# A Finitely Presented Semigroup with Unsolvable Word Problem[1]

Gabriel Conant
March 22, 2010

## I. Turing Machines

Informally, a Turing machine can be thought of as a box with a tape running through it. The tape is an infinite row of consecutive squares and the box is capable of printing a finite number of symbols $s_1, \ldots, s_n$ on the tape (one symbol per square). Additionally, at any stage of computation, the box can be in one of a finite number of states $q_0, \ldots, q_m$ and reading a single square on the tape. The computation of the machine is determined by an initial configuration of the tape, and consists of either stopping or entering a new state after obeying one of the following instructions:

1. Erase the symbol in the box and print a new one.
2. Move one square to the right and read this square.
3. Move one square to the left and read this square.

The computation may either stop or run forever.

To formalize this definition, fix sets of letters $S = \{s_0, s_1, s_2, \ldots\}$ (possible symbols) and $Q = \{q_0, q_1, q_2, \ldots\}$ (possible states). Let $S_N = \{s_0, \ldots, s_N\}$.

**Definition**. An **instruction** is a quadruple of one of the following types

1. $q_i s_j s_k q_l$ (if in state $q_i$ and reading symbol $s_j$, erase $s_j$, print $s_k$ and enter state $q_l$)
2. $q_i s_j R q_l$ (if in $q_i$ and reading $s_j$, move one square to the right and enter $q_l$)
3. $q_i s_j L q_l$ (if in $q_i$ and reading $s_j$, move one square to the left and enter $q_l$)

These quadruples summarize the informal instructions of the machine given above.

**Definition**. A **Turing machine** $T$ is a finite nonempty set of instructions, no two of which have the same first two letters.

[1]summary of Rotman, Joseph J. *An Introduction to the Theory of Groups, third edition.* Allyn and Bacon, Inc., 1984. pp. 351-361.

**Note:** By demanding that no two instructions of the machine have the same first two letters, we ensure that the machine is deterministic.

**Definition**. The **alphabet** of a Turing machine $T$ is the set $S_N \subseteq S$ of all letters that occur in the instructions of $T$. Note: We can always ensure $T$ uses all of $S_N$.

**Notation**: In general, if $X$ is a set of letters then we denote by $X^*$ the set of all finite words consisting of nonnegative powers of letters in $X$.

At this time, we fix the letter $s_0$ as only being used to represent a blank square of the tape.

**Definition**. An **instantaneous description** $\alpha$ of a Turing machine is a finite word consisting of letters in $S$ and a single instance of a letter in $Q$ (called the **state of** $\alpha$), which is not the last letter of $\alpha$.

An instantaneous description $\alpha$ should be thought of as a complete description of the machine at a given moment, c.f., the letters on the tape are given (in order) by the $s_i$ in $\alpha$ (with blanks elsewhere), and the machine is in state $q_j$ reading the letter in $\alpha$ that occurs after $q_j$.

**Definition**. Given a Turing machine $T$ and instantaneous descriptions $\alpha, \beta$, we say $\alpha \to \beta$ if there are words $v, w \in S^*$ such that one of the following conditions holds:

1. $\left.\begin{array}{l}\alpha = vq_is_jw \\ \beta = vq_ls_kw\end{array}\right\}$ where $q_is_js_kq_l \in T$
2. $\left.\begin{array}{l}\alpha = vq_is_js_kw \\ \beta = vs_jq_ls_kw\end{array}\right\}$ where $q_is_jRq_l \in T$
3. $\left.\begin{array}{l}\alpha = vq_is_j \\ \beta = vs_jq_ls_0\end{array}\right\}$ where $q_is_jRq_l \in T$
4. $\left.\begin{array}{l}\alpha = vs_kq_is_jw \\ \beta = vq_ls_ks_jw\end{array}\right\}$ where $q_is_jLq_l \in T$
5. $\left.\begin{array}{l}\alpha = q_is_jw \\ \beta = q_ls_0s_jw\end{array}\right\}$ where $q_is_jLq_l \in T$

Basically, $\alpha \to \beta$ should be thought to represent how an instantaneous description of $T$ can change to another after following a single instruction of the machine. The difference between conditions 2 and 3 deals with the situation where the machine is moving to the right and there is another letter to read, versus the situation where the machine is scanning the rightmost symbol of the tape and told to move to the right. In the latter situation, the string is lengthened by adding $s_0$ (which represents a blank

square) to the right of the string. The difference between conditions 4 and 5 is similar.

**Definition**. A **computation** of a Turing machine is a finite sequence of instantaneous descriptions $\alpha_1, \alpha_2, \ldots, \alpha_t$, where

$$\alpha_i \to \alpha_{i+1} \quad 1 \leq i \leq t-1$$

and $\alpha_t$ is **terminal**, i.e., there is no $\alpha$ such that $\alpha_t \to \alpha$. If $\alpha \to \beta$ is terminal, we also call the instruction of $T$ determining $\alpha \to \beta$ **terminal**.

**Examples of terminal instructions:**

1. $q_i s_j s_k q_l$ where no instruction of $T$ begins with $q_l s_k$
2. $q_i s_j * q_l$ where no instruction of $T$ begins with $q_l$

**Remark:** Suppose $\alpha$ is an instantaneous description of $T$, $q_i$ is the state of $\alpha$, and $s_j$ is the letter of $\alpha$ following $q_i$. Then $\alpha$ is terminal if no instruction of $T$ begins with $q_i s_j$. Otherwise there is a unique instantaneous description $\beta$ such that $\alpha \to \beta$.

**Definition**. If $S_N$ is the alphabet of $T$ and $w \in S_N^*$ then $T(w)$ **exists** if there is a computation of $T$ beginning with $q_1 w$. (We refer to $q_1$ as the **initial state of** $T$.)
**Note:** $q_1 w$ completely determines a sequence of instantaneous descriptions since $T$ is deterministic. Therefore $T(w)$ exists if and only if this sequence is a computation of $T$, i.e., if the sequence halts.
We say $T$ **enumerates** $e(T)$ where

$$e(T) = \{w \in S_N^* : T(w) \text{ exists}\}$$

**Definition**. For any $N$, a subset $E \subseteq S_N^*$ is **recursively enumerable (r.e.)** if there is some Turing machine $T$ whose alphabet contains $S_N$ such that $E = e(T)$.

In particular we can consider subsets $E \subseteq \mathbb{N}$ and say that $E$ is recursively enumerable if there is a Turing machine on the alphabet $S_1$ such that

$$E = \{n \in \mathbb{N} : T(s_1^{n+1}) \text{ exists}\}$$

**Example**: The set of even natural numbers is recursively enumerable.
**Proof**

Let $T = \{q_1 s_0 s_0 q_1,\ q_1 s_1 R q_2,\ q_2 s_0 s_0 q_0,\ q_2 s_1 R q_1\}$. If $n = 2k+1$ for $k \in \mathbb{N}$ then we have

$$q_1 s_1^{2k+1} \to s_1 q_2 s_1^{2k} \to s_1^2 q_1 s_1^{2k-1} \to s_1^3 q_2 s_1^{2k-2} \to \ldots \to s_1^{2k} q_1 s_1 \to s_1^{2k+1} q_2 s_0 \to s_1^{2k+1} q_0 s_0$$

and $s_1^{2k+1} q_0 s_0$ is terminal since no instruction of $T$ begins with $q_0$. Thus $T(s_1^{2k+1})$ exists.

Conversely, if $n = 2k$ then we have

$$q_1 s_1^{2k} \to s_1 q_2 s_1^{2k-1} \to s_1^2 q_1 s_1^{2k-2} \to \ldots \to s_1^{2k-1} q_2 s_1 \to s_1^{2k} q_1 s_0 \to s_1^{2k} q_1 s_0 \to s_1^{2k} q_1 s_0 \to \ldots$$

so $T(s_1^{2k})$ does not exist.

Altogether, $\{n \in \mathbb{N} : T(s_1^{n+1}) \text{ exists}\}$ is exactly the set of even natural numbers. $\square$

**Notation:** For $n \in \mathbb{N}$, we define "$T(n)$ exists" to mean "$T(s_1^{n+1})$ exists."

**Theorem 1.1** There are subsets of $\mathbb{N}$ that are not recursively enumerable.
**Proof**

A Turing machine $T$ can be thought of as a finite subset of $Q \times S \times (S \cup \{R, L\}) \times Q$, which is a countable set. Thus there are countably many Turing machines. Since a recursively enumerable subset of $\mathbb{N}$ is determined by a Turing machine in a well-defined way, there are only countably many recursively enumerable subsets of $\mathbb{N}$. $\square$

**Definition**. A subset $E \subseteq S_N$ is **recursive** if both $E$ and $S_N \backslash E$ are recursively enumerable.

The fundamental distinction between recursive and recursively enumerable is that for a recursively enumerable set $E \subseteq S_N$, even though $w \in E$ or $w \notin E$ for all $w \in S_N^*$, there is no way to know beforehand whether or not $T$ will stop on input $w$. In other words, there is no way to universally distinguish between $T$ running for a long time before halting and $T$ running forever. However, for a recursive set, there is a Turing machine $T_1$ that will stop if $w \in E$ and a Turing machine $T_2$ that will stop if $w \notin E$. Thus one of $T_1$ or $T_2$ is guaranteed to stop and we only need to wait for one of these two possibilities to happen.

Therefore, for a recursive set $E \subseteq S_N$, there is an algorithm that will decide, on input $w \in S_N^*$, whether or not $w \in E$. **Church's Thesis** proposes that the recursive sets are exactly those with such an algorithm.

**Proposition 1.1** Fix a finite alphabet $\Sigma$. The collection of recursively enumerable subsets of $\Sigma^*$ is closed under finite unions and intersections. The collection of recursive subsets of $\Sigma^*$ is closed under complementation and finite unions and intersections.

**Theorem 1.2** There is a recursively enumerable subset $E \subseteq \mathbb{N}$ that is not recursive.
**Proof**

From the theorem above, we know that there are countably many Turing machines. We will now effectively enumerate them.[2] Assign the number 0 to R, 1 to L, $2i+2$ to $q_i$, and $2i+3$ to $s_i$. Now, a Turing machine $T$ is given by $M$ instructions for some $M$ so juxtapose them to make a word $\sigma$ of length $4M$. Define $p_i$ to be the $i^{\text{th}}$ prime number and let

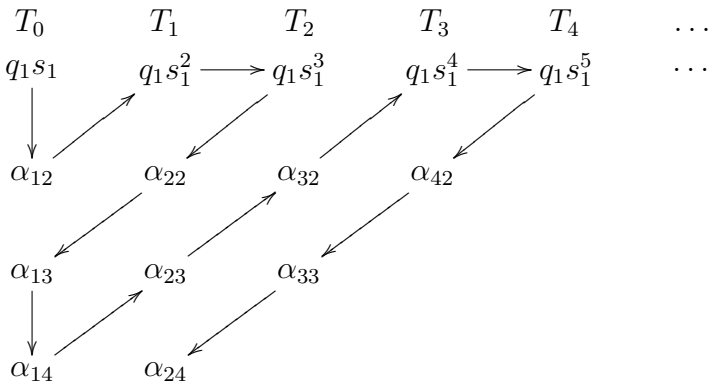
$$g(T) = \prod_{i=1}^{4M} p_i^{e_i}$$

[2]A list is **effectively enumerated** if there is some algorithm which will enumerate the list. Church's Thesis implies that "effectively enumerated" and "recursively enumerated" are equivalent.

where $e_i$ is the number assigned to the $i^{\text{th}}$ letter of $\sigma$. Uniqueness of prime factorization implies that each Turing machine will be given a distinct number $g(T)$. Now list these numbers in increasing order and this list will correspond to an effective enumeration $T_0, T_1, T_2, \ldots$ of all Turing machines. For a fixed $i$, let $q_1 s_1^{i+1} \to \alpha_{i+1,2} \to \alpha_{i+1,3} \to \ldots$ be the computation of $T_i$ on input $q_1 s_1^{i+1}$. Now define $E \subseteq \mathbb{N}$ such that

$$E = \{n \in \mathbb{N} : T_n(n) \text{ exists}\}$$

Then $E$ is recursively enumerable via a Turing machine which runs $T_i$ on input $q_1 s_1^{i+1}$ simultaneously for all $i$ according to the order of the arrows in the following diagram:




**Remark**: The computation of this machine is exactly that given above regardless of the input. In other words, let $C_i$ be the computation of $T_i$ on input $q_1 s_1^{i+1}$. Then this machine, regardless of the input, will run the first instantaneous description (i.d.) of $C_0$, the second i.d. of $C_0$, the first i.d. of $C_1$, the first i.d. of $C_2$, the second i.d. of $C_1$, the third i.d of $C_0$, the fourth i.d. of $C_0$, the third i.d. of $C_1$, and so on as indicated by the arrows in the diagram. Let $\mathcal{T}$ be this Turing machine. Then we say $\mathcal{T}(n)$ exists if $T_n$ halts at some point of this sole computation of $\mathcal{T}$. Thus $e(\mathcal{T}) = \{n \in \mathbb{N} : T_n(n) \text{ exists}\} = E$.

To show that $E$ is not recursive, it is enough to show that $\neg E = \mathbb{N} \backslash E$ is not recursively enumerable. So suppose otherwise. Then there is some Turing machine $T_k$ that enumerates $\neg E$. Thus we have

$$\begin{aligned} k \in E &\text{ iff } T_k(k) \text{ exists} \\ &\text{iff } k \in \neg E \end{aligned}$$

This contradiction implies $\neg E$ is not recursively enumerable and hence $E$ is not recursive. $\square$

**Lemma 1.1** Let $T$ be a Turing machine with $e(T) = E$. Then there is a Turing machine $T'$ with the same alphabet such that $e(T') = E$ and an instantaneous description $\alpha$ of $T'$ is terminal if and only if $\alpha$ involves the state $q_0$ (in this case we say

$q_0$ is the **stopping state** of $T'$).

**Proof**

Since $T$ is finite, let $M$ be such that $q_M$ does not occur in any instruction of $T$. First, let $T_0$ be the same set as $T$, but with all instances of $q_0$ replaced by $q_M$. Then $T_0$ is a Turing machine since if two instructions $I, J$ of $T_0$ begin the the same two letters (say $I = q_i s_j * q_k$ and $J = q_i s_j \star q_l$), we must have one of the following:

1. $q_i = q_M$, in which case $q_0 s_j * q'_k$ and $q_0 s_j \star q'_l$ are instructions of $T$ (where $q'_k = q_k$ iff $k \neq M$ and $q'_k = q_0$ iff $k = M$, similarly for $q'_l$). But then since $T$ is a Turing machine we must have $* = \star$ and $q'_k = q'_l$, which implies $q_k = q_l$ and thus $I = J$.

2. $q_i \neq q_M$ in which case $q_i s_j * q'_k$ and $q_i s_j \star q'_l$ are instructions of $T$ ($q'_k, q'_l$ as above), and so $* = \star$ and $q'_k = q'_l$, which implies $I = J$ similarly.

Note that $e(T) = e(T_0)$ since we have essentially only relabeled a state. Also, both Turing machines have the same alphabet $S_N$ since we have not changed the $S$ letters of any instruction. Finally, define

$$T' = T_0 \cup \{q_l s_j s_j q_0 : s_j \in S_N,\ q_l \text{ appears in an instruction of } T_0,\ q_l s_j s_k q_i \notin T_0\ \forall\ i, k\}$$

Note that $T'$ is finite since the number of added instructions is less than or equal to $N$ times the number of instructions of $T_0$. Also $T'$ is a Turing machine: no added instruction $q_l s_j s_j q_0$ can have the same first two letters of an instruction of $T_0$ since no instruction of $T_0$ begins with $q_l s_j$. Clearly, the alphabet of $T'$ is still $S_N$.

Next we show $T'$ has stopping state $q_0$. Suppose $\alpha$ is a terminal instantaneous description of $T'$. Let $q_l$ be the state of $\alpha$ and $s_j$ be the letter in $\alpha$ following $q_l$. Then $\alpha$ is reached from another instantaneous description via an instruction $I \in T'$. If $I \in T_0$ then $T_0$ involves $q_l$, but no instruction of $T_0$ can begin with $q_l s_j$ since this would also be an instruction of $T'$ contradicting that $\alpha$ is terminal. Then by definition of $T', q_l s_j s_j q_0 \in T'$ contradicting that $\alpha$ is terminal. Thus $I \in T' \backslash T_0$ so $I = q_i s_k s_k q_0$ for some $i, k$. If $q_l = q_k$ then, by definition of $T'$, some instruction of $T_0$ involves $q_l$, which we have shown to be impossible. Thus $q_l = q_0$. Conversely suppose that an instantaneous description $\alpha$ has state $q_0$. Since no instruction of $T'$ begins with $q_0$, $\alpha$ is terminal. Thus a description of $T'$ is terminal if and only if it involves $q_0$.

Finally we show $e(T') = E$. Suppose we have a computation $\alpha_1 \to \ldots \to \alpha_m$ of $T_0$. Then $\alpha_m$ is terminal with state $q_l$. Suppose $s_j$ is the letter following $q_l$ in $\alpha$. Then since $\alpha$ is terminal, no instruction of $T_0$ can begin with $q_l s_j$. Thus $q_l s_j s_j q_0$ is an instruction of $T'$ so in $T'$ we have the computation $\alpha_1 \to \ldots \to \alpha_m \to \beta$ where $\alpha_m \to \beta$ via the instruction $q_l s_j s_j q_0$. Note that $\beta$ is indeed a terminal description of $T'$ since its state is $q_0$. Therefore we have shown that $e(T_0) \subseteq e(T')$.

Conversely if $\alpha_1 \to \ldots \to \alpha_m$ is a computation of $T'$ then $\alpha_m$ is terminal with state $q_0$.

Case 1: $m = 1$. Since $q_1$ is the initial state of $T_0$, $q_1 \neq q_0$. Then $\alpha_1$ does not involve $q_1$ so does not contribute to $e(T')$.

Case 2: $m \neq 1$. Let $q_l$ be the state of $\alpha_{m-1}$. Then $\alpha_{m-1} \to \alpha_m$ must occur via the instruction $q_l s_j s_j q_0$ since no other instruction of $T'$ involves $q_0$. Then, by definition of $T'$, no instruction of $T_0$ begins with $q_l s_j$ so $\alpha_{m-1}$ is terminal in $T_0$, which implies $\alpha_1 \to \ldots \to \alpha_{m-1}$ is a computation of $T_0$.
Therefore $e(T') \subseteq e(T_0)$, and altogether we have

$$e(T') = e(T_0) = e(T) = E$$

□

We now fix $\mathcal{T}$ to be the Turing machine that both satisfies the conditions of the Lemma and also enumerates the recursively enumerable non-recursive set $E$ defined in Theorem 1.2.

## II. The Word Problem

**Definition**. Let $G$ be a finitely generated group with presentation $\langle x_1, \ldots, x_n \mid r_i, i > 0 \rangle$. Let $\Sigma = \{x_1, \ldots, x_n, x_1^{-1}, \ldots, x_n^{-1}\}$; we call $\Sigma$ a **symmetric generating set** for $G$. Define $\pi_G : \Sigma^* \longrightarrow G$ such that $\pi_G(w)$ is the element of $G$ representing $w$.

We say the **word problem** for $G$ is **solvable** if there exists an algorithm that will decide, for any $w \in \Sigma^*$, whether or not $\pi_G(w) = 1$. In the language of the previous section, $G$ has solvable word problem if $\{w \in \Sigma^* : \pi_G(w) = 1\}$ is a recursive subset of $\Sigma^*$.
**Fact**: If $G$ is finitely generated and recursively presented (i.e., defined by a recursively enumerable set of relations), then this definition does not depend on choice of presentation.

**Examples of groups with solvable word problem:**

1. Automatic groups
2. Finitely generated free groups
3. Finitely generated free abelian groups
4. Finite groups (since they are automatic)
5. Direct products of groups with solvable word problem
6. Finitely generated abelian groups (follows from 3,4,5)
7. Finitely generated subgroups of finitely generated groups with solvable word problem

We first observe the following corollary of Theorem 1.2.

**Corollary 2.1** There exists a finitely generated group $G$ with unsolvable word problem.

**Proof**

Let $E \subseteq \mathbb{N}$ be a recursively enumerable non-recursive set. Consider the group $G$ with the following recursive presentation:

$$G = \langle a, b, c, d \mid a^{-n}ba^n = c^{-n}dc^n,\ n \in E \rangle$$

Let $\Sigma = \{a, b, c, d, a^{-1}, b^{-1}, c^{-1}, d^{-1}\}$ and let $\Gamma = \{w \in \Sigma^* : \pi_G(w) = 1\}$. Then $a^{-n}ba^nc^{-n}d^{-1}c^n \in \Gamma$ if and only if $n \in E$. If $G$ has solvable word problem then $\Gamma$ is recursive, which implies $E$ is recursive,$\Rightarrow\Leftarrow$. Therefore $G$ does not have solvable word problem. $\square$

**Remark**: In light of the fact following the definition of solvable word problem, we observe that this theorem depends on $G$ being finitely generated. As an example let $F$ be the free group on countably many generators. Then

$$\mathcal{P}_1 = \langle x_1, x_2, x_3, \ldots \mid \emptyset \rangle \text{ and } \mathcal{P}_2 = \langle x_1, x_2, x_3 \ldots \mid x_i = 1,\ i \in E \rangle$$

are both (infinitely generated) recursive presentations for $F$. But the word problem is solvable for $\mathcal{P}_1$, while solvable word problem for $\mathcal{P}_2$ would imply that $E$ is recursive.

We now focus our efforts on constructing a finitely presented group with unsolvable word problem. Notice that, from our definition, it would seem that in order to prove that a group $G$ has solvable word problem, it is necessary to to prove that a certain set is recursive. However, the next theorem will show that for finitely presented groups the problem can be reduced to showing that a certain set is recursively enumerable.

**Theorem 2.1** Let $G$ be a finitely presented group with presentation

$$G = \langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$$

If $\Sigma$ is the associated symmetric generating set, then $\Gamma = \{w \in \Sigma^* : \pi_G(w) = 1\}$ is recursively enumerable.

**Proof**

We order $\Sigma$ by $x_1 < x_1^{-1} < x_2 < x_2^{-1} < \ldots < x_n < x_n^{-1}$. Then we order words in $\Sigma^*$ first by length and then lexicographically. Let $w_1, w_2, w_3, \ldots$ be an enumeration of $\Sigma^*$ which follows this order. For $w \in \Sigma^*$, say $w = a_1 \ldots a_t$ where $a_i \in \Sigma$, let $w^{-1}$ denote the word $a_t^{-1} \ldots a_1^{-1}$ where we declare $(x_i^{-1})^{-1} = x_i$. Then

$$\begin{aligned}\Gamma_0 &= \{w_i r_j w_i^{-1} : i > 0,\ 1 \leq j \leq m\} \\ &= \{w_1 r_1 w_1^{-1}, w_1 r_2 w_1^{-1}, \ldots, w_1 r_m w_1^{-1}, w_2 r_1 w_2^{-1}, w_2 r_2 w_2^{-1}, \ldots, w_2 r_m w_2^{-1}, \ldots\}\end{aligned}$$

is clearly recursively enumerable. So we have a recursive enumeration $\Gamma_0 = \{v_1, v_2, v_3, \ldots\}$; say $v_k = w_{\tilde{k}} r_{k-\tilde{k}+1} w_{\tilde{k}}^{-1}$ where $\tilde{k} = \lceil \frac{k}{m} \rceil$. Let $\Sigma_n$ be the finite set of words on $v_1, \ldots, v_n$ of length at most $n$. Then the following algorithm, on input $w \in \Sigma^*$, will halt if $w \in \Gamma$ and run forever otherwise:

**1.** Compute $v_1$. Check $w = z$ for all $z \in \Sigma_1$.

**2.** Compute $v_2$. Check $w = z$ for all $z \in \Sigma_2$.

$\vdots$

**N.** Compute $v_N$. Check $w = z$ for all $z \in \Sigma_N$.

$\vdots$

This gives a recursive enumeration for $\Gamma$ since

$$\begin{aligned} w \in \Gamma &\text{ iff } \pi_G(w) = 1 \\ &\text{iff } w \text{ is an element of the normal closure of } \{r_1, \ldots, r_m\} \\ &\text{iff } w = \prod_{j=1}^{K} v_{i_j} \text{ for some } K > 0 \text{ and } v_{i_j} \in \Gamma_0 \end{aligned}$$

Thus if $N = \max\{K, i_1, \ldots, i_K\}$, then $w \in \Sigma_N$ so the algorithm will halt during stage $N$. Conversely, if the algorithm halts for some input $w \in \Sigma^*$, then $w \in \Gamma$ since $\Sigma_n \subseteq \Gamma$ for all $n$. □

**Corollary 2.2** Suppose $G$ is a finitely presented group with symmetric generating set $\Sigma$. Then $G$ has solvable word problem if and only if

$$\begin{aligned} &\{w \in \Sigma^* : \pi_G(w) = 1\} \text{ is recursive} \\ \text{iff } &\{w \in \Sigma^* : \pi_G(w) = 1\} \text{ and } \{w \in \Sigma^* : \pi_G(w) \neq 1\} \text{ are recursively enumerable} \\ \text{iff } &\{w \in \Sigma^* : \pi_G(w) \neq 1\} \text{ is recursively enumerable.} \end{aligned}$$

We are now ready to continue the process of constructing a finitely presented semigroup with unsolvable word problem (this is the first step in constructing a finitely presented group with unsolvable word problem). For this, we must alter our definition slightly.

**Definition**. Suppose $G$ is a finitely generated semigroup with generating set $\Sigma = \{x_1, \ldots, x_n\}$. Then $G$ has **solvable word problem** if there is an algorithm that, for any words $v, w \in \Sigma^*$, will determine whether or not $\pi_G(v) = \pi_G(w)$. Equivalently, $G$ is has unsolvable word problem if there is a word $w_0 \in \Sigma^*$ such that $\{w \in \Sigma^* :$

$\pi_G(w) = \pi(w_0)\}$ is not recursive.

Suppose $G$ has presentation

$$\langle x_1 \ldots, x_n \mid r_1 = \hat{r}_1, \ldots, r_m = \hat{r}_m \rangle$$

Then for $v, w \in \Sigma^*$, we say $v \to w$ if $v = w$ or there are words $z_1, z_2 \in \Sigma^*$ and some $1 \leq i \leq m$ such that $v = z_1 r_i z_2$ and $w = z_1 \hat{r}_i z_2$ or $w = z_1 r_i z_2$ and $v = z_1 \hat{r}_i z_2$.

Now, for $v, w \in \Sigma^*$ we have $\pi_G(v) = \pi_G(w)$ if and only if there are $z_1, \ldots, z_t \in \Sigma^*$ such that $v \to z_1 \to \ldots \to z_t \to w$.

**Definition**. Let $T$ be a Turing machine. Let $S_N = \{s_0, \ldots, s_N\}$ be the alphabet of $T$ and let $Q_M = \{q_0, \ldots, q_M\}$ be the states occurring in the instructions of $T$. Let $q, h$ be letters not in $S_N$ or $Q_M$. Let $\Sigma_T = \{q, h\} \cup S_N \cup Q_M$. Define the following semigroup

$$G(T) = \langle \Sigma_T \mid R(T) \rangle$$

where $R(T)$ is a finite set of relations consisting of the following:

$$q_i s_j = q_l s_k \quad \text{if } q_i s_j s_k q_l \in T$$

and for all $b = 0, 1, \ldots, n$

$$\left.\begin{array}{rcl} q_i s_j s_b & = & s_j q_l s_b \\ q_i s_j h & = & s_j q_l s_0 h \end{array}\right\} \text{ if } q_i s_j R q_l \in T$$

$$\left.\begin{array}{rcl} s_b q_i s_j & = & q_l s_b s_j \\ h q_i s_j & = & h q_l s_0 s_j \end{array}\right\} \text{ if } q_i s_j L q_l \in T$$

$$q_0 s_b = q_0$$
$$s_b q_0 h = q_0 h$$
$$h q_0 h = q$$

The first five types of relations capture the basic moves of $T$, with the letter $h$ distinguishing between moves within the tape and moves at the ends of the tape. Thus we think of $h$ as marking the ends of the tape. In practice, we will work with Turing machines that contain a stopping state $q_0$ (as suggested by the final three relations). It will become apparent that we may think of $q$ as representing configurations of the tape that will halt.

We call a word in $\Sigma_T^*$ **$h$-special** if it is $hq_0h$ or of the form $h\alpha h$ for some instantaneous description $\alpha$ of $T$. Thus an $h$-special word describes the entire tape at a given moment of computation.

**Lemma 2.1** Let $T$ be a Turing machine with associated semigroup $G(T)$.

1. For $V, W \in \Sigma_T^*$ such that $V \neq q \neq W$ and $V \to W$, we have that $V$ is $h$-special if and only if $W$ is $h$-special.

2. If $V = h\alpha h$, $W \neq q$, and $V \to W$ via a relation from among the first five types, then $W = h\beta h$ where either $\alpha \to \beta$ or $\beta \to \alpha$ is a basic move of $T$.

**Proof**

1. The only relation that creates or destroys $h$ is $hq_0h = q$.

2. $W \neq hq_0h$ since $V \to W$ via a relation from among the first five types. So $W = h\beta h$ by part (1). If the relation is of the first five types then since $T$ is deterministic we either have $\beta$ obtained from $\alpha$ via the basic move corresponding to this relation, or vice versa. Thus $\alpha \to \beta$ or $\beta \to \alpha$. □

**Lemma 2.2** Let $T$ be a Turing machine with stopping state $q_0$. Let $S_N$ be the alphabet of $T$, let $E = e(T)$, and let $\pi = \pi_{G(T)}$. Then for $w \in S_N^*$ we have

$$w \in E \text{ if and only if } \pi(hq_1wh) = \pi(q)$$

**Proof**

Suppose $w \in E$. Then we have a computation $q_1w \to \alpha_1 \to \ldots \to \alpha_t$ for instantaneous descriptions $\alpha_i$ and $\alpha_t$ terminal (so $\alpha_t$ has state $q_0$). Thus we may use the relations in $G(T)$ corresponding to the basic moves of $T$ to see that $\pi(hq_1wh) = \pi(h\alpha_th)$. But $\alpha_t = vq_0w$ where $v, w \in S_N^*$. Thus repeated use of the relation $q_0s_b = q_0$ gives $\pi(h\alpha_th) = \pi(hvq_0h)$; and then repeated use of the relation $s_bq_0h = q_0h$ give $\pi(hvq_0h) = \pi(hq_0h)$. Altogether we have

$$\pi(hq_1wh) = \pi(h\alpha_th) = \pi(hvq_0h) = \pi(hq_0h) = \pi(q)$$

since $hq_0h = q$ is also a relation.

Conversely suppose $\pi(hq_1wh) = \pi(q)$. Then there are words $W_i \in \Sigma_T^*$ such that $hq_1wh = W_1 \to \ldots \to W_t \to q$ (we assume $W_i \neq q$ for all $i$). Then by part (1) of the previous lemma, we conclude that $W_i$ is $h$-special and we can write $W_i = h\alpha_ih$ where $\alpha_i$ is either $q_0$ or an instantaneous description of $T$. Now, $h\alpha_th \to q$ so, by inspection of the relations of $G(T)$, we must conclude that $q_0$ is the state of $\alpha_t$. Let $t_0 \leq t$ be minimal such that $\alpha_{t_0}$ is an instantaneous description of $T$ and the state of $\alpha_{t_0}$ is $q_0$. Note that $t_0 > 1$ since $q_1 \neq q_0$. By part (2) of the previous lemma, $\alpha_i \to \alpha_{i+1}$ or $\alpha_{i+1} \to \alpha_i$ for all $i < t_0$. Now, if we have $\alpha_{i-1} \leftarrow \alpha_i \to \alpha_{i+1}$ then since $T$ is deterministic, $\alpha_{i-1} = \alpha_{i+1}$ so $W_{i-1} = W_{i+1}$ and we shorten the sequence:

$$W_1 \to \ldots \to W_{i-2} \to W_{i+1} \to \ldots \to W_{t_0}$$

Furthermore, $\alpha_{t_0-1} \to \alpha_{t_0}$ since $\alpha_{t_0}$ is terminal and $\alpha_{t_0-1}$ does not involve $q_0$. Thus by induction we may assume $\alpha_i \to \alpha_{i+1}$ for all $i < t_0$. But $\alpha_{t_0}$ is terminal so

$q_1 w \to \ldots \to \alpha_{t_0}$ is a computation of $T$. Therefore $w \in E$. $\square$

**Theorem 2.2** Let $\mathcal{T}$ be the Turing machine specified at the end of the last section. Then $G(\mathcal{T})$ is a finitely presented semigroup with unsolvable word problem. There is no algorithm to determine, for an arbitrary $h$-special word $h\alpha h$, whether or not $\pi(h\alpha h) = \pi(q)$.

**Proof**

Let $T$ be a Turing machine with the same conditions and notation as the previous lemma. Define $\Gamma = \{W \in \Sigma_T^* : \pi(W) = \pi(q)\}$. We can identify $S_N^*$ with the following subset $\bar{S}_N$ of $\Sigma_T^*$:

$$\bar{S}_N = \{hq_1 wh : w \in S_N^*\}$$

Then $E \subseteq S_N^*$ corresponds to the following subset $\bar{E}$ of $\bar{S}_N$:

$$\bar{E} = \{hq_1 wh : w \in E\}$$

In other words, $\bar{S}_N = hq_1 S_N^* h$ and $\bar{E} = hq_1 Eh$. By the previous lemma, $w \in E$ if and only if $\pi(hq_1 wh) = \pi(q)$, i.e.,

$$\bar{E} = \Gamma \cap \bar{S}_N$$

Now suppose $T = \mathcal{T}$. If $G(\mathcal{T})$ has solvable word problem then $\Gamma$ is recursive. Clearly, $S_N^*$ recursive implies $\bar{S}_N$ is recursive, which, by Proposition 1.1, implies $\bar{E}$ is recursive. But $\bar{E}$ recursive clearly implies $E$ is recursive, which is a contradiction. Therefore $G(\mathcal{T})$ has unsolvable word problem.

Finally, let $\bar{F} = \{h\alpha h : \pi(h\alpha h) = \pi(q)\}$. If $\bar{F}$ is recursive then $\bar{E} = \bar{F} \cap \bar{S}_N$ is recursive,$\Rightarrow\Leftarrow$. Thus $\bar{F}$ is not recursive, which proves the second statement of the theorem. $\square$

We restate Theorem 2.2 in a more general way, which does not use the language of Turing machines.

**Corollary 2.3** There is a finitely presented semigroup

$$G = \langle q, q_1, \ldots, q_m, s_1, \ldots, s_n \mid u_i q_{i_1} v_i = w_i q_{i_2} z_i,\ i \in I \rangle$$

with unsolvable word problem (where $u_i, v_i, w_i, z_i \in \{s_1, \ldots, s_m\}^*$ and $q_{i_1}, q_{i_2} \in \{q, q_1, \ldots, q_m\}$). There is no algorithm to determine, given arbitrary $v, w \in \{s_1, \ldots, s_m\}^*$ and $q_j$, whether or not $\pi(vq_j w) = \pi(q)$.

**Proof**

The first statement only requires a relabeling of the semigroup in Theorem 2.2. For the second statement, if $\bar{F}$ is as in the theorem and $X = \{vq_j w : \pi(vq_j w) = \pi(q)\}$, then $\bar{F} = X \cap \{h\alpha h : \alpha$ an instantaneous description of $\mathcal{T}\}$. So $X$ recursive implies $\bar{F}$ recursive. $\square$