tools used in the revered proof of the Novikov-Boone theorem.

Definition 3.1. A modular machine M contains an integer $m >$ as well as a finite class of sets of the form (a, b, c, d) where $a, b, c \in \mathbb{Z}$ such that $0 \leq a, b, < m$ and $0 \leq c < m^2$, and $d \in \{L, R\}$. Each quadruple is determined by a and b.

Since each quadruple is determined by the choices for a, b, each quadruple in the class has a different pair for (a, b). This machine computes on pairs $(x, y) \in (\mathbb{N} \cup 0) \times (\mathbb{N} \cup 0)$. The machine first establishes integers u, v, a, b such that $um + a = x$ and $vm + b = y$ (where

$0 \leq a, b < m$. Essentially, u and v are the quotients when x and y are divided by m and a and b are the remainders respectively.

If there is a quadruple in the class with $a = x$ and $b = y$, the machine sends the pair (x, y) to the (x', y') where we have

$$(x', y') = (um^2 + c, v)$$

if $d = R$. Otherwise,

$$(x', y') = (u, vm^2 + c).$$

If there is no quadruple with $a = x$ and $b = y$, the pair (a, b) is a *terminal* configuration. A computation on M is a finite sequence of pairs

$$(x, y) = (x_1, y_1) \to \cdots \to (x_n, y_n) = (x', y')$$

where (x', y') is terminal.

Example. Take a modular machine M with $m = 2$ and the class being

$$\{0, 1, 1, L\}, \{1, 0, 1, R\}.$$

Let the input be $(7, 8)$. Then, following along with the prior exposition, we have

$$u(2) + a = 7$$

and

$$v(2) + b = 8.$$

Following the namesake, we solve these modular congruences to get $(u, a, v, b) = (3, 1, 4, 0)$. There is a quadruple with $a = 1$ and $b = 0$, namely $\{1, 0, 1, R\}$. Since we have $c = 1$ and $d = R$, we send $(7, 8)$ to

$$(3(2^2) + 1, 4) = (13, 4).$$

Like before, we have $(a, b) = (1, 0)$. So, we send $(13, 4)$ to

$$(6(2^2) + 1, 2) = (25, 2).$$

Following along, we end up with the sequence

$$(13, 4) \to (25, 2) \to (49, 1).$$

The pair $(49, 1)$ is terminal as there is no set in the class with $a = 1$ and $b = 1$

Remark 3.2. A quick way to calculate the values of (a, b, u, v) for a given (x, y) is the following:

$$u = x \pmod m$$

$$v = y \pmod m$$

$$a = \frac{x - u}{m}$$

$$b = \frac{y - v}{m}.$$

Although it doesn't seem like it, modular machines can do anything that a Turing machine can.

Theorem 3.3. *For all Turing machines T, there exists a modular machine M simulating T.*

It seems that the easiest way to prove this is to explicitly construct M. This way, we can algorithmically find the equivalent modular machine for any Turing machine.

Proof. Let T have alphabet A and states Q. Let m of the modular machine be $|A \cup Q|$. Thus, we can let $A = \{1, 2, \ldots, n-1, n\}$ and $Q = \{n+1, n+2, \ldots, m-1, m\}$.

As we have our m, our task is now to create an equivalent configuration to the Turing machine. Let

$$C = a_{i_k} \ldots a_{i_1} q a b_{i_1} \ldots b_{i_l}.$$

The data we need to preserve is in chronological order the left characters, state, current character, and the right characters. To preserve the left/right characters we assign

$$u = \sum_{j=1}^{k} a_{i_j} m^{j-1}$$

$$v = \sum_{j=1}^{l} b_{i_j} m^{j-1}.$$

Thus for the Turing machine configuration C, we create two configurations of the modular machine to compensate: $(um+q, vm+a)$ and $(um+a, vm+q)$. We use the $\sim$ symbol to denote if a modular machine configuration is associated with a Turing machine configuration.

To create the class of quadruples associated with the modular machine, we assign for each transition of T in the form $qaq'a'd$ the modular machine transitions

$$(q, a, a'm + q', D), (q, a, a'm + q', d) \in M.$$

We have completed our construction of M. All that is left to be shown is that M does simulate T.

Suppose that (x, y) is a configuration of M and C, C' be configurations of T such that $(x, y) \sim C$. We then have.

Claim 3.4. If $C \to C'$, then $(x, y) \to (x', y')$ where $(x', y') \sim C'$.

Proof. Let

$$C = a_{i_k} \ldots a_{i_1} q a b_{i_1} \ldots b_{i_l}$$

and

$$C' = a_{i_k} \ldots a_{i_1} a' q' b_{i_1} \ldots b_{i_l}.$$

Thus there exists a transition quintuple

$$qaq'a'R$$

with the associated modular machine quadruples

$$(q, a, a'm + q', d), (q, a, a'm + q', d) \in M.$$

As (x, y) C, we have $(x, y) = (um + q, vm + a)$ or $(x, y) = (um + a, vm + q)$. We then have

$$(x, y) \to (um^2 + a'm + a', v).$$

All we have to do to complete the proof of the claim is show that this configuration is one of the configurations for C'. We have

$$u' = \sum_{j=1}^{k} a_{i_j} m^j + a'm^0 = um + a'$$

$$v' = \sum_{j=2}^{l} b_{i_j} m^{j-2} = m^{-1}v - m^{-1}b_{i_1}.$$

From this, we see that

$$(u'm + q', v'm + b_{i_1}) = ((um + a')m + q', m_{-1}(v - b_{i_1})m + b_{i_1})$$

which simplifies to

$$(um^2 + a'm + q', v)$$

completing the proof of the claim. ■

Claim 3.5. It is true that (x, y) is terminal $\iff$ C is terminal.

Proof. Take C to be what it was assigned in the prior proof. Then, $(x, y) = (um+q, vm+a)$ or $(x, y) = (um+a, vm+q)$. The condition for C being terminal is that there is no quintuple $qaq'a'R \in T$ which is equivalent to the condition for (x, y) being terminal which is that there do not exist quadruples $(q, a, a'm + q', d), (q, a, a'm + q', d) \in M$. ■

The theorem follows from the prior claims. ■

Definition 3.6. If M is a modular machine then M halts on the configuration (x, y) if it is sent to another configuration.

Theorem 3.7. *There exists a modular machine M such that $H_M = \{(x, y)|(x, y) \xrightarrow{M} (0, 0)\}$ is undecidable.*

A proof of this theorem can be found in [Cra21].

4. Solvability of the Word Problem

Armed with group theory jargon from the previous section, we can make our way to one of the most famous problems in the field of Combinatorical Group Theory.

Problem 4.1 (The Word Problem). Let G be a finite group with a presentation $\langle S|R\rangle$. Can we decide for some $s \in S, s = e \in G$?

One way to accomplish this is to create some algorithm to do so. The question for whether we can create this algorithm tells us about the solvability of this problem.

Definition 4.2. Let G be a finite group with presentation $\langle S|R\rangle$. Then, G has a **solvable word problem** if there exists an algorithm which can determine if an element $s \in S$ is equivalent to the trivial element in G.

An initial criticism of this problem is

Complaint 4.3. The solvability of the word problem of a group G with presentation $\langle S|R\rangle$ may be different for the choice of a different presentation.

By (fill in theorem), all presentations of a finite group are isomorphic. Hence, the solvability of the word problem for a group is independent of the choice of presentation. Thus the complaint is false and the problem is well-defined. It turns out that some types of groups do have solveable word problems. Some examples of these classes are euclidean groups, finite groups, and hyperbolic groups.

Theorem 4.4. *Free groups with finite presentation are solvable.*

Proof. Let G be a free group with finite presentation $\langle S|R\rangle$. Since G is free, $R = \varnothing$. So, all trivial words are those reducing to the empty word. Hence we propose the following algorithm to solve the Word Problem on G. This algorithm runs in finite time due to the fact that w is finite. ■

Algorithm 1 Solver of the Word Problem over G

Require: $G$is free with finite presentation
Ensure: $w \in S$
  Reduce w to w'
  if $w' = e_S$ **then**
    return true

Theorem 4.5. *Residually finite groups with finite presentation have a solvable word problem.*

Proof. Let G be a residually finite group with finite presentation $\langle S|R\rangle$. To create an algorithm to determine if $w \in S$ is equivalent to e_G, we can create two lists. We can make $L_1 =$ words equivalent to the trivial element and $L_2 =$ words equivalent to the trivial element. In our algorithm we will alternate between checking if $w \in S$ is in L_1 or in L_2. Clearly, we have

$$L_1 = \Pi_{j=1}^{n}(s_j^{-1} r_{i_j}^{\epsilon_j} s_j)$$

for $s_i \in S, r_{i_j} \in R$, and $\epsilon_i = \pm 1$. Each finite group F can be characterized by its Cayley table. We can then let L_2 be homomorphisms $\phi_i : G \to F$ that map generators in S to elements of F and elements of R to e_F. Finally, we can create an algorithm to decide this problem. ■

Algorithm 2 Solver of the Word Problem over G

Require: G is residually finite with finite presentation
Ensure: $w \in S$
  isNotFinished $\leftarrow$ true
  counter $\leftarrow 1$
  while isNotFinished **do**
    if $w = L_{1\text{counter}}$ **then**
      isNotFinished $\leftarrow$ false
      return true
    else if $\phi_{\text{counter}}(w) \neq 1$ **then**
      isNotFinished $\leftarrow$ false
      return false
      counter $\leftarrow$ counter $+ 1$

5. Undecidability of the Word Problem

Our task is to finally prove that the Word Problem is undecidable for arbitrary groups. We do this by utilizing two claims which are given and proved in the proof of the theorem below.

Theorem 5.1. *The word problem for groups in general is undecidable. In other words, there exists a group for which the word problem in undecidable.*

Let M be the corresponding machine that simulates a Turing machine for which the halting problem is undecidable. That means the the halting set $H_M = \{(a,b)|(a,b) \to (0,0)$ after a finite number of iterations of n$\}$, is undecidable. We shall prove that if a word problem for a particular finitely presented group G' is solvable, then H_M is decidable, and that will create a contradiction, which proves our claim.
Consider $G = \langle x, y, t | xy = yx$. This is the free group of rank 3, but with x and y allowed to commute.
Let $T \leq G$ be equal to $\langle G, t(a,b) | \forall a, b \in \mathbb{Z} \rangle$.
Here, $t(a,b) = x^{-a}y^{-b}tx^a y^b$.
Also define (for any $a, b, P, Q \in Z$ with $0 \leq a < P$ and $0 \leq b < Q$) the subgroup G_{ab}^{PQ} of G by $\langle G, t(a,b), x^P, y^Q \rangle$.
We turn our attention back to M. Let m be the assigned integer for M. We label the transition quadruples of M as:
$\{(a_i, b_i c_i, R) | i \in I\} \cup \{(a_j, b_j c_j, L) | j \in J\}$.
For $i \in I$, there is a canonical isomorphism ϕ_i from $G_{a_i b_i}^{MM} \to G_{c_i 0}^{M^2 1}$ by $t(a_i, b_i) \to t(c_i, 0), x^M \to x^{M^2}$ and $y^M \to y$.
A similar isomorphism ψ_j can be created for $G_{a_j b_j}^{MM} \to G_{0 c_j}^{1 M^2}$.
These isomorphisms essentially model the modular machine M (ϕ_i models an R transition, and ψ_j models an L transition).
We use these isomorphisms between subgroups of G to create an HNN-extension of G, denoted as G' (we'll skip writing formally as it's obvious from the definition given in section 2). Denote $r_i, i \in I$ and $l_j, j \in J$ as the stable variables used in the HNN extension of G into G'.
A final construction before the first lemma. Let $T_M = \langle t(\alpha, \beta) | (\alpha, \beta) \in H_M \rangle$.

Lemma 5.2. *$T_M = T'_M \cap G$ (here T'_M is generated by $t(\alpha, \beta)$ and the stable letter r_i, l_j.)*

By Proposition 2.3, it , in general, show that T_M is a *good subgroup* of G with respect to the HNN extension G'. Since ϕ_i is a bijective isomorphism, we have:
$\phi_i(T_M \cap G_{a_i b_i}^{MM}) = \phi_i(T_m) \cap \phi_i(G_{a_i b_i}^{MM}) = \phi_i(T_m) \cap G_{c_i 0}^{M^2 1}$.
We now recall that ϕ_i acts like an R transition on $t(a,b)$. This means, $\phi_i(t(a,b)) = t(a_1, b_1)$ whenever $(a,b) \to (a_1, b_1)$ via M, and thus
$t(a,b) \in T_M \iff (a,b) \in H_M \iff (a_1, b_1) \in H_M \iff t(a_1, b_1) \in T_M$.
By this, it follows from $\phi_i(t(a,b)) = t(a_1, b_1)$ that $t(a,b) \in T_M \iff t(a_1, b_1) \in T_M$, hence $\phi_i(T_M) = T_M$
A similar argument can be made to show $\psi_j(T_M) = T_M$. Hence the lemma is proven. We know prove the following:

Lemma 5.3. *$T'_M = T'$*

Here, T' is an HNN extension (a subgroup of G') that is generated by $t(a,b)$ $(a, b \in \mathbb{Z})$, and the stable letters r_i and l_j. The inclusion $T' \subseteq T'_M$ is short to prove. This is because $(0,0) \in H_M \implies t = t(0,0) \in T_M$. Hence, the generators of T' are in T'_M (namely t, r_i's and l_j's).
We now prove the other inclusion. Recall for $(a.b) \in H_M$ we have the following chain as computations using the modular machine M:
$(a,b) = (a_1, b_1) \to (a_2, b_2) \to \ldots \to (a_n, b_n) = (0,0)$
We use induction on n to show that $t(a,b) \in T'$, hence proving the equality.

For $n = 0$, we have $(a, b) = (0, 0)$, and $t(a, b) = t(0, 0) = t \in T'$.
We now assume the claim for n. That means, we know $t(a_2, b_2) \in T'$ whenever $(a_2, b_2) \to \ldots \to (0, 0)$ has length n.
We need to prove that if $(a, b) \to (a_2, b_2)$, then $t(a, b) \in T'$. Without loss of generality, assume that $(a, b) \to (a_2, b_2)$ using an transition $(a_i, b_i, c_i, R) \in M$. Then, we have u and v such that $(a, b) = (uM + a_i, vM + b_i)$ and $(a_2, b_2) = (uM^2 + c_i, v)$. We thus have:

$$\begin{aligned} t(a,b) = x^{-a}y^{-b}tx^{a}y^{b} &= x^{-uM-a_i}y^{-vM-b_i}tx^{uM+a_i}y^{vM+b_i} \\ &= x^{-uM}y^{-vM}x^{-a_i}y^{-b_i}tx^{a_i}y^{b_i}x^{uM}y^{vM} \\ &= x^{-uM}y^{-vM}t(a_i,b_i)x^{uM}y^{vM} \end{aligned}$$

Since T' is an HNN-extension of T with stable letters r_i, we have $r_i^{-1}T'r_i = T'$. Thus, if we prove that $t(a, b)$ and $t(a_2, b_2) \in T'$ are conjugates with respect to r_i, we'll be done. We thus have:

$$\begin{aligned} r_i^{-1}t(a,b)r_i = \phi_i(x^{-uM}y^{-vM}t(a_i,b_i)x^{uM}y^{vM} \\ = x^{-uM^2}y^{-v}t(c_i,0)x^{uM^2}y^{v} \\ = x^{-uM^2}y^{-v}x^{-c_i}tx^{c_i}x^{uM^2}y^{v} \\ = x^{-uM^2-c_i}y^{-v}tx^{uM^2+c_i}y^{-v} \\ = t(uM+c_i,v) = t(a_2,b_2) \end{aligned}$$

We've now proved that $t(a, b) \in T$ for all $(a, b) \in H_M$. This shows that $T'_M \subseteq T'$. With combination of the trivial $T' \subset T'_M$, we get equality $T'_M = T'$.
We now culminate the proof. Consider the HNN-extension $(G')' = \{G, k | k^{-1}hk = h, h \in T'\}$. This is the HNN-extension of G' with respect to the identity map on T'. As G' and T' is finitely presented, so is $(G')'$. According to Proposition 2.3, an element $g \in G'$ is in T' if and only if $k^{-1}gk = g$. Since $T \subset G'$, we thus have $k^{-1}t(a, b)k = t(a, b)$ for all $a, b \in \mathbb{Z}$.
From the previous lemmas, we have $T_M = T'_M \cap G$ and $T' = T'_M$; together, they yield $T_M = T' \cap G$. We have $t(a, b) \in G$ (as $T \leq G$) and $t(a, b) \in T'$ (T' is an HNN-extension of T). As $t(a, b) \in G$ for all a and b, we have $t(a, b) \in T'$ if and only if $t(a, b) \in T_M$. Hence we have:

$$k^{-1}t(a.b)k = t(a,b) \iff t(a,b) \in T' \iff t(a,b) \in T_M \iff (a,b) \in H_M$$

Suppose, for contradiction, the word problem from $(G')'$ is unsolvable. That means the set $W \subseteq (G')'$, defined by $W = \{w \in W | w = 1\}$ (here equality means equivalence), is decidable. This means that the word problem $k^{-1}t(a, b)k = t(a, b)$ is decidable for all (a, b). This implies $(a, b) \in H_M$ for all (a, b), which contradicts the unsolvability of the halting problem. Hence $(G')'$'s word problem is unsolvable.

6. Complexity Theory

We conclude this paper with a brief discussion of the applications of complexity theory to this problem. For proofs of the theorems given in this section, refer to [Hau]. We start with a new computational model: the *straight-line program.*

Definition 6.1. A straight-line program over the alphabet Σ is a context free grammar $S = (V, \Sigma, S, P)$. In this quadruple, V is the set of nonterminals, Σ is the set of terminals, S is the initial nonterminal, and P is the set of productions. Furthermore, it is required that

for each $v \in V$, there is exactly one $\alpha \in (V \cup \Sigma)^*$ such that $(v, \alpha) \in P$ and that there are no cycles in the relation $\{(v, w) \in v \times v \mid \exists \alpha \colon (v, \alpha) \in p, w \in alph(\alpha)\}$

In this definition, we use *alph* to represent the set of the individual characters of a given word. We denote the word generated by S as $val(S)$. If $v \in V$ is nonterminal, then it generates one word which we denote as $val(S, v)$. Now, note that a grammar of this type can be converted into Chomsky Normal Form in polynomial time.

Definition 6.2. The *compressible word problem* on a finite group G with a finite generating set Σ is the decision problem asking for a SLP, S, over the terminal alphabet $\tilde{\Sigma}$, if $val(S) = 1)$ is valid in G.

Shown in [Hau], we have the following theorems

Theorem 6.3. *The problem of being given a group G, automorphisms $\phi_1, \ldots, \phi_n$, and an SLP S over $\Sigma = G \cup \{t_1, t_1^{-1}, \ldots, t_n, t_n^1\}$ and asked to determine if $val(S) = 1 \in \langle G, t_1, \ldots, t_n \mid g^{t_i} = \phi_i(g) (1 \leqslant i \leqslant n, g \in G) \rangle$ is solvable in polynomial time.*

We denote the words as the t_i^j.

Theorem 6.4. *The compressible word problem on the free product $G_1 * G_2$ is polynomial-time reducible to the set $(\{0\} \times CWP(G_1)) \cup (\{1\} \times CWP(G_2))$.*

Note that in this theorem we are just taking the disjoint union of the two compressable word problems. We again refer to the interested reader who would like to see the proofs of these theorems as well as some further results relating complexity theory to HNN-Extensions to [Hau].

Acknowledgements

Both authors would like to thank Simon Rubinstein-Salzedo for teaching the course "Theory of Computation" from which the authors learned about topics in computational and complexity theory. The first author would like to thank his TA Bryan Li and the second author would like to thank his TA Lance Mathias for helpful conversations.

References

[Art15] M. Artin. *Algebra*. Pearson India Education Services Pvt. Limited, 2015.

[Cra21] Will Cravitz. An introduction to the word problem for groups. *University of Chicago REU*, 2021.

[Hau] Niko Haubold. Compressed word problems in hnn-extensions and amalgamated products. *Institut für Informatik, Universität Leipzig.*

Email address: aashir.mehrotra@pathways.in
Email address: arpit.mittal.2.71@gmail.com