AN INTRODUCTION TO THE WORD PROBLEM FOR GROUPS

WILL CRAVITZ

Abstract. As the title suggests, this paper serves as an introduction to a topic lying at the intersection of combinatorial group theory and computability theory: the word problem. We assume only a basic familiarity with group theory and some exposure to computation. The paper proceeds to introduce free groups, group presentations, Turing machines, a special class of automaton called modular machines, then culiminates with the Aanderaa-Cohen proof of the general unsolvability of the word problem.

Contents

1. Words and Free Groups

The topic of this paper is the word problem, so it should come as no surprise that we begin with a discussion of *words*. Recall that if G is a group and X a generating set of G, then any group element $g \in G$ can be expressed as a product of generators $x_i \in X$ and their inverses. This expression will take the form of something like $g = x_1^{-1}x_2x_1x_3^{-1}x_2$. It is not much of a stretch to liken X to an alphabet of letters while some product of these letters, written as a string, is a word written in the alphabet—I'm guessing this is where the following terminology comes from. We will abstract away from generating sets in the following definitions, but keep in mind that these definitions emerge quite naturally from the rules governing products of generators.

Definition 1.1. Let S be a set and let S^{-1} be a set disjoint from S with the same cardinality. It follows that there exists a bijection $S \to S^{-1}$ defined by $x \mapsto x^{-1}$. We define a *word* in S as a finite sequence consisting of symbols from $S \cup S^{-1}$, and use the following notation to represent words:

$$w = x_1^{\epsilon_1}x_2^{\epsilon_2}...x_r^{\epsilon_r},$$

Date: August 2021.

where $x_i \in S$, $\epsilon_i = \pm 1$, and $r \geq 0$. The *length* of a word is r. When $r = 0$, we call w the *empty word* and denote it by 1. Two words are equal if they consist of the same symbols in the same order—if they are the same sequence.

Example 1.2. The finite sequence $abb^{-1}a^{-1}bba$ is a word in $\{a, b\}$ with length 7.

Definition 1.3. The *product* of two words $w = x_1^{\epsilon_1}x_2^{\epsilon_2}...x_r^{\epsilon_r}$ and $v = y_1^{\eta_1}y_2^{\eta_2}...y_r^{\eta_r}$ is the word formed by concatenation, meaning

$$wv = x_1^{\epsilon_1}x_2^{\epsilon_2}...x_r^{\epsilon_r}y_1^{\eta_1}y_2^{\eta_2}...y_r^{\eta_r}.$$

Note that concatenating the empty word to any other word has no effect, so $w1 = 1w = w$.

Definition 1.4. The *inverse* of a word $w = x_1^{\epsilon_1}x_2^{\epsilon_2}...x_r^{\epsilon_r}$ is formed by inverting each symbol as well as inverting the order; thus,

$$w^{-1} = x_r^{-\epsilon_r}...x_2^{-\epsilon_2}x_1^{-\epsilon_1}.$$

Definition 1.5. A word $w = x_1^{\epsilon_1}x_2^{\epsilon_2}...x_r^{\epsilon_r}$ is *reduced* if it is the empty word or if no symbol $x_i^{\epsilon_i}$ is ever adjacent to its inverse $x_i^{-\epsilon_i}$. That is, we never see $x_ix_i^{-1}$ or $x_i^{-1}x_i$.

If w is not reduced, then it can be turned into a reduced word by deleting all instances of adjacent inverse symbols. We will denote the resulting reduced word as $\overline{w}$, and it can be checked using induction that the order of deletions does not affect $\overline{w}$.

Example 1.6. Let $w = abb^{-1}aaba^{-1}ab$ be a word in $\{a, b\}$. In order to turn w into a reduced word, we can first remove bb^{-1} to obtain $aaaba^{-1}ab$, then remove $a^{-1}a$ to get the reduced word $\overline{w} = aaabb$.

Definition 1.7. We define the *reduced product* of two words w and v as $\overline{wv}$.

Proposition 1.8. *Let F_S be the set of reduced words in S. Then F_S equipped with the reduced product forms a group.*

Proof. Our goal here is to verify that F_S satisfies the group axioms. Let $w, v, u \in F_S$.

As we already hinted at in Definition 1.3, the empty word $1 \in F_S$ serves as the identity. Because w is already reduced, then $w = \overline{w}$; thus, we have

$$\overline{w1} = \overline{1w} = \overline{w} = w.$$

The inverse of the reduced word $w = x_1^{\epsilon_1}x_2^{\epsilon_2}...x_r^{\epsilon_r}$ in F_S is the inverse $w^{-1} = x_r^{-\epsilon_r}...x_2^{-\epsilon_2}x_1^{-\epsilon_1}$ as described in Defintion 1.4. Note that when we take the reduced product

$$\overline{ww^{-1}} = \overline{x_1^{\epsilon_1}x_2^{\epsilon_2}...x_r^{\epsilon_r}x_r^{-\epsilon_r}...x_2^{-\epsilon_2}x_1^{-\epsilon_1}},$$

the reduction process deletes each pair of adjacent symbols, starting from the center with $x_r^{\epsilon_r}x_r^{-\epsilon_r}$, ending with $x_1^{\epsilon_1}x_1^{-\epsilon_1}$. We are left with the empty word, so $\overline{ww^{-1}} = 1$.

To verify that the reduced product is associative, we show $\overline{\overline{wv}u} = \overline{w\overline{vu}}$. If we started with the (potentially) unreduced word wvu, then both $\overline{\overline{wv}u}$ and $\overline{w\overline{vu}}$ represent different orders of deletions we could take in order to obtain $\overline{wvu}$. Recall that the order of deletions does not affect the resulting reduced word, which means

$$\overline{\overline{wv}u} = \overline{w\overline{vu}} = \overline{wvu}.$$

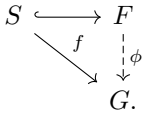
Finally, notice that it follows from the definition of a generating set that S generates F_S. □

Remark 1.9. An alternate approach to considering the group of reduced words is to instead construct a group of equivalence classes, where two words w and v are said to be equivalent if $\overline{w} = \overline{v}$. For further explanation, see Section 2.1 of Robinson [2].

We now take a brief aside from words to posit a special type of group called a *free group*. There is a sense in which free groups have minimal structure—the only relations between elements of the group are direct consequences of the group axioms. This property will make them useful for describing other groups, which we will return to when discussing group presentations.

We start by defining the universal property of free groups, then show that the group of reduced words satisfies this property. Finally, we conclude this section by presenting some other, more concrete properties of free groups.

Definition 1.10. (Universal Property of Free Groups) Let F be a group with generating set $S \subseteq F$. Then F is a free group with basis S if, for every group G and every map $f : S \to G$, there exists a unique homomorphism $\phi : F \to G$ such that $\phi = f$ on S and the following diagram commutes:



Theorem 1.11. *There exists a free group F_S with basis S.*

Proof. Let F_S be the group of reduced words in S. We now show that F_S satisfies the universal property of free groups.

Let G be a group and consider some map $f : S \to G$. Recall that $w \in F_S$ is a reduced word $x_1^{\epsilon_1} x_2^{\epsilon_2} ... x_r^{\epsilon_r}$. We define $\phi : F_S \to G$ by

$$\phi(w) = \phi(x_1^{\epsilon_1} x_2^{\epsilon_2} ... x_r^{\epsilon_r}) = f(x_1)^{\epsilon_1} f(x_2)^{\epsilon_2} ... f(x_r)^{\epsilon_r}.$$

Note that ϕ clearly extends f, because if $x \in S$, then $\phi(x) = f(x)$. If we can show that ϕ is a homomorphism, then the uniqueness property will follow—this is because if some other homomorphism $\phi' : F_S \to G$ also extends f, then ϕ and ϕ' are equal on the generating set S, which implies $\phi = \phi'$.

All that's left is to show ϕ is indeed a homomorphism. Let $w, v \in F_S$. We divide into two cases:

First, suppose that wv is already reduced, so $wv = \overline{wv}$. Letting $w = x_1^{\epsilon_1} x_2^{\epsilon_2} ... x_r^{\epsilon_r}$ and $v = y_1^{\eta_1} y_2^{\eta_2} ... y_r^{\eta_r}$, we have

$$\overline{wv} = wv = x_1^{\epsilon_1} x_2^{\epsilon_2} ... x_r^{\epsilon_r} y_1^{\eta_1} y_2^{\eta_2} ... y_r^{\eta_r},$$

so it follows that

$$\phi(\overline{wv}) = f(x_1)^{\epsilon_1} ... f(x_r)^{\epsilon_r} f(y_1)^{\eta_1} ... f(y_r)^{\eta_r} = \phi(w)\phi(v).$$

Second, we consider when wv is not reduced. If this is the case, then there is some word u such that, with $w = w'u$ and $v = u^{-1}v'$, we know $w'v'$ is reduced. To that end, $\overline{wv} = w'v'$, so we obtain

$$\phi(\overline{wv}) = \phi(w'v') = \phi(\overline{w'v'}) = \phi(w')\phi(v')$$

by the first case. We also know that both $w'u$ and $u^{-1}v'$ are reduced because they're elements of F_S, so the first case also gives us

$$\phi(w) = \phi(w'u) = \phi(\overline{w'u}) = \phi(w')\phi(u) \text{ and}$$
$$\phi(v) = \phi(u^{-1}v') = \phi(\overline{u^{-1}v'}) = \phi(u^{-1})\phi(v') = \phi(u)^{-1}\phi(v').$$

Multiplying these equalities, we get

$$\phi(w)\phi(v) = \phi(w')\phi(u)\phi(u)^{-1}\phi(v') = \phi(w')\phi(v'),$$

so $\phi(\overline{wv}) = \phi(w')\phi(v') = \phi(w)\phi(v)$. Thus, ϕ is a homomorphism, and F_S satisfies the universal property of free groups. □

We now list, without proof, a few consequences of the universal property. Hopefully, these provide a better sense of the behavior of free groups.

Proposition 1.12. *The following statements are true of the free group F.*

(i) If F has bases S and T, then $|S| = |T|$.
(ii) All $x \in F$ where $x \neq 1$ have infinite order.
(iii) For all $x, y \in F$, we have $xy \neq yx$ unless $x = u^k$ and $y = u^q$ where $u \in F$, $k, q \in \mathbb{Z}$
(iv) Every subgroup $H \leq F$ is a free group.

2. Group Presentations

In this section, I loosely follow the approach to introducing group presentations given by Clay and Margalit [4].

We start with a simple question: what is the best way to describe a group? There is, of course, the brute-force method—explicitly listing the products of any two elements in the group organized in a multiplication table. While this method is effective for small finite groups, it becomes tedious when dealing with larger, more complicated groups.

Much of the information of a group is effectively captured by an often smaller generating set, so it seems like a promising place to start. Right away, however, we run into a problem: different groups can have equivalent generating sets. To provide an example, consider the integers $\mathbb{Z}$ and the cyclic group C_n, both of which are generated by a single element, yet they clearly behave differently (one is finite and the other is infinite). Perhaps by defining constraints on the generators, we can recapture the differences. Let C_n be generated by a, then the equation $a^n = 1$ describes the cyclic, "wrapping-around" nature of the group.

We have now landed on the central idea of a group presentation: a group can be described by its generators and a set of equalities describing how the generators behave, which we term *relations*. To illustrate this notion, we say C_n has the presentation

$$\langle a \mid a^n = 1 \rangle,$$

because the generating set is a and the only relation is $a^n = 1$. I encourage the reader to justify for themselves why S_3 has the presentation

$$\langle a, b \mid a^3 = 1, b^2 = 1, ba = a^2 b \rangle.$$

Note that we could've just as easily written this last presentation as

$$\langle a, b \mid a^3, b^2, bab^{-1}a^{-2} \rangle,$$

where each relation is instead now a product of generators equal to the identity (so we can omit the explicit equation). Written this way, the specified trivial products are called *relators*. To be consistent, we will write a presentation as $\langle S \mid R \rangle$, where S is a generating set and R is a set of relators in the generators. Now we proceed to formalize the notion of a group presentation, utilizing the features of free groups.

Definition 2.1. Let G be a group and $R \subseteq G$. The *normal closure* of R is the smallest normal subgroup $N(R) \trianglelefteq G$ such that $R \subseteq N(R)$. Equivalently, we can characterize the normal closure as

$$N(R) = \bigcap_{R \subseteq N \trianglelefteq G} N.$$

Definition 2.2. A group G has presentation $\langle S \mid R \rangle$ if G is isomorphic to $F_S / N(R)$.

Here's how we think about this definition. Recall that F_S is the set of all reduced words w in S. In the quotient $F_S/N(R)$, a word w is equivalent if we add or remove any $r \in R$. Thinking of these words as products in G, this is exactly the behavior we want—given that we defined each r to be trivial. When some word w in S (viewed as a product in G) is equal to some group element $g \in G$, then we write $w =_G g$. Finally, we show that every group admits a presentation.

Theorem 2.3. *Every group G is a quotient of a free group.*

Proof. Let S be a generating set for G, and construct the free group F_S with basis S. There exists a unique homomorphism $\phi : F_S \to G$ extending the inclusion $f : S \to G$ by the universal property of free groups. Because the image $\phi(F_S) \leq G$ contains the generating set S, it must be equal to the entire group G. Thus, ϕ is a surjective homomorphism, so by the first isomorphism theorem, $G \cong F_S / \ker \phi$. □

Corollary 2.4. *Every group G admits a presentation.*

Proof. Once again, let S be a generating set for G. If we define the same ϕ from above, then $G \cong F_S / \ker \phi$. Let $R = \ker \phi$. Then R is a normal subgroup, so $R = N(R)$. Therefore, we have $G \cong F_S/N(R)$, so G has the presentation $\langle S \mid R \rangle$. □

Definition 2.5. Denote any presentation as $\langle S \mid R \rangle$. We say that a group G is

- *finitely generated* if it admits a presentation with finite S.
- *finitely related* if it admits a presentation with finite R.
- *finitely presented* if it admits a presentation with both finite S and R.

Examples 2.6. The following are examples of presentations for some familiar groups.

(i) $F_n = \langle a_1, a_2, ..., a_n \mid \rangle$
(ii) $\mathbb{Z}^2 = \langle a, b \mid aba^{-1}b^{-1} \rangle$
(iii) $C_2 \times C_2 = \langle a, b \mid a^2, b^2, aba^{-1}b^{-1} \rangle$
(iv) $D_{2n} = \langle a, b \mid a^n, b^2, abab^{-1} \rangle$
(v) $Q_8 = \langle i, j \mid i^4, j^2 i^{-2}, ijij^{-1} \rangle$

3. Solvable Word Problems

We settled on the idea of a presentation as an efficient way to describe groups. It's natural to ask what information we can gather from a group just by looking at its presentation. Most of the time, the answer will turn out to be: not much.

Historically, there are three main questions we ask about presentations, first posed together by Max Dehn in 1911 [5].

(1) *The Word Problem*: Let G be a finitely presented group given by the presentation $\langle S \mid R\rangle$. Can we decide whether a given word in S is trivial in G?
(2) *The Conjugacy Problem*: Let G be a finitely presented group given by the presentation $\langle S \mid R\rangle$. Can we decide whether two words in S are conjugate in G?
(3) *The Isomorphism Problem*: Let G and G' be finitely presented groups given by the presentations $\langle S \mid R\rangle$ and $\langle S' \mid R'\rangle$, respectively. Can we decide whether G and G' are isomorphic based on their presentations?

These problems are examples of what we call decision problems—essentially a yes/no question that can be decided by an algorithm. For now, what we mean by an algorithm is some well-defined, terminating procedure. We'll provide a far more rigorous treatment in the next section. Out of the three problems, our focus will of course be on the word problem.

Definition 3.1. Let $G = \langle S \mid R\rangle$ be a finitely presented group. We say that G has a *solvable word problem* if there exists some algorithm that, given any word w in S, can we determine whether $w =_G 1$.

Remark 3.2. A different statement of the word problem is to ask: given two words w, v in S, can we determine whether $w =_G v$ in the group G? This formulation is equivalent to the one above, for it is the same as asking whether the word $wv^{-1} =_G 1$.

Note that the word problem of a group is independent of the presentation and only depends on the group itself. This comes from the fact that there is an isomorphism between any two presentations of the same group.

We now sketch algorithms for solving the word problems of certain important classes of groups.

Example 3.3. The word problem for finitely presented free groups is solvable.

Proof. Recall that a free group F_S has presentation $\langle S \mid \emptyset\rangle$. Let w be a word in S.

The procedure for determining whether $w =_S 1$ is not very involved: reduce w to $\overline{w}$; if $\overline{w}$ is the empty word, then $w =_S 1$, otherwise $w \neq_S 1$. We can be sure that the process of reducing w to $\overline{w}$ terminates in a finite number of steps because the original word w is finite.

The intuition behind this procedure is that $\langle S \mid \emptyset\rangle$ has no relators, so the only obviously trivial word is the empty word. Two words that are reduced to the same word are equivalent when thought of as a product—it follows that all nonempty words are trivial if and only if they can be reduced to the empty word. □

Definition 3.4. A group G is *residually finite* if, for all $g \in G$ such that $g \neq 1$, there exists a normal subgroup $N \lhd G$ such that $g \notin N$ and G/N is finite. Equivalently, we could characterize this property by saying that for all $g \in G$ such that $g \neq 1$, there exists a finite group F and a homomorphism $\theta : G \to F$ such that $\theta(g) \neq 1$ in F.

Remark 3.5. Several classes of important groups turn out to be residually finite, including finite groups and finitely generated abelian groups. In fact, free groups

are residually finite, too, so the earlier procedure was somewhat irrelevant given that the following procedure is more general.

The following algorithm is given in Section 2.2 of Robinson [2].

Example 3.6. The word problem for finitely presented residually finite groups is solvable.

Proof. Let the residually finite group G have presentation $\langle S \mid R \rangle$, and let w be a word in S. Now we create two enumerable lists: one which will allow us to determine $w =_S 1$ and the other to determine $w \neq_S 1$.

First, we list all possible products of conjugates of relators: words of the form

$$(s_1^{-1} r_{i_1}^{\epsilon_1} s_1)(s_2^{-1} r_{i_2}^{\epsilon_2} s_2)...(s_j^{-1} r_{i_j}^{\epsilon_j} s_j)$$

for $r_i \in R$, $s_i \in S$, and $\epsilon_i = \pm 1$. This list L_1 will contain all words equal to the identity in G.

Second, we list all finite groups F by explicitly writing all possible multiplication tables. For each F, we then construct all maps from the generators S to the elements of F, which we can extend to homomorphisms $\theta_i : G \to F$ by specifying that all relators $r \in R$ map to $1 \in F$. The result is a list L_2 of homomorphisms from G to finite groups.

With these two lists, the procedure is simple. We alternate between L_1 and L_2, checking w against the items in each list. If w is equal to some word in L_1, then we know $w =_S 1$. If instead, for some homomorphism θ_i in L_2, we find that $\theta_i(w) \neq 1$ in F, then we know $w \neq_S 1$. This process—though it could take a while—is guaranteed to terminate at some point, telling us whether w is trivial. □

Lastly, we make note of which operations on groups preserve solvability of the word problem.

Proposition 3.7. *If G and H are finitely presented groups with solvable word problems, then their free product $G * H$ and direct product $G \times H$ have solvable word problems as well.*

4. Turing Machines and Decidability

We turn, now, to the task of formalizing the notion of an algorithm. We want to think of an algorithm as being some set of instructions that can run on a computer. However, in order for this to make any sense, we need to create a standard, ideal computer, free from any physical constraints or special circumstances. This is where the Turing machine enters, invented by Alan Turing in 1936 [6].

Definition 4.1. (Turing Machine) A *Turing machine* essentially consists of a tape head that can move along the cells of an infinitely long roll of tape (we will consider the tape as extending infinitely only to the right). The tape is initially blank, except for a finite number of cells filled in starting at the left end. At any given point, the tape head can read and write information on the cell it is currently over, then move one cell left or right.

Let us begin to mathematicize these notions, presenting one one of many equivalent formalizations. A Turing machine T consists of three major components:

(1) The *alphabet* Γ is a finite set of characters $\{s_0, s_1, s_2, ..., s_n\}$ which can be written on the tape, where s_0 is a special "blank" character.

(2) Q is a finite set of internal *states* $\{q_0, q_1, q_2, ..., q_m\}$, where q_0 is the specified "start-state". We may think of a state as providing the tape head with information independent of the cell it's currently over.
(3) The nonempty set of *transitions* T contains quintuples of the form (q, a, q', a', D), where $q, q' \in Q$, $a, a' \in \Gamma$, and $D \in \{L, R\}$. We will omit the parentheses and commas to abbreviate a transition as a string $qaq'a'D$. Transitions are instructions for the machine; we interpret the quintuple $qaq'a'D \in T$ as saying: if the tape head is in state q at a cell that reads character a, then switch to state q' and write a' on the cell, and finally move one cell in direction D (L for left and R for right). In addition, we require that a transition quintuple is uniquely determined by the first two letters so that the tape head has a single well-defined action. We express this by stating that if $qaq_1a_1D_1 \in T$ and $qaq_2a_2D_2 \in T$, then $q_1a_1D_1 = q_2a_2D_2$.

To encapsulate the current information on the tape and state, we specify a *machine configuration*

$$C = a_{i_k}...a_{i_1}qab_{i_1}...b_{i_l},$$

where $a_{i_1}, ..., a_{i_k} \in \Gamma$ are the characters starting at the left end of the tape up until the tape head, $q \in Q$ is the current state, $a \in \Gamma$ is the character under the tape head, and $b_{i_1}, ..., b_{i_l} \in \Gamma$ are the characters starting at the head with b_{i_l} being the right-most non-blank character.

Let C be a machine configuration. If there is some transition that takes C to a new configuration C', then we write $C \to C'$. For example, let

$$C = a_{i_k}...a_{i_1}qab_{i_1}...b_{i_l} \text{ and } qaq'a'R \in T.$$

Then $C \to C'$, where

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

We call C a *terminal configuration* if there does not exist a transition to take C to a new configuration. Equivalently, there does not exist a C' such that $C \to C'$. With this, we now define a *computation* of T as a finite sequence of configurations such that

$$C = C_1 \to C_2 \to ... \to C_t = \overline{C},$$

where $\overline{C}$ is a terminal configuration. We write such a sequence as $C \xrightarrow{T} \overline{C}$. In this computation, the input is C and the output is $\overline{C}$, except that we remove the state symbol from both strings.

Definition 4.2. Utilizing one of many equivalent formulations of the Church-Turing Thesis, we define a function f to be *computable* if it can be computed by a Turing machine T; that is to say, every input-output pair of f is the input-output pair of some computation on T.

Definition 4.3. Let T be a Turing machine with alphabet Γ and S be the set of all strings in Γ (or words in Γ that don't contain any symbols from Γ^{-1}). For some $s \in S$, we write that $T(s)$ *halts* if there exists some computation on T with input s. We may equivalently state that when T computes on a configuration $C = q_0s$ (where q_0 is the start-state), then there exists some terminal configuration $\overline{C}$ such that $C \xrightarrow{T} \overline{C}$.

Definition 4.4. Let $A \subseteq X$. The characteristic function of A, denoted as $1_A : X \to \{0, 1\}$, is defined by

$$1_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}.$$

Definition 4.5. A subset S of a countable set X is *decidable* if its characteristic function 1_S is computable. For some input $x \in X$, we write that $T(x)$ *accepts* if $T(x)$ *halts* and outputs $x \in S$, and $T(x)$ *rejects* if $T(x)$ *halts* and outputs $x \notin S$. So an equivalent formulation of what it means for S to be decidable is that

$$T(x) = \begin{cases} accepts, & x \in S \\ rejects, & x \in X \setminus S \end{cases}.$$

Definition 4.6. A subset S of a countable set X is *semi-decidable* if its characteristic function 1_S is only guaranteed to be computable on the domain S. This means that there exists a T such that $T(x)$ *accepts* for inputs $x \in S$, but for inputs $x \in X \setminus S$, the behavior of T is unpredictable: it's possible that $T(x)$ $rejects$ or it might not halt at all. To summarize,

$$T(x) = \begin{cases} accepts, & x \in S \\ rejects \text{ or } does\ not\ halt, & x \in X \setminus S \end{cases}.$$

Note that every decidable set is semi-decidable, but not every semi-decidable set is decidable. In addition, a set S is decidable if and only if both S and $X \setminus S$ are semi-decidable.

Remark 4.7. Most of the referenced textbooks and resources use the terms *recursive* and *recursively enumerable* in place of decidable and semi-decidable, respectively. I've opted to use the latter terms because I find them to be more suggestive of the concepts—in this context, at least.

We now state one of the landmark results in early computability theory, again courtesy of Alan Turing [6].

Theorem 4.8. *(Halting Problem) There exists a Turing machine U such that the set $H_T = \{x \in X \mid U(x)\ halts\}$ is undecidable.*

Briefly and informally, we will describe a relevant piece of notation. Let O be some mathematical object (a graph or set, for example) then $\langle O \rangle$ is a numerical coding of O that acts as a valid input for a Turing machine—what machine this is will be context dependent. If we have multiple objects $O_1, O_2, ..., O_n$, then $\langle O_1, O_2, ..., O_n \rangle$ is a valid encoding as well.

We only present a sketch of the proof here—as instructing a Turing machine to *not halt* is slightly suspect. Much of what is presented comes from Sipser [7]; for a rigorous treatment, which includes a diagonalization argument, I recommend the reader go straight to the source: Sections 4.2 and 5.1.

Proof. Let T denote an arbitrary Turing machine and s some input for T. Suppose that $A_T = \{\langle T, s \rangle \mid T(s) halts\}$ is decidable. Then there exists some Turing machine H that accepts a machine/input encoding $\langle T, s \rangle$ such that

$$H(\langle T, s \rangle) = \begin{cases} accepts, & T(s)\ halts \\ rejects, & T(s)\ does\ not\ halt \end{cases}.$$

Now we construct a new Turing machine D that tests what happens when T is run on the encoding of itself $\langle T \rangle$ inside H. We define

$$D(\langle T \rangle) = \begin{cases} halt, & H(\langle T, \langle T \rangle \rangle) \; rejects \\ does \; not \; halt, & H(\langle T, \langle T \rangle \rangle) \; accepts \end{cases},$$

and combining this with the explicit definition of H gives a more clear-cut definition for D:

$$D(\langle T \rangle) = \begin{cases} halt, & T(\langle T \rangle) \; does \; not \; halt \\ does \; not \; halt, & T(\langle T \rangle) \; halts \end{cases}.$$

The twist, however, is that we run D on itself, which yields the following:

$$D(\langle D \rangle) = \begin{cases} halt, & D(\langle D \rangle) \; does \; not \; halt \\ does \; not \; halt, & D(\langle D \rangle) \; halts \end{cases}.$$

The obvious contradiction here is that D is told to output the opposite of whatever it outputs. Therefore, $A_T = \{\langle T, s \rangle \mid T(s) \; halts\}$ is undecidable.

Finally, we construct U, the *universal* Turing machine. The way U operates is that it takes in a machine/input encoding $\langle T, s \rangle$ and outputs whatever T outputs on the input s, so $U(\langle T, s \rangle) \; halts$ if and only if $T(s) \; halts$. We let X be the set of all machine/input encodings $\{\langle T, s \rangle\}$. Then it follows directly from the fact that A_T is undecidable that $H_T = \{x \in X \mid U(x) \; halts\}$ is undecidable. □

5. Modular Machines

Given the definition of the Turing machine, we can gain some intuition as to why and how it can perform useful computation. The following automata, the modular machine, does not share this trait. It is not useful on an intuitive level; rather, it will act as a useful tool for certain group theoretic proofs. We can think of modular machines as merely wrapping around or encoding a Turing machine, providing a new numerical form to describe computation. We prove, in this section, that any Turing machine can be simulated/encoded by a modular machine.

Definition 5.1. (Modular Machine) A *modular machine* M consists of an integer $m > 1$ along with a finite set of quadruples of the form (a, b, c, D) where $a, b, c \in \mathbb{Z}$ such that $0 \leq a, b < m$ and $0 \leq c \leq m^2$, and D is either L or R. Each quadruple is uniquely defined by a and b, so if (a, b, c, D) and (a, b, c', D') are both in M, then it must be the case that $c = c'$ and $D = D'$.

Now we turn to how M computes. A configuration of M is a pair of nonnegative integers $(\alpha, \beta) \in (\mathbb{N} \cup \{0\})^2$. To decide what configuration (α, β) updates to, we assign values to the letters u, v, a, b in the following way: let $um + a = \alpha$ and $vm + b = \beta$ where $0 \leq a, b < m$, perhaps better thought of as $a = \alpha \mod m$ and $b = \beta \mod m$. If there is a quadruple $(a, b, c, D) \in M$ whose first two numbers are a and b, then we calculate the new configuration (α', β') as follows:

(1) If $D = R$, then let $\alpha' = um^2 + c$ and $\beta' = v$.
(2) If $D = L$, then let $\alpha' = u$ and $\beta' = vm^2 + c$.

We denote this transition to a new configuration as $(\alpha, \beta) \to (\alpha', \beta')$. If, on the other hand, there does not exists a quadruple beginning with a and b, then we say

that (α, β) is a terminal configuration. Similar to Turing machines, a computation on M is a finite sequence of configurations such that

$$(\alpha, \beta) = (\alpha_1, \beta_1) \to (\alpha_2, \beta_2) \to ... \to (\alpha_t, \beta_t) = (\overline{\alpha}, \overline{\beta}),$$

where $(\overline{\alpha}, \overline{\beta})$ is a terminal configuration. We write such a sequence as $(\alpha, \beta) \xrightarrow{M} (\overline{\alpha}, \overline{\beta})$.

Example 5.2. Let M be the modular machine with $m = 6$ and quadruples

$$\{(3, 2, 7, L), (2, 1, 3, R), (0, 5, 2, R)\}.$$

We will run M on the configuration $(8, 13)$. First, we let

$$8 = um + a = 1(6) + 2 \text{ and } 13 = vm + b = 2(6) + 1,$$

which leaves us with $u = 1, v = 2, a = 2$, and $b = 1$. These values of a and b correspond to the quadruple $(2, 1, 3, R)$, so we get $c = 3$ and $D = R$. Because $D = R$, we let

$$\alpha_2 = um^2 + c = 1(6^2) + 3 = 39 \text{ and } \beta_2 = v = 2,$$

giving us the first transition $(8, 13) \to (39, 2)$. I encourage the reader to follow the same process and complete the computation, which will yield the sequence

$$(8, 13) \to (39, 2) \to (6, 11) \to (38, 1) \to (219, 0).$$

Note that the configuration $(219, 0)$ is terminal because it results in $a = 3$ and $b = 0$, for which there is no corresponding quadruple.

Theorem 5.3. *Given any Turing machine T, there exists a modular machine M that simulates T, by which we mean any computation on T corresponds to an equivalent computation on M.*

Proof. We provide a proof by explicit construction of the modular machine M. Let T be a Turing machine with alphabet Γ and states Q. We let m of the modular machine be equal to the cardinality of $\Gamma \cup Q$, and we may safely assert without loss of generality that $\Gamma = \{1, ..., n\}$ and $Q = \{n + 1, ...m\}$.

Next, we describe how to associate (mm) configurations of M with (tm) configurations of T. Let $C = a_{i_k}...a_{i_1}qab_{i_1}...b_{i_l}$ be a configuration of T. There are essentially four pieces of information we need to capture: the left characters $a_{i_k}...a_{i_1}$, the right characters $b_{i_1}...b_{i_l}$, the current character a, and the state q. We assign the left and right characters to symbols u and v, respectively, through summations:

$$u = \sum_{j=1}^{k} a_{i_j} m^{j-1} \text{ and } v = \sum_{j=1}^{l} b_{i_j} m^{j-1}.$$

We create two mm configurations $(um+q, vm+a)$ and $(um+a, vm+q)$ to associate with C. If an mm configuration (α, β) is associated with a tm configuration C, then we write $(\alpha, \beta) \sim C$.

Finally, we assign (mm) transition quadruples of M with (tm) transitions quintuple of T. Let $qaq'a'D \in T$ be a tm transition, then we associate it with two mm transitions $(q, a, a'm + q', D), (q, a, a'm + q', D) \in M$.

Having made the necessary associations, we prove M does in fact simulate T by showing two facts. Let (α, β) be an mm configuration and C, C' be tm configurations such that $(\alpha, \beta) \sim C$.

(1) If $C \to C'$, then $(\alpha, \beta) \to (\alpha', \beta')$ where (α', β') is an mm configuration such that $(\alpha', \beta') \sim C'$.
(2) C is terminal if and only if (α, β) is terminal.

To prove the first claim, let $C = a_{i_k}...a_{i_1}qab_{i_1}...b_{i_l}$ and $C' = a_{i_k}...a_{i_1}a'q'b_{i_1}...b_{i_l}$, which implies the existence of a tm transition quintuple $qaq'a'R \in T$. There must be an associated pair of mm transition quadruples, $(q, a, a'm + q', D), (q, a, a'm + q', D) \in M$. Because $(\alpha, \beta) \sim C$, we know (α, β) is equal to $(um + q, vm + a)$ or $(um + a, vm + q)$. Either way, by the computation rules of M, we get that

$$(\alpha, \beta) \to (um^2 + a'm + a', v).$$

Let us check that $(um^2 + a'm + a', v) \sim C'$ as we expect. We construct the mm configurations associated with C' by first calculating

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

and

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

We plug these values into one of the mm configuration formulas $(u'm+q', v'm+b_{i_1})$ to get

$$((um + a')m + q', m^{-1}(v - b_{i_1})m + b_{i_1}) = (um^2 + a'm + q', v),$$

which is indeed the same configuration implied by the computation of M, thereby proving the first claim. Note that an analogous procedure can be followed if the tm transition instead specified a move $D = L$ to the left.

We turn our attention to the second claim. Once again, let $C = a_{i_k}...a_{i_1}qab_{i_1}...b_{i_l}$. Then because $(\alpha, \beta) \sim C$, it must be the case that (α, β) is equal to $(um+q, vm+a)$ or $(um + a, vm + q)$. In order for C to be terminal, there must not exist a tm transition quintuple $qaq'a'R \in T$, which is true if and only if there do not exist mm transition quadruples $(q, a, a'm + q', D), (q, a, a'm + q', D) \in M$. Thus, (α, β) is terminal as well, proving our second claim.

From these two facts, it follows that any computation on T is associated with an equivalent computation on M.

□

Note that we can apply an analogous definition of halting for modular machines as for Turing machines: for an input (α, β), we write $M(\alpha, \beta)$ *halts* if there exists some $(\overline{\alpha}, \overline{\beta})$ such that $(\alpha, \beta) \to (\overline{\alpha}, \overline{\beta})$. Now we obtain the following corollary as a direct consequence of the Halting Problem and the fact that modular machines can simulate Turing machines.

Corollary 5.4. *There exists a modular machine M such that the set $H_M = \{(\alpha, \beta) \mid (\alpha, \beta) \xrightarrow{M} (0, 0)\}$ is undecidable.*

Proof. Let T be a Turing machine for which the set H_T is undecidable. Now we construct a modular machine M that simulates T. Because there is a perfect correspondence between computations on T and computations on M, for associated input configurations $q_0 s \sim (\alpha, \beta)$, we know $T(s)$ *halts* if and only if $M(\alpha, \beta)$ *halts*.

Whenever $M(\alpha, \beta) halts$, then M has reached some terminal configuration $(\overline{\alpha}, \overline{\beta})$. We modify M in the following way: add a finite number of new transitions $(a_i, b_i, c_i, D) \in M$ for $0 < i \leq n$ such that

$$(\overline{\alpha}, \overline{\beta}) = (\alpha_1, \beta_1) \to ... \to (\alpha_n, \beta_n) = (0, 0).$$

Note that any M which simulates a T will never contain a transition $(0, 0, c, D) \in M$, so $(0, 0)$ will always be terminal. The upshot is that $M(\alpha, \beta)$ *halts* is now equivalent to $(\alpha, \beta) \xrightarrow{M} (0, 0)$, so $T(s)$ *halts* if and only if $(\alpha, \beta) \xrightarrow{M} (0, 0)$. Thus, the fact that $H_T = \{s \in S \mid T(s) \; halts\}$ is undecidable immediately implies $H_M = \{(\alpha, \beta) \mid (\alpha, \beta) \xrightarrow{M} (0, 0)\}$ is undecidable as well. □

6. HNN Extensions and Britton's Lemma

The final tools we will need to prove the general unsolvability of the word problem are HNN extensions and Britton's Lemma. We will view them as just that—tools—and nothing more. To that end, we provide the following statements with little motivation or proof, as additional knowledge of free groups and free products is necessary. If the reader seeks to gain a deeper understanding, I recommend them to Section 2.3 of Simpson [11], where these definitions are taken from.

Theorem 6.1. *(Higman/Neumann/Neumann) Let G be a group. Let H, K be subgroups of G which are isomorphic to each other, and let $\phi : H \to K$ be a particular isomorphism of H onto K. Then there exists a group $G' \supseteq G$ and a group element $p \in G'$ such that $p^{-1}hp = \phi(h)$ for all $h \in H$.*

Definition 6.2. (HNN Extension). Let G be any group, and let $\phi_i : H_i \cong K_i$, $i \in I$, be a family of isomorphisms between subgroups of G. Then the group

$$G' = \langle G, p_i, i \in I \mid p_i^{-1} h p_i = \phi_i(h), h \in H_i, i \in I \rangle$$

is called an *HNN extension* of G with stable letters p_i and associated subgroups H_i, K_i for $i \in I$.

For the remaining definitions and results in this section, we will assume that we are working with G', an HNN extension of G with stable letters p_i and associated subgroups H_i, K_i for $i \in I$, as defined above in Definition 6.2.

Definition 6.3. A *pinch* is a word of the form $p_i^{-1} v p_i$ or $p_i v p_i^{-1}$ where p_i is a stable letter and v is a word in G lying in H_i or K_i, respectively.

Lemma 6.4. *(Britton's Lemma) Let w be a word in $G \cup \{p_i\}$ containing some stable letter p_i or its inverse p_i^{-1}. If $w =_{G'} 1$, then w contains a pinch.*

Proposition 6.5. *The results below follow directly from Britton's Lemma and are taken from Aanderaa and Cohen* [12].

(i) Recall that G' is an HNN extension of G as given in Definition 6.2. Then G embeds in G'. That is to say, the canonical homomorphism $\phi : G \to G'$ is injective.

(ii) A good subgroup *of G is a subgroup $A \leq G$ such that $\phi_i(A \cap H_i) = A \cap K_i$ for all $i \in I$. Let A' be the subgroup of G' generated by $A, p_i, i \in I$, i.e., A plus the stable letters. Then A' is an HNN extension of A with the same stable letters, and $A' \cap G = A$.*

(iii) In the group $\langle G, p \mid p^{-1}hp = \phi(h) = h, h \in H \rangle$, where the isomorphism ϕ is the identity, we have $p^{-1}gp = g$ for $g \in G$ only if $g \in H$.

7. Unsolvability of the Word Problem

We now restate the word problem having built up the formal background of Turing machines and decidability.

Definition 7.1. Let $G = \langle S \mid R \rangle$ be a finitely presented group and W be the set of all words in S. We say G has a *solvable word problem* if the the set $\{w \in W \mid w =_G 1\}$ is decidable.

At last, we have reached the seminal proof of this paper, that the word problem for finitely presented groups is, in general, unsolvable. The result is usually termed the Novikov-Boone Theorem as it was proven independently by Pyotr S. Novikov and William W. Boone in 1955 and 1958, respectively. John L. Britton would later provide a simpler proof in 1963. All of these proofs in some way rely on the Halting Problem for Turing machines, and I encourage the reader to turn to Chapter 12 of Rotman [1] for an excellent treatment of them.

The following proof is due to Stål Aanderaa and Daniel E. Cohen (1980) [12], and it utilizes modular machines instead of Turing machines. The presentation we give below is in large part inspired by a recap of the proof written by Simpson [10].

Theorem 7.2. *There exists a finitely presented group with an unsolvable word problem.*

Let M be the modular machine for which the halting set $H_M = \{(\alpha, \beta) \mid (\alpha, \beta) \xrightarrow{M} (0,0)\}$ is undecidable. To provide a broad overview of the proof, we will show that if the word problem for a certain carefully constructed finitely presented group $(G'_M)'$ is solvable, then the halting set H_M is decidable. Because H_M is undecidable, it follows that $(G'_M)'$ must have an unsolvable word problem.

In order to get to this final step, however, we must first construct a number of groups. The following diagram provides a sense of how these groups are related and will hopefully serve as a useful reference:

$$\begin{array}{ccccc} G & \hookrightarrow & G'_M & \hookrightarrow & (G'_M)' \\ \cup & & \cup & & \\ T & \hookrightarrow & T' & & \\ \cup & & \cup & & \\ T_M & \hookrightarrow & T'_M & & \end{array}$$

The bulk of the proof will consist of proving the equality

$$T_M = T'_M \cap G = \langle t \rangle' \cap G,$$

which we accomplish in two steps: first, by showing that $T_M = T'_M \cap G$ and second, that $T'_M = \langle t \rangle'$.

We will begin to define most of the necessary groups here; the rest will be defined once they are required in the preliminary proofs. To start, we define

$$G = \langle t, x, y \mid xy = yx \rangle \cong \mathbb{Z} * \mathbb{Z}^2.$$

We use the notation $t(\alpha, \beta)$ to represent $x^{-\alpha}y^{-\beta}tx^{\alpha}y^{\beta}$ where $\alpha, \beta \in \mathbb{Z}$. Note that $t = t(0, 0)$. Next, let

$$G > T = \langle t(\alpha, \beta) \mid \alpha, \beta \in \mathbb{Z} \rangle.$$

For any $a, b, M, N \in \mathbb{Z}$ such that $0 \leq a < M$ and $0 \leq b < N$, we define the subgroup

$$G \geq G_{ab}^{MN} = \langle t(a, b), x^M, y^N \rangle.$$

The observant reader might notice that our requirements of a, b, M, and N are somewhat reminiscent of the requirements placed on the construction of a modular machine—wherein we have an integer m and require that $0 \leq a, b < m$ for all transition quadruples (a, b, c, D). This is so that the elements G_{ab}^{MN} can be used to describe configurations of M. Soon, we will locate functions which can describe transitions between the configurations.

Label the transition quadruples of M as

$$\{(a_i, b_i, c_i, R) \mid i \in I\} \cup \{(a_j, b_j, c_j, L) \mid j \in J\}.$$

For $i \in I$, the subgroups $G_{a_i b_i}^{MM}, G_{c_i 0}^{M^2 1}$ have a family of canonical isomorphisms

$$\phi_i : G_{a_i b_i}^{MM} \to G_{c_i 0}^{M^2 1} \text{ defined by } t(a_i, b_i) \mapsto t(c_i, 0), x^M \mapsto x^{M^2}, y^M \mapsto y,$$

and for $j \in J$, the isomorphisms $\psi_j : G_{a_j b_j}^{MM} \to G_{0 c_j}^{1 M^2}$ are defined in an analogous manner. Note that ϕ_i essentially performs an R transition, and ψ_j performs an L transition. Now we can construct the HNN extension G'_M of G with stable letters $r_i, i \in I$ and $l_j, j \in J$. We require that $g \mapsto r_i^{-1} g r_i$ extends ϕ_i and $g \mapsto l_j^{-1} g k_j$ extends ψ_j. To write it explicitly (and slightly byzantine):

$$G'_M = \langle G, r_i, l_j \mid r_i^{-1} g_i r_i = \phi_i(g_i), l_j^{-1} g_j l_j = \psi_j(g_j), g_i \in G_{a_i b_i}^{MM}, g_j \in G_{a_j b_j}^{MM} \rangle.$$

Finally, we define

$$T_M = \langle t(\alpha, \beta) \mid (\alpha, \beta) \in H_M \rangle,$$

then proceed to prove two results about it.

Lemma 7.3. $T_M = T'_M \cap G$

Proof. Note that T'_M is the subgroup of G'_M generated by $t(\alpha, \beta), r_i, l_j$ for $i \in I, j \in J$, and $\alpha, \beta \in \mathbb{Z}$ (so it's an HNN extension with stable letters r_i, l_j). By Proposition 6.5.ii, to prove $T_M = T'_M \cap G$, it will suffice to show that T_M is a good subgroup of G with respect to G'_M.

We already have

$$\phi_i(T_M \cap G_{a_i b_i}^{MM}) = \phi_i(T_M) \cap \phi_i(G_{a_i b_i}^{MM}) = \phi_i(T_M) \cap G_{c_i 0}^{M^2 1}$$

for all $i \in I$. Now we prove that $\phi_i(T_M) = T_M$. Recall that ϕ_i acts on a configuration $t(\alpha, \beta)$ as an R transition, which can easily be checked by the definition of ϕ_i. We have $\phi_i(t(\alpha, \beta)) = t(\alpha_1, \beta_1)$ implies $(\alpha, \beta) \to (\alpha_1, \beta_1)$, so

$$t(\alpha, \beta) \in T_M \iff (\alpha, \beta) \in H_M \iff (\alpha_1, \beta_1) \in H_M \iff t(\alpha_1, \beta_1) \in T_M.$$

Therefore, it follows from $\phi_i(t(\alpha, \beta)) = t(\alpha_1, \beta_1)$ that $t(\alpha, \beta) \in T_M$ if and only if $t(\alpha_1, \beta_1) \in T_M$; this proves that $\phi_i(T_M) = T_M$, which then gives us $\phi_i(T_M \cap G_{a_i b_i}^{MM}) = T_M \cap G_{c_i 0}^{M^2 1}$. An identical argument proves $\psi_j(T_M \cap G_{a_j b_j}^{MM}) = T_M \cap G_{0 c_j}^{1 M^2}$ for all $j \in J$, so T_M is a good subgroup of G. □

Lemma 7.4. $T'_M = \langle t \rangle'$

Proof. Similar to T'_M, we define $\langle t\rangle'$ as the subgroup of G'_M generated by t, r_i, l_j for $i \in I, j \in J$, (also an HNN extension with stable letters r_i, l_j).

First, we show that $\langle t\rangle' \subseteq T'_M$. We know that $(0,0) \in H_M$ (as the case where M halts immediately on the input), so $t(0,0) = t \in T_M$. Because T_M has t as a generator, then T'_M does as well, so it follows that $\langle t\rangle' \subseteq T'_M$.

Next, recall that any $(\alpha, \beta) \in H_M$ is the start of a finite sequence of configurations

$$(\alpha,\beta) = (\alpha_1,\beta_1) \to (\alpha_2,\beta_2) \to ... \to (\alpha_n,\beta_n) = (0,0),$$

where n is the length of computation on (α, β). We use induction on the length of computation n to show that $t(\alpha, \beta) \in \langle t\rangle'$ for all $(\alpha, \beta) \in H_M$.

For the base case of $n = 1$, we have the trivial computation $(\alpha, \beta) = (\alpha_1, \beta_1) = (0, 0)$, where clearly $t(0, 0) = t \in \langle t\rangle'$.

Now we suppose that $t(\alpha, \beta) \in \langle t\rangle'$ for all $(\alpha, \beta) \in H_M$ with length of computation n. Consider the following computation of length $n + 1$:

$$(\alpha,\beta) = (\alpha_1,\beta_1) \to (\alpha_2,\beta_2) \to ... \to (\alpha_{n+1},\beta_{n+1}) = (0,0).$$

The computation consisting of the subsequence

$$(\alpha_2,\beta_2) \to ... \to (\alpha_{n+1},\beta_{n+1}) = (0,0)$$

has length n, which implies $t(\alpha_2, \beta_2) \in \langle t\rangle'$ by our supposition. All that is left for us to show is that $(\alpha, \beta) \to (\alpha_2, \beta_2)$ means $t(\alpha, \beta) \in \langle t\rangle'$ as well. Let us assume without loss of generality that $(\alpha, \beta) \to (\alpha_2, \beta_2)$ by the transition quadruple $(a_i, b_i, c_i, R) \in M$, so by the definition of the modular machine, we have $(\alpha, \beta) = (uM+a_i, vM+b_i)$ and $(\alpha_2, \beta_2) = (uM^2 + c_i, v)$. We can now expand $t(\alpha, \beta)$ to get

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

Because $\langle t\rangle'$ is an HNN extension with stable letters $r_i, i \in I$, we know that $r_i\langle t\rangle' r_i^{-1} = \langle t\rangle'$. So if we can show that $t(\alpha, \beta)$ and $t(\alpha_2, \beta_2) \in \langle t\rangle'$ are conjugate with respect to r_i, then it will follow that $t(\alpha, \beta) \in \langle t\rangle'$ as well. Recall that $g \mapsto r_i^{-1} g r_i$ extends the canonical isomorphism $G^{MM}_{a_i b_i} \to G^{M^2 1}_{c_i 0}$; thus,

$$\begin{aligned} r_i^{-1}t(\alpha,\beta)r_i &= \phi_i(x^{-uM}y^{-vM}t(a_i,b_i)x^{uM}y^{vM}) \\ &= x^{-uM^2}y^{-v}t(c_i,0)x^{uM^2}y^{v} \\ &= x^{-uM^2}y^{-v}x^{-c_i}tx^{c_i}x^{uM^2}y^{v} \\ &= x^{-uM^2-c_i}y^{-v}tx^{uM^2+c_i}y^{v} \\ &= t(uM^2+c_i,v) \\ &= t(\alpha_2,\beta_2). \end{aligned}$$

We are left with $r_i^{-1}t(\alpha, \beta)r_i = t(\alpha_2, \beta_2)$, which implies $t(\alpha, \beta) = r_i t(\alpha_2, \beta_2) r_i^{-1}$. Therefore, $t(\alpha, \beta)$ is in fact conjugate to $t(\alpha_2, \beta_2)$, so $t(\alpha, \beta) \in \langle t\rangle'$.

By the inductive hypothesis, $t(\alpha, \beta) \in \langle t\rangle'$ for all $(\alpha, \beta) \in H_M$, and this is sufficient to show that $T'_M \subseteq \langle t\rangle'$. Combining this with the earlier result $\langle t\rangle' \subseteq T'_M$, we get the desired end result: $T'_M = \langle t\rangle'$. □

Everything will now come together as we provide a finitely presented group with an unsolvable word problem, proving the Novikov-Boone Theorem.

Proof. Consider the HNN extension

$$(G'_M)' = \langle G'_M, k \mid k^{-1}hk = h, h \in \langle t \rangle' \rangle$$

Because G'_M and $\langle t \rangle'$ are finitely generated, then $(G'_M)'$ has finitely many generators and relators, so it is finitely presented. Furthermore, for $g \in G'_M$, Proposition 6.5.iii tells us $g \in \langle t \rangle'$ if and only if $k^{-1}gk = g$. Because $T \subset G'_M$, then specifically all $t(\alpha, \beta) \in \langle t \rangle'$ if and only $k^{-1}t(\alpha, \beta)k = t(\alpha, \beta)$.

From the previous lemmas, we have $T_M = T'_M \cap G$ and $T'_M = \langle t \rangle'$; putting these equalities together yields $\langle t \rangle' \cap G = T_M$. It follows that if $t(\alpha, \beta) \in \langle t \rangle'$, then $t(\alpha, \beta) \in T_M$, which means $(\alpha, \beta) \in H_M$.

To recap the entire chain of logic, we have

$$k^{-1}t(\alpha, \beta)k = t(\alpha, \beta) \iff t(\alpha, \beta) \in \langle t \rangle' \iff t(\alpha, \beta) \in T_M \iff (\alpha, \beta) \in H_M.$$

If we have a way to determine whether $k^{-1}t(\alpha, \beta)k = t(\alpha, \beta)$ is true, then it will tell us whether M halts on (α, β). To reword that last statement more suggestively, if there is an algorithm to determine if the word $t(\alpha, \beta)^{-1}k^{-1}t(\alpha, \beta)k$ is trivial, then the same algorithm can also solve the Halting Problem for M. We cannot systematically determine whether $(\alpha, \beta) \in H_M$, so there must be some words $t(\alpha, \beta)^{-1}k^{-1}t(\alpha, \beta)k$ whose triviality we also cannot determine.

As the reader can probably tell, we have essentially finished the proof. Yet let us briefly rewrite the conclusion more formally:

Suppose, for later contradiction, that the word problem for $(G'_M)'$ is solvable. Then, letting W be all words in $G'_M \cup \{k\}$, the set $\{w \in W \mid w =_{(G'_M)'} 1\}$ is decidable. Define $w(\alpha, \beta)$ to be any word of the form $t(\alpha, \beta)^{-1}k^{-1}t(\alpha, \beta)k$. The subset $\{w(\alpha, \beta) \in W \mid w(\alpha, \beta) =_{(G'_M)'} 1\}$ must be decidable as well. Because $w(\alpha, \beta) =_{(G'_M)'} 1$ if and only if $(\alpha, \beta) \in H_M$, then H_M is decidable; this is a clear contradiction. Therefore, the word problem for $(G'_M)'$ is unsolvable. □

Acknowledgments

First and foremost, thank you to my mentor Peng Hui How for sharing countless resources, clarifying key concepts, providing critical feedback on my paper, and patiently bearing with me as I struggled to settle on a topic. Next, I want to thank Daniil Rudenko for directing the Apprentice Program, which provided an excellent background for many of the topics in this paper. Thank you as well to Jacob Fiedler and Connor Lockhart for running the apprentice program problem sessions. Lastly, thank you to Peter May; none of this would've been possible without his organization of the REU program.

References

[1] Joseph J. Rotman. An Introduction to the Theory of Groups. 4th ed. Springer-Verlag New York, Inc. 1995.

[2] Derek J.S. Robinson. A Course in the Theory of Groups. 2nd ed. Springer-Verlag New York, Inc. 1996.

[3] Mark Brittenham. Lecture Notes, Group Theory MATH 4106/5106 Winter 2007. `https://www.math.unl.edu/~mbrittenham2/classwk/990s08/public/bumagin.-.free.groups.notes.pdf`

[4] Matt Clay, Dan Margalit. Office Hours with a Geometric Group Theorist. Princeton University Press. 2017.

[5] Max Dehn. Über unendliche diskontinuierliche Gruppen. Mathematische Annalen 71. 116-144.
[6] Alan M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society. Volume s2-42, Issue 1, 1937. 230–265.
[7] Michael Sipser. Introduction to the Theory of Computation. 3rd ed. Cengage Learning. 2013.
[8] Charles F. Miller III. Turing Machines to Word Problems. ftp://ftp.math.ucla.edu/pub/asl/final/joyce/turningbook/turingbook/miller/miller-tmtwp-2012-09-28.pdf
[9] Joseph P. MacManus. Modular Machines and their equivalence to Turing Machines. https://jpmacmanus.me/2019/06/18/modularmachines.html
[10] Stephen G. Simpson. A Slick Proof of the Unsolvability of the Word Problem for Finitely Presented Groups. http://www.personal.psu.edu/t20/logic/seminar/050517.pdf
[11] Stephen G. Simpson. Topics in Logic and Foundations: Spring 2005. http://www.personal.psu.edu/t20/notes/topics-s05.pdf
[12] Stål Aanderaa and Daniel E. Cohen. Modular Machines, The Word Problem for Finitely Presented Groups and Collins' Theorem. Studies in Logic and the Foundations of Mathematics. North-Holland Publishing Company. 1980.