

# THE WORD PROBLEM FOR AUTOMATA GROUPS

ARDEN RASMUSSEN

## 1. INTRODUCTION

The word problem in group theory is the problem of deciding if two representations of generators represent the same element. A more specific example of this is to determine if an element is equivalent to the identity element. For an arbitrary group, this is not necessarily possible, however, for automata groups, it has been proven that it is a solvable problem. The main process that is used to show the solvability of the word problem is the Knuth-Bendix algorithm, and the automaton structure of groups.

## 2. FINITE STATE AUTOMATA

We will first begin with some definitions that are used in the description of automatic groups. Firstly  $\Sigma$  is the finite set of *letters*, this is commonly called the *alphabet*. The free monoid generated by  $\Sigma$  will be denoted as  $\Sigma^*$ , and the elements of  $\Sigma^*$  are commonly called words. The free monoid  $\Sigma^*$  can be thought of as the set of all possible combinations of letters in  $\Sigma$ . We will also include the identity element of  $\Sigma^*$  to be denoted as  $\epsilon$ . For example consider the alphabet

$$\Sigma = \{a, b\} \Rightarrow \Sigma^* = \{a, b, aa, ab, ba, bb, \dots\}.$$

The product of two strings  $u, v \in \Sigma^*$  is denoted simply as  $uv$  and represents the concatenation of the strings. We will also make use of the notation  $|w|$  to mean the length of a string for some string  $w \in \Sigma^*$ . A subset of  $\Sigma^*$  is called a *language*, and a subset of  $\Sigma^* \times \Sigma^*$  is called a *relation*.

In our case we can commonly view the alphabet  $\Sigma$  to be the set of generators  $X$  for some group  $G$  along with their inverses, so we can write  $\Sigma = X \cup X^{-1}$ . Then  $\Sigma^*$  is the set of products of generators, of arbitrary length.

The other key component of an automaton is a *state*. For the scope of this paper, a state can be considered to be an element of the group.

A Finite State Automaton is defined as the quintuple  $(\Sigma, S, s_0, \delta, F)$ . Where  $\Sigma$  is the alphabet of symbols,  $S$  is a non-empty finite set of states,  $s_0$  is the initial state such that  $s_0 \in S$ ,  $\delta$  is the state-transition function  $\delta : S \times \Sigma \rightarrow S$ , and  $F$  is the finite set of states  $F \subseteq S$ , and  $F$  could be the empty set. The set  $F$  is the set of states that we want the automaton to recognize, for example if  $F = \{1\}$ , will mean that the automaton recognizes representations of the identity element.

Since the automaton that we will be working with are deterministic, then there is exactly one output for every input. So we can consider the automaton  $A$  as a mapping  $A : \Sigma^* \rightarrow S$ .

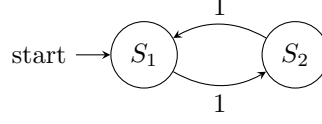


FIGURE 1. Very simple Finite state automata

Consider the Finite State Automaton presented in figure 1. For this automaton  $\Sigma = \{0, 1\}$ ,  $S = \{S_1, S_2\}$ ,  $s_0 = S_1$ , and

$$\delta(s, l) = \begin{cases} S_1 & \text{if } s = S_2 \text{ and } l = 1 \\ S_2 & \text{if } s = S_1 \text{ and } l = 1 \end{cases}.$$

Note that the explicit format for  $\delta$  can be relatively complex, so it is common to express the transition function as a table. Thus the transition table for this automaton is given in Table 1.

TABLE 1. Transition table for Figure1.

| $\delta$ | 1     |
|----------|-------|
| $S_1$    | $S_2$ |
| $S_2$    | $S_1$ |

We can use this table to determine the output of the finite state automaton. Let us consider the example where  $w \in \Sigma^*$ , with  $w = 1111$ . Running the automaton with this input of  $w$  we see that each stage is described below

- |                          |                              |
|--------------------------|------------------------------|
| (1) $s = S_1, w = 1111.$ | (4) $s = S_2, w = 1.$        |
| (2) $s = S_2, w = 111.$  | (5) $s = S_1, w = \epsilon.$ |
| (3) $s = S_1, w = 11.$   |                              |

Thus after the automaton has been run on the input of  $w$  the output of the automaton is  $S_1$ . We can see that this automaton is actually representative of  $\mathbb{Z}/2\mathbb{Z}$ , when we replace  $S_1$  with 0, and  $S_2$  with 1.

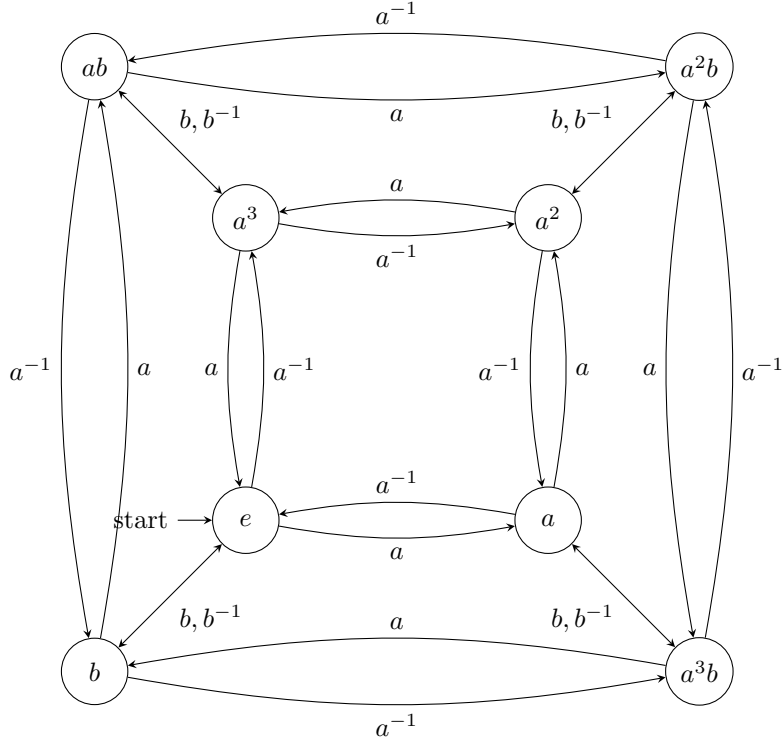
To construct an automaton from a group, let us consider the group  $D_4$ , the generators for  $D_4$  are given as  $\langle a, b | a^4 = 1, b^2 = 1, (ab)^2 = 1 \rangle$ . We will denote the states, as the most reduced form of their generators, thus  $S = \{e, a, a^2, a^3, b, ab, a^2b, a^3b\}$ . Now we consider our  $X = \{a, b\}$ . Then we find  $\Sigma = \{a, a^{-1}, b, b^{-1}\}$ . And finally the transition table for  $D_4$  is given in table 2;

Using this table, and the states, we can construct the diagram representing the automaton for  $D_4$ .

With this automaton representation of  $D_4$  we can consider any sequence of generators  $w \in \Sigma^*$ , and determine the reduced representation of this element, we just need to apply the automaton as we did in the prior example. Lets consider  $w = aabaaba^{-1}bb^{-1}aab$ . By applying the same process as before, we find that  $w = a^3b$ .

TABLE 2. Transition table for  $D_4$ 

| $\delta_{D_4}$ | $a$    | $a^{-1}$ | $b$    | $b^{-1}$ |
|----------------|--------|----------|--------|----------|
| $e$            | $a$    | $a^3$    | $b$    | $b$      |
| $a$            | $a^2$  | $e$      | $a^3b$ | $a^3b$   |
| $a^2$          | $a^3$  | $a$      | $a^2b$ | $a^2b$   |
| $a^3$          | $e$    | $a^2$    | $ab$   | $ab$     |
| $b$            | $ab$   | $a^3b$   | $e$    | $e$      |
| $ab$           | $a^2b$ | $b$      | $a^3$  | $a^3$    |
| $a^2b$         | $a^3b$ | $ab$     | $a^2b$ | $a^2b$   |
| $a^3b$         | $b$    | $a^2b$   | $a$    | $a$      |

FIGURE 2. Finite state automaton for  $D_4$ 

### 3. THE WORD PROBLEM

The common form of the word problem is given two representations of elements in a set, determine if the two expressions represent the same element. As we saw in the previous section, the finite state automaton can represent a mapping from elements of  $\Sigma^*$  to states  $S$ , where we defined each state as equivalent to an element of the group. Thus the formulation of the word problem in the context of Automatic groups is

**Theorem 3.1.** For a finitely generated group  $G$ , and two representations of an element in that group  $w, u, \in \Sigma^*$ . And an automaton  $A = (\Sigma, G, e)$ . Then it is possible to state whether  $w = u$ .

An equivalent definition for the word problem is to consider it is a problem of rewriting elements in some group  $G$ . For some set of generators  $x, y, z, \dots$  for  $G$ , we introduce one letter for  $x$  and another for  $x^{-1}$ . We will then call these letters the alphabet  $\Sigma$  for the problem. With this definition every element in  $G$  is represented in some way by a product  $abc \cdots pqr$  of symbols from  $\Sigma$  of some length. The string of length 0 is the representative of the identity of  $G$ . With this construction the question is to be able to recognize all different ways to express the identity  $e$  of  $G$ , given some relations.

At first this may appear trivial, to state whether two elements are the same, or not. However, it has been proven that the word problem is not universally solvable. That is to say, given some arbitrary group, it may not be possible to say if two representations are the same element or not. However, given an Automatic group it is solvable.

To get some intuition to the motivation of the word problem, consider for  $D_4$ ,  $w = aababa^{-1}bbabbaab^{-1}a^{-1}b$  and  $u = aba^{-1}a^{-1}baaab^{-1}a^{-1}a^{-1}b$ , although these appear very different, these are actually equivalent and both  $w$  and  $u$  represent  $e$ .

#### 4. CYCLIC GROUP OF ORDER 3

Proving the word problem is solvable for the Cyclic group of order 3 is relatively simple, and so we will use it to explain the concept more generally.

$$(1) \quad Z_3 = \langle x | x^3 = 1 \rangle$$

In this group, our alphabet consists of  $\Sigma = \{x, x^{-1}\}$ , and so  $\Sigma^*$  is the set of all strings combining any number of the letters  $x$  and  $x^{-1}$ . The first step is to write out the relations that we know. The relations in  $Z_3$  provide us with a list of strings that can be inserted or canceled out whenever they appear in the string, without altering the represented element of the group. In this case, the relations are listed below.

$$\begin{array}{ll} (1) \quad xxx = e & (4) \quad x^{-1}x^{-1}x^{-1} = e \\ (2) \quad xx^{-1} = e & (5) \quad x^{-1} = xx \\ (3) \quad x^{-1}x = e & (6) \quad x^{-1}x^{-1} = x \end{array}$$

The first relation is derived directly from the definition of  $Z_3$ . The second and third come from the definition of an inverse of the element  $x$ . Relation 4 is a result that since the cube of  $x$  is the identity, then so is the cube of the inverse of  $x$ . Relation 5 and 6 come as a result of multiplying by  $e$ , and from relation 1, expanding this out to be of the form  $xxxx^{-1}$  and  $xxxx^{-1}x^{-1}$  respectively, then by relation 2 we cancel elements and are left with  $xx$  and  $x$  respectively.

With these relations we can reduce any string to either the empty string  $e$ ,  $x$ , or  $xx$ . And thus the word problem is solvable for  $Z_3$ . Let us test this by considering

$$w = xxx^{-1}xx^{-1}x^{-1}xxxx.$$

$$w = xxx^{-1}xx^{-1}x^{-1}xxxx$$

$$(2) \rightarrow xxxx^{-1}x^{-1}xxxx$$

$$(1) \rightarrow x^{-1}x^{-1}xxxx$$

$$(3) \rightarrow x^{-1}xxx$$

$$(3) \rightarrow xx$$

Thus we see that  $w = xx = x^2$ .

## 5. AUTOMATIC GROUP STRUCTURE

This process cannot be done for any arbitrary set, or for that matter any automaton. This is why we must use the Knuth-Bendix method to formulate the automaton into a structure that this process will work.

## 6. SHORTLEX ORDERING

Before we are able to use the Knuth-Bendix method, we need to have some notion of ordering for our representations. In our case we will simply use shortlex ordering. Shortlex ordering is alphabetical ordering, where shorter words are considered smaller. So given the alphabet  $\Sigma = \{x, y\}$ . Then the shortlex ordering is represented by

$$\varepsilon < x < y < xx < xy < yx < yy < \dots$$

where  $\varepsilon$  is the empty string.

## 7. KNUTH-BENDIX METHOD

The Knuth-Bendix method is an algorithm that is used to construct the relations that we will use to solve the word problem. The general concept of the algorithm is that given a set of equations between terms, it will attempt to construct a rewriting system that is equivalent to the original method for which it has been written. The new writing system is constructed such that only  $e = e$ , and no other presentation of elements in the group  $G$  is equivalent to  $e$ . Thus if it the Knuth-Bendix algorithm succeeds, then the word problem has been solved, for that group.

Note, even if the Knuth-Bendix algorithm does not succeed, this does not mean that the word problem is unsolvable for the group. There are other methods that can be used to solve the algorithm, which may work.

In our example of  $Z_3$ , we took our original writing system that consisted of some number of  $x$  and  $x^{-1}$  in any order, and rewrote it into one of  $\{e, x, x^2\}$ . However, in our example we constructed the relations manually. In more complex groups constructing all of the relations manually would not be feasible, and thus we would use the Knuth-Bendix method to construct the relations, that would then in turn be used to rewrite the strings into the new writing systems. Once the strings have been rewritten, it is trivial to check if it is the identity, and thus the word problem has been solved.

A monoid is a pretty much a group, without the gaurentee of inverses.

Confluence is its own thing, I need to expand more on.

Consider the finitely presented monoid  $M = \langle \Sigma | R \rangle$ , where  $\Sigma$  is the set of generators, and  $R$  is the set of relations. First we will consider the set of all possible words denoted  $\Sigma^*$ . Now we apply the concept of shortlex ordering to  $\Sigma^*$  to define an ordering on  $\Sigma^*$ , which we will denote using  $<$ .

The first step in the algorithm is to construct our initial set of relations. To do this consider  $P_i = Q_i \in R$ , without loss of generality assume  $Q_i < P_i$ , then we define the relation  $P_i \rightarrow Q_i$ , for all  $i$ .

The next step is to progressively construct new relations to remove the dependence on relations that do not preserve confluence. Consider some  $P_i, P_j$  with  $i \neq j$ , that have some overlap. There are two cases for  $P_i$  and  $P_j$  to overlap.

- (1) Either the prefix of  $P_i$  is equal to the suffix of  $P_j$  or the reverse is true. We can write  $P_i = BC$  and  $P_j = AB$  in the first case and  $P_i = AB$  and  $P_j = BC$  in the second.
- (2) Either  $P_i$  is contained entirely within  $P_j$  or  $P_j$  is contained within  $P_i$ . In this case we write  $P_i = B$ , and  $P_j = ABC$  in the first case, and  $P_i = ABC$  and  $P_j = B$  in the second.

Now we reduce the word given by  $ABC$  by using  $P_i$  and call this result  $r_i$ . We do the same for  $P_j$  to get  $r_j$ . If  $r_i \neq r_j$ , then we have a new relation which we will define by

$$\max\{r_i, r_j\} \rightarrow \min\{r_i, r_j\}$$

After adding this new rule, remove any relations that have a reducible left side. This process is repeated until no relations have reducible left sides.

**7.1. Example.** Let us consider a very simplistic example and use the Knuth-Bendix algorithm to rewrite the relations. We will consider the monoid given by

$$\langle x, y | x^3 = y^3 = (xy)^3 = 1 \rangle.$$

To begin with there are three reductions that have been defined

$$x^3 \rightarrow 1 \quad (1)$$

$$y^3 \rightarrow 1 \quad (2)$$

$$xyxyxy \rightarrow 1 \quad (3)$$

Considering  $P_1$  and  $P_3$  we see that there is some overlap, so we will consider the word  $x^3xyxy$ , and attempt to reduce that.

$$x^3xyxy \xrightarrow{(1)} xyxy \quad x^3xyxy \xrightarrow{(3)} x^2$$

since we cannot reduce  $xyxy$  or  $x^2$  further with our given relations, we must construct a new relation. By shortlex ordering  $x^2 < xyxy$ , so this will be given by

$$xyxy \rightarrow x^2 \quad (4)$$

Now we repeat the process with  $P_2$  and  $P_3$  and consider the word  $xyxyxy^3$ .

$$xyxyxy^3 \xrightarrow{(1)} xyxyx \quad xyxyxy^3 \xrightarrow{(3)} y^2$$

Thus resulting in the relation

$$xyxyx \rightarrow y^2 \quad (5)$$

Now there are no other existing overlaps. We first notice that  $xyxyxy$  can be reduced, so we eliminate that relation. Now the set of relations is given by

$$x^3 \rightarrow 1 \quad (1)$$

$$y^3 \rightarrow 1 \quad (2)$$

$$yxyxy \rightarrow x^2 \quad (4)$$

$$xyxyx \rightarrow y^2 \quad (5)$$

Now we repeat the entire process again.

Considering  $P_1$  and  $P_5$ , we will consider the word  $x^3yxyx$ , and  $xyxyx^3$ .

$$\begin{aligned} x^3yxyx &\xrightarrow{(1)} yxyx & x^3yxyx &\xrightarrow{(5)} x^2y^2 \\ xyxyx^3 &\xrightarrow{(1)} xyxy & xyxyx^3 &\xrightarrow{(5)} y^2x^2 \\ yxyx &\rightarrow x^2y^2 & & (6) \end{aligned}$$

$$y^2x^2 \rightarrow xyxy \quad (7)$$

Considering  $P_2$  and  $P_4$ , we will consider the word  $y^3xyxy$  and  $xyxyy^3$ .

$$\begin{aligned} y^3xyxy &\xrightarrow{(1)} xyxy & y^3xyxy &\xrightarrow{(4)} y^2x^2 \\ xyxyy^3 &\xrightarrow{(1)} yxyx & xyxyy^3 &\xrightarrow{(4)} x^2y^2 \\ yxyx &\rightarrow x^2y^2 & & (6) \end{aligned}$$

$$y^2x^2 \rightarrow xyxy \quad (7)$$

Once again we notice that the left hand side of (4) and (5) can be reduced using these two new relations, so those two relations are removed. Meaning our set of relations is now

$$x^3 \rightarrow 1 \quad (1)$$

$$y^3 \rightarrow 1 \quad (2)$$

$$yxyx \rightarrow x^2y^2 \quad (6)$$

$$y^2x^2 \rightarrow xyxy \quad (7)$$

Repeating the process for this new set of relations, we will be considering the words  $xyx^3$ ,  $y^3xyx$ ,  $y^2x^3$ , and  $y^3x^2$ .

$$\begin{aligned} xyx^3 &\xrightarrow{(1)} xyx & xyx^3 &\xrightarrow{(6)} x^2y^2x^2 \\ x^2y^2x^2 &\rightarrow xyx \end{aligned} \tag{8}$$

$$\begin{aligned} y^3xyx &\xrightarrow{(2)} xyx & y^3xyx &\xrightarrow{(6)} y^2x^2y^2 \\ y^2x^2y^2 &\rightarrow xyx \end{aligned} \tag{9}$$

$$\begin{aligned} y^2x^3 &\xrightarrow{(1)} y^2 & y^2x^3 &\xrightarrow{(7)} xyxyx \\ xyxyx &\rightarrow x^2 \end{aligned} \tag{10}$$

$$\begin{aligned} y^3x^2 &\xrightarrow{(2)} x^2 & y^3x^2 &\xrightarrow{(7)} yxyxy \\ yxyxy &\rightarrow x^2 \end{aligned} \tag{11}$$

Now we remove all relations whos left side is reducible, this would be (8), (9), (10), (11), (8) and (9) are reducible by (7), and (10), and (11) are reducible by (6). Thus we can conclude that the final set of relations are

$$x^3 \rightarrow 1 \tag{1}$$

$$y^3 \rightarrow 1 \tag{2}$$

$$xyxy \rightarrow x^2y^2 \tag{6}$$

$$y^2x^2 \rightarrow xyxy \tag{7}$$

## 8. SOLVING

Now with the Knuth-Bendix method, we are able to take any finitely generated group, and apply the Knuth-Bendix method to find a finite set of reductions. Then this set of reductions can be used to construct an automaton. Then as we have shown previously any group represented by an automaton has a solvable word problem.

**8.1. Reasoning.** The reason for the necessity of the Knuth-Bendix method, consider the group from section 7.1. Consider the element  $xyxyx$ . By the original relations, the automaton would have no way to reduce this element. Thus according to the automaton,  $xyxyx$  is a unique element. However once we apply our new relations, we can see that this element can actually be reduced to  $y^2$ .

Without the relations derived from the Knuth-Bendix method the automaton would not successfully recognize many elements of the group. Thus by using the derived relations we are able to reduce the representations to a simplified form, which acts as a representative for all elements which are equivalent. Thus making the comparison of elements trivial.

## REFERENCES

- [1] D. B. A. Epstein, D. F. Holt, and S. E. Rees. The use of knuth-bendix methods to solve the word problem in automatic groups. *Symbolic Computation*, 12:397–414, 1991.
- [2] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras'. *Proceeding of a Conference Held at Oxford Under the Auspices of the Science Research Council Atlas Computer Laboratory*, pages 263–297, 1970.