Juliana Viola
Yale University
May 5, 2022

Groups Generated by Automata

An Undergraduate-Level Exposition

Contents

1 Introduction

During my time at Yale, my favorite course was CPSC 460 (Automata Theory taught by Andrew Bridy), and I knew I wanted to have my senior project somehow relate to automata theory. Last semester, I took MATH 350 (Introduction to Abstract Algebra), which exposed me to the basics of group theory and ring theory. Given my passion for automata theory and the ubiquity of group theory, I want to look at their intersection this semester for my senior project.

In my preliminary readings on the intersection between group theory and automata theory, I found most of the material to be quite dense and not particularly digestible for undergraduate readers [1, 3, 4]. My goal for this project was to write an exposition that explores the basics of groups generated by automata. I aimed to write in an accessible way so that an undergraduate student with exposure to both automata theory and group theory can follow along without too much extra legwork.

2 Group Theory Background

Recall the definition of a group from group theory.

Definition 2.1. A **group** is a set G along with a binary operation $\star$ obeying the following properties:

1. Associativity: For all $x, y, z \in G$, $(x \star y) \star z = x \star (y \star z)$.

2. Existence of an identity element: There exists some element $e \in G$ called the identity such that for all $x \in G$, $e \star x = x \star e = x$.

3. Existence of inverses: For every element $x \in G$, there exists an element $x^{-1} \in G$ which is the inverse of x, such that $x \star x^{-1} = x^{-1} \star x = e$.

Oftentimes, it can be useful to think about how a group interacts with another set. A classic example involves the symmetric group S_n. Formally, the elements of the symmetric group are permutations on the set $\{1, 2, 3, \ldots, n\}$; the group elements are *not* $1, 2, 3, \ldots, n$. With that being said, it's natural to consider how an element of S_n acts on the integers 1 through n. For instance, given the group S_5 and the permutation $\sigma = (123)(45)$, we see that σ maps 1 to 2, 2 to 3, 3 to 1, 4 to 5, and 5 to 4. With this intuition in mind, we can formally define group actions, starting with *left* group actions.

Definition 2.2. Given a group G and a set A, a **left group action** is a map from $G \times A$ to A. Each pair (g, a) for all $g \in G$ and all $a \in A$ is mapped to the output $g \cdot a \in A$. The map satisfies the following conditions:

1. For all $g_1, g_2 \in G$ and all $a \in A$, $g_1 \cdot (g_2 \cdot a) = (g_1 \cdot g_2) \cdot a$

2. For all $a \in A$, $e \cdot a = a$ (where e is the identity in G)

We define right groups actions similarly:

Definition 2.3. Given a group G and a set A, a **right group action** is a map from $A \times G$ to A. Each pair (a, g) for all $g \in G$ and all $a \in A$ is mapped to the output $a \cdot g \in A$. The map the satisfies following conditions:

1. For all $g_1, g_2 \in G$ and all $a \in A$, $(a \cdot g_1) \cdot g_2 = a \cdot (g_1 \cdot g_2)$

2. For all $a \in A$, $a \cdot e = a$

Finally, we introduce the idea of *generating* a group.

Definition 2.4. Given a group G and some subset $S \subseteq G$, we say that S **generates** G if any element in G can be expressed as a product of the elements of S and their inverses. We denote generation using the notation $G = \langle S \rangle$.

3 Automata Theory Background

Definition 3.1. An **alphabet** is a finite set of characters.

If we call our alphabet Σ, then we can introduce the idea of words (both finite in length and infinite in length).

Definition 3.2. Given an alphabet Σ, Σ^* is the set of all **finite-length words** over Σ.

Definition 3.3. Given an alphabet Σ, Σ^ω is the set of all (one-sided) **infinite sequences** over Σ. That is,

$$\Sigma^\omega = \{x_1x_2x_3x_4\ldots : x_i \in \Sigma \text{ for all } i\}$$

Recall from automata theory that a deterministic finite automata (DFA) takes a word as an input and outputs a state: either accept or reject. Transducers are different in that they take words as input and output other words.

Definition 3.4. A **finite state transducer (FST)** is a 6-tuple $(Q, \Sigma, \delta, q_0, \Delta, \lambda)$ where:

- Q is a set of states
- Σ is the input alphabet
- $\delta : Q \times \Sigma \to Q$ is a transition function
- q_0 is the initial state
- Δ is the output alphabet
- $\lambda : Q \times \Sigma \to \Delta^*$ is an output function

The computation works as follows. The machine reads in a word $w = x_1x_2\ldots x_n \in \Sigma^*$. The machine proceeds deterministically through a sequence of states $q_0, q_1, q_2, \ldots, q_n$ and outputs a word $y_1y_2\ldots y_n \in \Delta^*$ where:

1. $\delta(q_i, x_{i+1}) = q_{i+1}$ for all i
2. $\lambda(q_i, x_{i+1}) = y_i$ for all i. (Descriptively, each y_i is a word in Δ^*.)

With this definition in mind, we turn to a particular type of transducer: the Mealy machine.

Definition 3.5. A **Mealy machine** is a particular type of FST where every output is exactly one letter.

For an example of a Mealy machine, see Figure 1.

Although we often think of Mealy machines as taking in words of finite length as input, there's no reason we can't extend a Mealy machine to act on (one-sided) sequences of infinite length, too. So, instead of reading in words from Σ^* and outputting words from Δ^*, a Mealy machine can actually read in words from $\Sigma^* \cup \Sigma^\omega$ and output words from $\Delta^* \cup \Delta^\omega$.

Consider two machines that have the same input and output languages, i.e., suppose we have two machines, $\mathcal{M} = (Q, \Sigma, \delta, q_0, \Sigma, \lambda)$ and $\mathcal{N} = (R, \Sigma, \alpha, r_0, \Sigma, \beta)$. We can define a new Mealy machine $\mathcal{K} = \mathcal{M}\mathcal{N}$ which is the composition of $\mathcal{M}$ and $\mathcal{N}$ as follows:

- Let $Q \times R$ be the state set of $\mathcal{K}$
- Let (q_0, r_0) be the initial state of $\mathcal{K}$

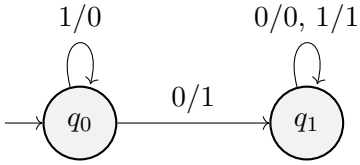


Figure 1: A basic Mealy machine. The notation i/o next to each transition signifies the input and output symbols. Explicitly, our initial state is q_0 (indicated by the incoming arrow), and we have two states total: q_0 and q_1. We have the same input and output alphabet: $\{0, 1\}$. If we let δ be our transition function and λ be our output function, then we can explicitly define each function. For δ: $\delta(q_0, 0) = q_1$; $\delta(q_0, 1) = q_0$; $\delta(q_1, 0) = q_1$; $\delta(q_1, 1) = q_1$. For λ: $\lambda(q_0, 0) = 1$; $\lambda(q_0, 1) = 0$; $\lambda(q_1, 0) = 0$; $\lambda(q_1, 1) = 1$. This machine can be viewed as performing addition by 1 in binary, if we think of binary numbers in reverse (i.e., if we read them from right to left). For example, consider the input 11001, which will be transformed into 00101. 11001 read right to left is 19 in decimal; 00101 read right to left is $20 = 19 + 1$ in decimal.

- Define the transition function $\psi : (Q \times R) \times \Sigma \to Q \times R$ as follows: for each $q \in Q$, each $r \in R$, and each $x \in \Sigma$, set $\psi((q, r), x) = (\delta(q, x), \alpha(r, \lambda(q, x)))$
- Define the output function $\omega : (Q \times R) \times \Sigma \to \Sigma$ as follows: for each $q \in Q$, each $r \in R$, and each $x \in \Sigma$, set $\omega((q, r), x) = \beta(r, \lambda(q, x))$

Proposition 3.1. *Consider any word $w \in \Sigma^* \cup \Sigma^\omega$. Suppose that given input w, $\mathcal{M}$ outputs the word $u \in \Sigma^* \cup \Sigma^\omega$, and then u is fed into $\mathcal{N}$, which outputs the word $v \in \Sigma^* \cup \Sigma^\omega$. If w is fed into $K = \mathcal{MN}$, then the output is v.*

Proof. Let v' be the word that $\mathcal{K}$ outputs upon receiving the input w. We need to show that $v' = v$. Suppose w has length n, which may be infinite. It's enough to show that for all $i \leq n$, $v'_i = v_i$ (that is, v and v' are identical at every index). We proceed by induction on i.

Base case: Let $i = 1$, and let $\mathcal{K}$ be in state (q_0, r_0). Suppose $\delta(q_0, w_1) = q_1$, $\lambda(q_0, w_1) = u_1$, $\alpha(r_0, u_1) = r_1$, and $\beta(r_0, u_1) = v_1$. By construction, $v'_1 = \omega((q_0, r_0), w_1) = \beta(r_0, \lambda(q_0, w_1)) = \beta(r_0, u_1) = v_1$. Thus v' and v begin with the same symbol. Also note that the machine is in state (q_1, r_1) because $\psi((q_0, r_0), w_1) = (\delta(q_0, w_1), \alpha(r_0, \lambda(q_0, w_1))) = (q_1, \alpha(r_0, u_1)) = (q_1, r_1)$.

Inductive hypothesis: $v'_i = v_i$ for all $i \leq k < n$. Furthermore, after processing $v_1 \dots v_k$, $\mathcal{K}$ is in state (q_k, r_k).

Inductive step: Assume the inductive hypothesis to be true. We need to show that $v'_{k+1} = v_{k+1}$, and after processing $v_1 \dots v_{k+1}$, $\mathcal{K}$ is in state (q_{k+1}, r_{k+1}). We have $\delta(q_k, w_{k+1}) = q_{k+1}$, $\lambda(q_k, w_{k+1}) = u_{k+1}$, $\alpha(r_k, u_{k+1}) = r_{k+1}$, and $\beta(r_k, u_{k+1}) = v_{k+1}$. By construction,

$$v'_{k+1} = \omega((q_k, r_k), w_{k+1}) = \beta(r_k, \lambda(q_k, w_{k+1})) = \beta(r_k, u_{k+1}) = v_{k+1}.$$

Thus $v'_{k+1} = v_{k+1}$, so $v'_1 \dots v'_{k+1} = v_1 \dots v_{k+1}$. Additionally, the machine is in state (q_{k+1}, r_{k+1})

because $\psi((q_k, r_k), w_{k+1}) = (\delta(q_k, w_{k+1}), \alpha(r_k, \lambda(q_k, w_{k+1}))) = (q_{k+1}, \alpha(r_k, u_{k+1})) = (q_{k+1}, r_{k+1})$.

Therefore, by induction, $v = v'$. □

We can also consider what it would mean for a Mealy machine (with the same input and output alphabets) to be invertible. First, though, we need to define what it means for a Mealy machine to be complete with respect to input and complete with respect to output [1].

Definition 3.6. A Mealy machine is **complete with respect to input** if for each of its states and for each letter x in its input alphabet, there is a unique outgoing edge with input x.

Definition 3.7. A Mealy machine is **complete with respect to output** if for each of its states and for each letter y in its output alphabet, there is a unique outgoing edge with output y.

Definition 3.8. A Mealy machine that has the same input and output alphabets is **invertible** if it is complete with respect to both input and output.

Thus, invertibility is a *local property* in the sense that a machine is invertible if and only if each one of its states is invertible. For an example of invertible and non-invertible Mealy machines, see Figures 2 and 3, respectively.

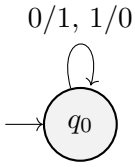


Figure 2: A simple invertible Mealy machine that takes the one's complement of the inputted word – that is, it flips each bit in the inputted word. For example, the word 10110 would be transformed into 01001.

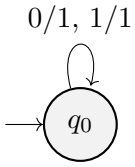


Figure 3: A simple invertible Mealy machine that's *not* invertible – the output function maps both $(q_0, 0)$ and $(q_0, 1)$ to 1. Any word of length ℓ would be transformed into a sequence of ℓ consecutive ones. Intuitively, it's clear that the machine is not invertible because given an outputted word, one cannot deduce the input.

To take the inverse of a Mealy machine $\mathcal{A} = (Q, \Sigma, \delta, q_0, \Sigma, \lambda)$, where δ maps from $Q \times \Sigma$ to Q and λ maps from $Q \times \Sigma$ to Σ, we use the following procedure:

1. Let R represent the states of $\mathcal{A}^{-1}$. Define $R = \{q_i^{-1} : q_i \in Q\}$.

2. Let q_0^{-1} be the initial state of $\mathcal{A}^{-1}$.

3. Let the input and output alphabet for $\mathcal{A}^{-1}$ be Σ.

4. Define the transition function $\epsilon : R \times \Sigma \to R$ as follows: for each q_i and each x where $\delta(q_i, x) = q_j$ and $\lambda(q_i, x) = y$, we set $\epsilon(q_i^{-1}, y) = q_j^{-1}$.

5. Define the output function $\mu : R \times \Sigma \to \Sigma$ as follows: for each q_i and each x where $\lambda(q_i, x) = y$, we set $\mu(q_i^{-1}, y) = x$.

Proposition 3.2. *Let $\mathcal{B} = \mathcal{A}\mathcal{A}^{-1}$ and let $\mathcal{C} = \mathcal{A}^{-1}\mathcal{A}$. Then $\mathcal{B}$ and $\mathcal{C}$ are both trivial machines. That is, given any input word $w \in \Sigma^* \cup \Sigma^\omega$, the output of both $\mathcal{B}$ and $\mathcal{C}$ is w.*

Proof. Using machine composition as described previously, we define the following functions:

- The transition function for $\mathcal{B}$ is $\psi_B : (Q \times R) \times \Sigma \to Q \times R$. For each $q \in Q$, each $r \in R$, and each $x \in \Sigma$, set $\psi_B((q, r), x) = (\delta(q, x), \epsilon(r, \lambda(q, x)))$

- The output function for $\mathcal{B}$ is $\omega_B : (Q \times R) \times \Sigma \to \Sigma$. For each $q \in Q$, each $r \in R$, and each $x \in \Sigma$, set $\omega_B((q, r), x) = \mu(r, \lambda(q, x))$

- The transition function for $\mathcal{C}$ is $\psi_C : (R \times Q) \times \Sigma \to R \times Q$. For each $r \in R$, each $q \in Q$, and each $x \in \Sigma$, set $\psi_C((r, q), x) = (\epsilon(r, x), \delta(q, \mu(r, x)))$

- The output function for $\mathcal{C}$ is $\omega_C : (R \times Q) \times \Sigma \to \Sigma$. For each $r \in R$, each $q \in Q$, and each $x \in \Sigma$, set $\omega_C((r, q), x) = \lambda(q, \mu(r, x))$

Let w' be the word that $\mathcal{B}$ outputs upon receiving the input w. Similarly, let w'' be the word that $\mathcal{C}$ outputs upon receiving the input w. We need to show that $w'' = w' = w$. Suppose w has length n, which may be infinite. We need to show that for all $i \leq n$, $w_i'' = w_i' = w_i$. We proceed by induction on i.

Base case: Let $i = 1$, let $\mathcal{B}$ be in state (q_0, q_0^{-1}), and let $\mathcal{C}$ be in state (q_0^{-1}, q_0). We start by proving the base case for $\mathcal{B}$. Suppose $y_1 = \lambda(q_0, w_1)$. By construction, $w_1 = \mu(q_0^{-1}, y_1)$. Then $w_1' = \omega_B((q_0, q_0^{-1}), w_1) = \mu(q_0^{-1}, \lambda(q_0, w_1)) = \mu(q_0^{-1}, y_1) = w_1$. Thus w and w' both start with the same symbol. Also, $\mathcal{B}$ is in state (q_1, q_1^{-1}) because $\psi_B((q_0, q_0^{-1}), w_1) = (\delta(q_0, w_1), \epsilon(q_0^{-1}, \lambda(q_0, w_1))) = (q_1, \epsilon(q_0^{-1}, y_1)) = (q_1, q_1^{-1})$.

Similarly, to prove the base case for $\mathcal{C}$, suppose $z_1 = \mu(q_0^{-1}, w_1)$. By construction, $w_1 = \lambda(q_0, z_1)$. Then $w_1'' = \omega_C((q_0^{-1}, q_0), w_1) = \lambda(q_0, \mu(q_0^{-1}, w_1)) = \lambda(q_0, z_1) = w_1$. Thus w'' and w begin with the same symbol. Also, $\mathcal{C}$ is in state (q_1^{-1}, q_1) because $\psi_C((q_0^{-1}, q_0), w_1) = (\epsilon(q_0^{-1}, w_1), \delta(q_0, \mu(q_0^{-1}, w_1))) = (q_1^{-1}, \delta(q_0, z_1)) = (q_1^{-1}, q_1)$

Inductive hypothesis: $w_i'' = w_i' = w_i$ for all $i \leq k < n$. Furthermore, after processing $w_1 \dots w_k$, $\mathcal{B}$ is in state (q_k, q_k^{-1}) and $\mathcal{C}$ is in state (q_k^{-1}, q_k).

Inductive step: Assume the inductive hypothesis to be true. We need to show that $w_{k+1}'' = w_{k+1}' = w_{k+1}$, and after processing $w_1 \dots w_{k+1}$, $\mathcal{B}$ is in state (q_{k+1}, q_{k+1}^{-1}) and $\mathcal{C}$ is in state

(q_{k+1}^{-1}, q_{k+1}).

Assume $\mathcal{B}$ is in state (q_k, q_k^{-1}), and $\mathcal{C}$ is in state (q_k^{-1}, q_k). We will first show $w'_{k+1} = w_{k+1}$, and then show that $w''_{k+1} = w_{k+1}$. Suppose $y_{k+1} = \lambda(q_k, w_{k+1})$. By construction, $w_{k+1} = \mu(q_k^{-1}, y_{k+1})$. Then $w'_{k+1} = \omega_B((q_k, q_k^{-1}), w_{k+1}) = \mu(q_k^{-1}, \lambda(q_k, w_{k+1})) = \mu(q_k^{-1}, y_{k+1}) = w_{k+1}$. Also, $\mathcal{B}$ is in state (q_{k+1}, q_{k+1}^{-1}) because $\psi_B((q_k, q_k^{-1}), w_{k+1}) = (\delta(q_k, w_{k+1}), \epsilon(q_k^{-1}, \lambda(q_k, w_{k+1}))) = (q_{k+1}, \epsilon(q_k^{-1}, y_{k+1})) = (q_{k+1}, q_{k+1}^{-1})$.

We can show a similar result for $\mathcal{C}$. Suppose $z_{k+1} = \mu(q_k^{-1}, w_{k+1})$. By construction, $w_{k+1} = \lambda(q_k, z_{k+1})$. Then $w''_{k+1} = \omega_C((q_k^{-1}, q_k), w_{k+1}) = \lambda(q_k, \mu(q_k^{-1}, w_{k+1})) = \lambda(q_k, z_{k+1}) = w_{k+1}$. Thus $w'' = w = w_{k+1}$. Also, $\mathcal{C}$ is in state (q_{k+1}^{-1}, q_{k+1}) because:

$$\psi_C((q_k^{-1}, q_k), w_{k+1}) = (\epsilon(q_k^{-1}, w_{k+1}), \delta(q_k, \mu(q_k^{-1}, w_{k+1}))) = (q_{k+1}^{-1}, \delta(q_k, z_{k+1})) = (q_{k+1}^{-1}, q_{k+1}).$$

Therefore, by induction, we've shown that $\mathcal{B}$ and $\mathcal{C}$ are both trivial machines, i.e., that $\mathcal{A}$ and $\mathcal{A}^{-1}$ are inverses. □

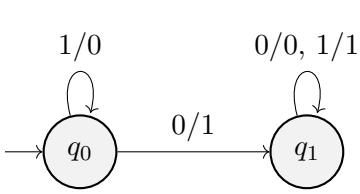
One final note on Mealy machines is that they can be *reduced* or *minimized* using an $O(n \log n)$ algorithm described by Hopcroft [5]. This algorithm takes a machine as input and outputs an equivalent machine that is guaranteed to have the minimum number of states possible. I won't describe the algorithm in this exposition or prove its correctness, but I mention this fact because it will be useful later on when we encounter the word problem.

4 Group Theory Meets Automata Theory

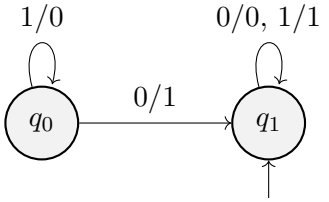
4.1 A First Example

If a Mealy machine is invertible, we can consider what it would mean to construct a group from the machine. Given a machine with n states q_0 through q_{n-1}, we can consider n copies A_0 through A_{n-1} of the machine, where the initial state of copy A_i is q_i. (That is, we are considering every state as a possible initial state of the machine.) These copies are the generators of the group. We define the binary operation of an automata group as the *composition* of machines, as defined in the previous section. Since automata encode functions from one language to another, we say that two elements in an automata group are equal if they induce the same function on input words.

For example, consider the Mealy machine from Figure 1, which can be viewed as adding 1 in binary. Intuitively, this machine is clearly invertible because binary addition is an invertible operation, its inverse being subtraction. If we call this machine $\mathcal{M}$, then the group of $\mathcal{M}$, denoted $G(\mathcal{M})$, is generated by two automata (and their inverses) – one automaton with an initial state of q_0, and another automaton with an initial state of q_1, which turns out to be the identity automaton (Figure 4).



(i) A version of $\mathcal{M}$ with initial state q_0



(ii) A version of $\mathcal{M}$ with initial state q_1

Figure 4: The automata created by considering multiple possible initial states for $\mathcal{M}$ from Figure 1. The machine on the left is an exact copy of $\mathcal{M}$. The machine on the right has q_1 as its initial state. It maps 0 to 0 and 1 to 1; therefore, it's the trivial automaton.

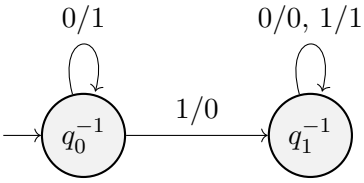


Figure 5: The inverse of $\mathcal{M}$, which represents binary subtraction by 1. For example, consider the input 00101, which will be transformed into 11001. 00101, read backwards, represents 20 in decimal; 11001, read from backwards, represents $19 = 20 - 1$ in decimal.

Using the inversion procedure described above, we can construct the inverse of $\mathcal{M}$ (Figure 5), which, intuitively, should represent binary subtraction by 1. At this point, we've found at least three elements in $G(\mathcal{M})$: $\mathcal{M}, \mathcal{M}^{-1}$, and the identity automaton. If we create a new automaton $\mathcal{M}^n$ by composing n copies of $\mathcal{M}$, this new machine represents addition by n in binary. Similarly, if we create a new automaton $(\mathcal{M}^{-1})^n$ by composing n copies of $\mathcal{M}^{-1}$, this new machine represents *subtraction* by n in binary. This group is infinite cyclic, because it's isomorphic to the integers.

4.2 A Classification Problem

Consider all possible (invertible) Mealy machines one could create given two states and an alphabet containing two letters, 0 and 1. We will associate an output function with each state. One of the states will be associated with identity function Id. The other state will be associated with ϵ, which flips the input, mapping 0 to 1 and 1 to 0.

We can encode each state of these machines using the following notation. Suppose that in state q_i, an input of 0 sends the machine to q_j and an input of 1 sends the machine to q_k. Let t be the output function associated with state q_i. We can represent q_i compactly by writing $q_i = (q_j, q_k)t$. (Throughout this paper, I'll use the the notation g_{q_i} to represent the transition function over state q_i, so that $q_i = (g_{q_i}(0), g_{q_i}(1))t$). For example, consider the binary addition machine in Figure 1. State q_0 corresponds to ϵ and state q_1 corresponds to Id. We could encode the binary

adder by writing $q_0 = (q_1, q_0)\epsilon$ and $q_1 = (q_1, q_1)\,\mathrm{Id}$.

It turns out that this rather strange notation is motivated by a concept from group theory: the wreath product.

Definition 4.1. Let G be a finitely-generated group, and let S_d be the symmetric group acting on the set $D = \{0, 1, \ldots, d-1\}$. We define the **wreath product** $G \wr_D S_d$ as a group that consists of pairs of the form (g, σ), where:

- g maps from G to D. Furthermore, there are finitely many elements $x \in G$ for which $g(x) \neq id_G$. That is, $g(y) = id_G$ for all choices of y barring a finite number of exceptions.
- $\sigma \in S_d$, i.e., σ is a permutation that acts on D.

Within the wreath product, we define the multiplication of two elements as follows

$$(g_1, \sigma_1)(g_2, \sigma_2) = (g_3, \sigma_1\sigma_2)$$

where $g_3(x) = g_1(x)g_2(\sigma_1(x))$.

As a matter of convention, we will write elements of the wreath product in the following form, rather than in the (g, σ) pair form given in the definition. Suppose $g(0) = a_0$, $g(1) = a_1$, $\ldots$, $g(d-1) = a_{d-1}$. Then we would represent the pair (g, σ) as $(a_0, a_1, \ldots, a_{d-1})\sigma$. This is exactly the notation described above to encode each of state of a machine!

The notion of wreath composition is useful for proving the following theorem.

Theorem 4.1. *There are exactly six groups generated by two-state automata over two letters:*

1. *The trivial group*
2. *$\mathbb{Z}/2\mathbb{Z}$, the group of order 2*
3. *$\mathbb{Z}_2 \oplus \mathbb{Z}_2$, the Klein four-group*
4. *$\mathbb{Z}$, the infinite cyclic group*
5. *D_∞, the infinite dihedral group*
6. *$(\oplus_{\mathbb{Z}}\mathbb{Z}/2\mathbb{Z}) \rtimes \mathbb{Z}$, the lamplighter group*

Proof. Consider an arbitrary two-state machine $\mathcal{M}$. Call its states a and b. We proceed by casework on a and b.

First, suppose $a = b$. If the label on the state is Id, then the corresponding group is the trivial group. If the label on the state is ϵ, then the machine "flips" each bit (Figure 6). It's clear that this machine is its own inverse, so the group generated by this machine is $\mathbb{Z}/2\mathbb{Z}$, the group of order 2.

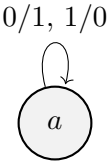


Figure 6: The machine where $a = b$ and the output function is ϵ.

Now consider the case where $a \neq b$. Without loss of generality, suppose a corresponds to the output function Id, and b corresponds to the output function ϵ. Then either $a = (a, a)\,\mathrm{Id}$ or $a = (a, b)\,\mathrm{Id}$ or $a = (b, a)\,\mathrm{Id}$ or $a = (b, b)\,\mathrm{Id}$, and either $b = (a, a)\epsilon$ or $b = (a, b)\epsilon$ or $b = (b, a)\epsilon$ or $b = (b, b)\epsilon$.

Case 1: $a = (a, a)\,\mathrm{Id}$. If this is the case, then there are four possible automata, depicted in Figure 7. Consider the group generated by each possible machine.

Case 1.i: $a = (a, a)\,\mathrm{Id}$ and $b = (a, a)\epsilon$. Then $\mathcal{M}_a$ is the identity machine, and $\mathcal{M}_b$ is the machine that flips the first bit of an input, then copies the rest of the input. $\mathcal{M}_b$ is its own inverse, so $G(\mathcal{M})$ is isomorphic to $\mathbb{Z}/2\mathbb{Z}$.

Case 1.ii: $a = (a, a)\,\mathrm{Id}$ and $b = (a, b)\epsilon$. Then $\mathcal{M}_a$ is the identity machine, and $\mathcal{M}_b$ is the binary adder described previously. We already described the group corresponding to $G(\mathcal{M})$ – it's the infinite cyclic group, $\mathbb{Z}$.

Case 1.iii: $a = (a, a)\,\mathrm{Id}$ and $b = (b, a)\epsilon$. Then $\mathcal{M}_a$ is the identity machine, and $\mathcal{M}_b$ is the inverse of the binary adder described previously (i.e., it is the binary *subtractor*). Thus $G(\mathcal{M})$ is the infinite cyclic group, $\mathbb{Z}$.

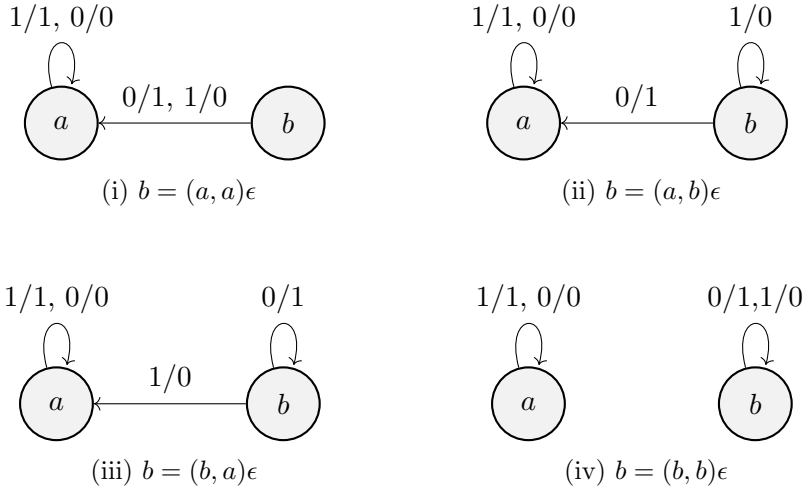
Case 1.iv: $a = (a, a)\,\mathrm{Id}$ and $b = (b, b)\epsilon$. Then $\mathcal{M}_a$ is the identity machine, and $\mathcal{M}_b$ flips each bit in its input. Clearly, $\mathcal{M}_b$ is its own inverse, so $G(\mathcal{M})$ is $\mathbb{Z}/2\mathbb{Z}$.

Case 2: $a = (a, b)\,\mathrm{Id}$. If this is the case, then there are four possible automata, depicted in Figure 8.

Case 2.i: $a = (a, b)\,\mathrm{Id}$ and $b = (a, a)\epsilon$. First, notice that a and b both have order 2. Clearly, $a^2 = (a^2, b^2)$. Since b has the output function ϵ, b^2 will have the output function Id. We compute g_{b^2} below:

$$g_{b^2}(0) = g_b(0)g_b(\epsilon(0)) = g_b(0)g_b(1) = a^2$$
$$g_{b^2}(1) = g_b(1)g_b(\epsilon(1)) = g_b(1)g_b(0) = a^2$$

Therefore, $b^2 = (a^2, a^2)$. Both a^2 and b^2 act trivially on any word, so the order of both a



(i) $b = (a, a)\epsilon$ (ii) $b = (a, b)\epsilon$

(iii) $b = (b, a)\epsilon$ (iv) $b = (b, b)\epsilon$

Figure 7: All possible automata where $a = (a, a)\,\mathrm{Id}$

and b is 2.

Now, consider the element $a^{-1}b = ab$ since $a^{-1} = a$. The output function over $a^{-1}b$ is ϵ, and we can compute $g_{a^{-1}b}$:

$$g_{a^{-1}b}(0) = g_{ab}(0) = g_a(0)g_b(0) = a^2 = 1$$
$$g_{a^{-1}b}(1) = g_{ab}(1) = g_a(1)g_b(1) = ba$$

So $a^{-1}b = (1, ba)\epsilon$. What is the order of this element? Clearly, it cannot be odd, because for any $n \in \mathbb{N}$, $(a^{-1}b)^{2n+1}$ would have the output function $\epsilon^{2n+1} = \epsilon \neq \mathrm{Id}$. Now consider whether the order of $a^{-1}b$ could be even. Notice that $(a^{-1}b)^2 = (ba, ba)$:

$$g_{(ab)^2}(0) = g_{ab}(0)g_{ab}(\epsilon(0)) = g_{ab}(0)g_{ab}(1) = ba$$
$$g_{(ab)^2}(1) = g_{ab}(1)g_{ab}(\epsilon(1)) = g_{ab}(1)g_{ab}(0) = ba$$

Also, $ba = (a^{-1}b)^{-1}$, so ba and $a^{-1}b$ should have the same order. Suppose the order of ab is $2n$ for some integer n. Then $(a^{-1}b)^{2n} = ((ba)^n, (ba)^n)$ is the identity. But then this implies that $n = 2n$, which only holds for $n = 0$. The order of a group element must be positive, so $a^{-1}b$ must have infinite order. Given the fact that $|a| = |b| = 2$ and $a^{-1}b$ has infinite order, $G(\mathcal{M})$ is the infinite dihedral group.

Case 2.ii: $a = (a, b)\,\mathrm{Id}$ and $b = (a, b)\epsilon$. It can be shown that in this case, the group generated corresponds to the lamplighter group, $(\oplus_{\mathbb{Z}}\mathbb{Z}/2\mathbb{Z}) \rtimes \mathbb{Z}$. I will not go into detail about the lamplighter group in this paper.

Case 2.iii: $a = (a, b)\,\mathrm{Id}$ and $b = (b, a)\epsilon$. Similar to Case 2.ii, it can be shown that the group generated in this case corresponds to the lamplighter group, $(\oplus_{\mathbb{Z}}\mathbb{Z}/2\mathbb{Z}) \rtimes \mathbb{Z}$.

Case 2.iv: $a = (a, b)\,\mathrm{Id}$ and $b = (b, b)\epsilon$. First, we show that both a and b have order 2. The output function for b^2 is $\epsilon^2 = \mathrm{Id}$, and we can also see that $g_{b^2}(x) = b^2$ for all x

$$g_{b^2}(0) = g_b(0)g_b(\epsilon(0)) = g_b(0)g_b(1) = b^2$$
$$g_{b^2}(1) = g_b(1)g_b(\epsilon(1)) = g_b(1)g_b(0) = b^2$$

So $b^2 = (b^2, b^2)\,\mathrm{Id}$ is the identity, implying that $|b| = 2$. Similarly, the output function for a^2 is Id, and we can determine $g_{a^2}(x)$:

$$g_{a^2}(0) = g_a(0)g_a(0) = a^2$$
$$g_{a^2}(1) = g_a(1)g_a(1) = b^2$$

Therefore, $a^2 = (a^2, b^2)\,\mathrm{Id}$. If the machine starts in a^2, it will process any word in the same manner that it would be processed if the machine had started in state b^2. Thus, $a^2 = b^2$, i.e., $|a| = 2$.

Now consider the element $a^{-1}b$, which has output function ϵ. $a^{-1} = a$ since $|a| = 2$. We can determine $g_{a^{-1}b}$:

$$g_{a^{-1}b}(0) = g_a(0)g_b(0) = ab = a^{-1}b$$
$$g_{a^{-1}b}(1) = g_a(1)g_b(1) = b^2 = 1$$

Therefore, $a^{-1}b = (a^{-1}b, 1)\epsilon$. Set $x = a^{-1}b$. We now determine the order of x. x cannot have odd order, since the output function for x^{2n+1} is $\epsilon^{2n+1} = \epsilon \neq \mathrm{Id}$ for all n. Suppose x has even order. Notice that $x^2 = (x, x)$:

$$g_{x^2}(0) = g_x(0)g_x(\epsilon(0)) = a^{-1}b = x$$
$$g_{x^2}(1) = g_x(1)g_x(\epsilon(1)) = a^{-1}b = x$$

Therefore, for all n, $x^{2n} = (x^n, x^n)$. Suppose x has order $2k$. Then $x^{2k} = (x^k, x^k)$ is the identity. But then this would imply that x actually has order k. The only possible value for k is 0, but order must be positive, so $x = a^{-1}b$ must not have finite order. Given that $a^{-1}b$ has infinite order and $|a| = |b| = 2$, $G(\mathcal{M}) = \langle a, b\rangle$ is the infinite dihedral group.

Case 3: $a = (b, a)\,\mathrm{Id}$. This case is equivalent to Case 2 if we swap 0s and 1s.

Case 4: $a = (b, b)\,\mathrm{Id}$. If this is the case, then there are four possible automata, depicted in Figure 9.

Case 4.i: $a = (b, b)\,\mathrm{Id}$ and $b = (a, a)\epsilon$. Then $\mathcal{M}_a$ is the machine that flips every other bit, starting with the second bit and $\mathcal{M}_b$ is the machine that flips every other bit, starting with the first bit. $\mathcal{M}_a$ is its own inverse and $\mathcal{M}_b$ is its own inverse. Composing them in

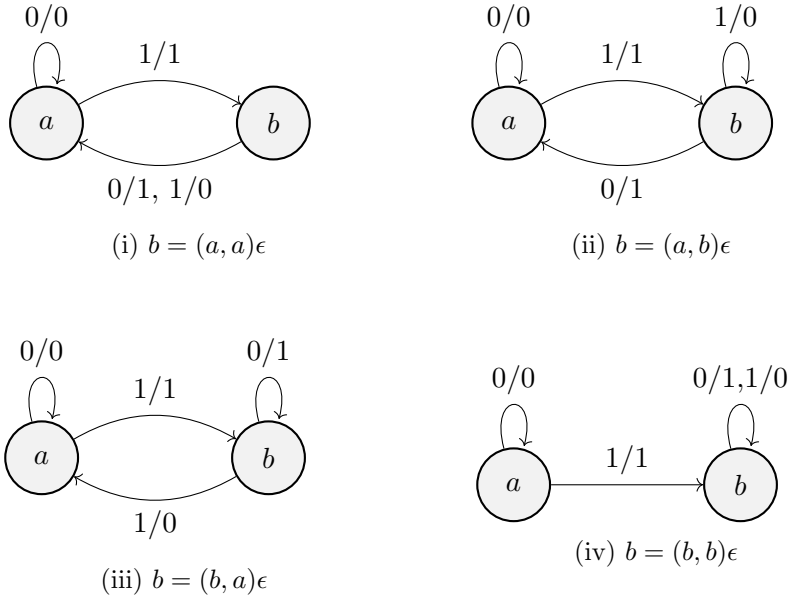


Figure 8: All possible automata where $a = (a, b)\,\mathrm{Id}$

either order gives the machine which flips every bit of the inputted word; this machine is its own inverse. $G(\mathcal{M})$ is therefore the Klein four-group, $\mathbb{Z}_2 \oplus \mathbb{Z}_2$, because

$$G(\mathcal{M}) = \langle \mathcal{M}_a, \mathcal{M}_b \mid \mathcal{M}_a^2 = \mathcal{M}_b^2 = (\mathcal{M}_a \mathcal{M}_b)^2 = e \rangle.$$

Case 4.ii: $a = (b, b)\,\mathrm{Id}$ and $b = (a, b)\epsilon$. First, we compute ab and ba. The output function for both ab and ba will be ϵ, but what about the transition functions g_{ab} and g_{ba}? We can determine each function explicitly:

$$\begin{aligned}
g_{ab}(0) &= g_a(0) g_b(0) = ba \\
g_{ab}(1) &= g_a(1) g_b(1) = b^2 \\
g_{ba}(0) &= g_b(0) g_a(\epsilon(0)) = g_b(0) g_a(1) = ab \\
g_{ba}(1) &= g_b(1) g_a(\epsilon(1)) = g_b(1) g_a(0) = b^2
\end{aligned}$$

So $ab = (ba, b^2)\epsilon$ and $ba = (ab, b^2)\epsilon$. It turns out that these elements are the same, i.e., that a and b commute. We can see that $ab = ba$ by considering how each state would transform any possible word. For a word consisting entirely of 1s, it's clear that the word will be transformed into all 0s regardless of whether the machine starts in state ab or ba, because it will toggle back and forth between ab and ba, which both have the output function ϵ. Now consider an input word containing at least one 0. The word is prefixed by some number of 1s (possibly zero). While processing the 1s, the machine (whether it starts in state ab or ba) will toggle back and forth between ab and ba, outputting 1s. When the reaches the first 0 in the input word, it will output a 1 and transition to state b^2. At that point, the rest of the word will be processed the same, regardless of whether the machine started in state ab or ba. Therefore, for all possible words, we've shown that

$ab = ba$, i.e., that a and b commute.

Also remark that b^2a is the identity because its output function is $\epsilon^2 = \text{Id}$ and its next state is b^2a regardless of input:

$$g_{b^2a}(0) = g_b(0)g_{ba}(\epsilon(0)) = g_b(0)g_{ba}(1) = ab^2 = b^2a$$
$$g_{b^2a}(1) = g_b(1)g_{ba}(\epsilon(1)) = g_b(1)g_{ba}(0) = b(ab) = b^2a$$

Since b^2a is the identity, $b^2 = a^{-1}$. We now deduce the order of b. b must have either finite or infinite order. By way of contradiction, suppose that b has finite order. Then the order of b must be odd or even. b cannot have odd order because for any $n \in \mathbb{N}$, the output function for b^{2n+1} will be $\epsilon^{2n+1} = \epsilon \neq \text{Id}$. So if b has finite order, it must have even order. But this also leads to a contradiction. Suppose the order of b is $2n$ for some $n \in \mathbb{N}$. Then $(b^2)^n = (a^{-1})^n = e$, so $|a^{-1}| = |a| = n = \frac{|b|}{2}$. Since $a = (b, b)$, it follows that $a^n = (b^n, b^n)$, which must be the identity since $|a| = n$. But this would imply that $n = 2n$, which is only possible for $n = 0$. However, order must be positive, so this is impossible. Therefore, the order of b cannot be finite, so it is infinite. So we have $G(\mathcal{M}) = \langle a, b\rangle = \langle b\rangle$ since $(b^{-1})^2 = a$. Since we have shown that b has infinite order, $G(\mathcal{M})$ is therefore the infinite cyclic group.

Case 4.iii: $a = (b, b)\,\text{Id}$ and $b = (b, a)\epsilon$. This case is equivalent to Case 4.ii if we swap 0s and 1s.

Case 4.iv: $a = (b, b)\,\text{Id}$ and $b = (b, b)\epsilon$. Then $\mathcal{M}_a$ is the machine that copies the first bit of input and then flips all of the other bits. $\mathcal{M}_b$ flips every bit of input. $\mathcal{M}_a$ is its own inverse and $\mathcal{M}_b$ is its own inverse. Composing them in either order results in a machine that flips the first bit of input, then copies all other bits; this machine is its own inverse. $G(\mathcal{M})$ is therefore the Klein four-group, by the same reasoning used in Case 4.i.

In each case enumerated above, the group generated by $\mathcal{M}$ corresponded to one of six groups: the trivial group, the group of order 2, the Klein four-group, the infinite cyclic group, the infinite dihedral group, and the lamplighter group. Thus, we've classified all possible groups generated by automata with two states and an input/output alphabet with two letters. □

5 The Word Problem

Consider the generators of a finitely-generated group G. Consider any "word," written as the product of elements in the generating set and their inverses. The word problem is the problem of determining in a finite number of steps whether or not this word represents the identity element. It has been proven by Pyotr Novikov [6] and William Boone [2] that in general, the word problem is undecidable. Remarkably, the word problem is solvable for groups generated by automata.

Theorem 5.1. *The word problem is solvable for groups generated by automata.*

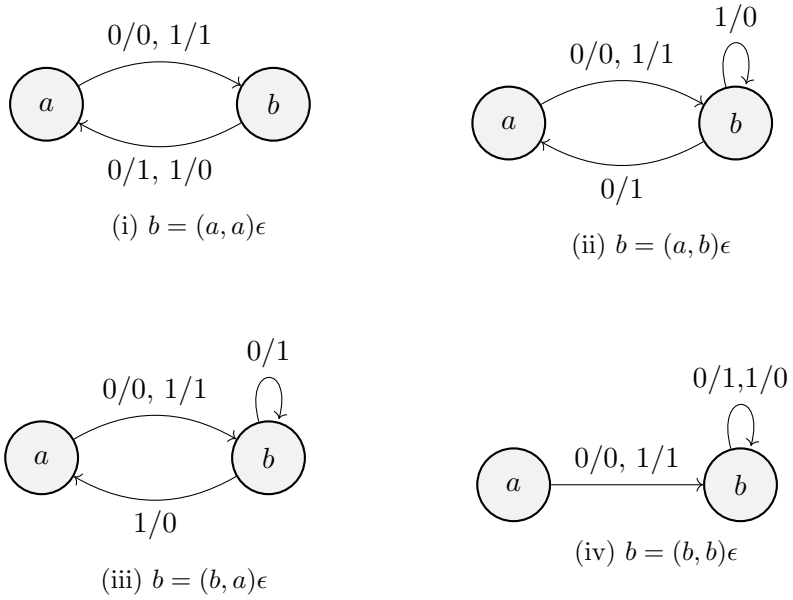


Figure 9: All possible automata where $a = (b, b)\,\mathrm{Id}$

Proof. Consider an arbitrary invertible automaton $\mathcal{A}$. For each state $q \in Q$, the state set of $\mathcal{A}$, consider the automaton $\mathcal{A}_q$ with initial state q. Also consider this machine's inverse, $(\mathcal{A}_q)^{-1}$. Use a minimization algorithm to minimize $\mathcal{A}_q$ and $(\mathcal{A}_q)^{-1}$ into corresponding minimized machines, denoted $\mathcal{M}_q$ and $(\mathcal{M}_q)^{-1}$, respectively. Let C represent the maximum number of states in $\mathcal{M}_q$ and $(\mathcal{M}_q)^{-1}$ for each $q \in Q$. Consider any word $w = \mathcal{L}_{q_1} \circ \ldots \circ \mathcal{L}_{q_n} \in G(\mathcal{A})$ where each $\mathcal{L}_{q_i} \in \{\mathcal{A}_q \mid q \in Q\}$ or $(\mathcal{L}_{q_i})^{-1} \in \{\mathcal{A}_q \mid q \in Q\}$. Then this word can be written equivalently as $w = \mathcal{N}_{q_1} \circ \ldots \cdot \mathcal{N}_{q_n}$ where each $\mathcal{N}_{q_i} \in \{\mathcal{M}_q \mid q \in Q\}$ or $(\mathcal{N}_{q_i})^{-1} \in \{\mathcal{M}_q \mid q \in Q\}$. Then w has at most C^n states (i.e., it has a finite number of states).

Suppose w is trivial, i.e., for any inputted word v, the output under w is also v. Then for every state r reachable from the initial state, it must be the case that the output function $\lambda(r, x) = x$ for all $x \in \Sigma$, the input/output alphabet.

Now suppose that for every state r reachable from the initial state, the output function $\lambda(r, x) = x$ for all $x \in \Sigma$. Then w acts trivially on any word in Σ^*, i.e., w is trivial.

Thus, we've established a way to determine whether a word in $G(\mathcal{A})$ is trivial, so the word problem is solvable in $G(\mathcal{A})$. □

6 Acknowledgements

I am very grateful for my advisors for this project, Andrew Bridy and Anna Gilbert. I would also like to thank Igor Frenkel, who taught me in MATH 350 last semester.

References

[1] Laurent Bartholdi and Pedro V. Silva. "Groups defined by automata". In: CoRR abs/1012.1531 (2010). arXiv: 1012.1531. URL: http://arxiv.org/abs/1012.1531.

[2] William W. Boone. "The word problem". In: Proceedings of the National Academy of Sciences 44.10 (1958), pp. 1061–1065. DOI: 10.1073/pnas.44.10.1061.

[3] R. Grigorchuk, Volodymyr Nekrashevych, and V. Sushchanskii. "Automata, dynamical systems, and groups". In: Proceedings of the Steklov Institute of Mathematics 231 (Jan. 2000), pp. 128–203.

[4] Rostislav Grigorchuk and Andrzej Żuk. Automata groups, their spectra and classification.

[5] John E. Hopcroft. An n log n algorithm for minimizing states in a finite automaton. 1971. URL: http://i.stanford.edu/pub/cstr/reports/cs/tr/71/190/CS-TR-71-190.pdf.

[6] P. S. Novikov. "Algorithmic Unsolvability of the Word Problem in Group Theory". In: Journal of Symbolic Logic 23.1 (1958), pp. 50–52. DOI: 10.2307/2964487.