THE SOLVABILITY OF THE WORD PROBLEM FOR AUTOMATA GROUPS

ARDEN RASMUSSEN

1. Introduction

The word problem in group theory is the question of if two products of generators are equivalent to the same element of the group. For example consider the group D_4, then the question becomes; are $rrsrrsr^{-1}ss^{-2}rrs$, and $rsr^{-1}r^{-1}srrrs^{-1}r^{-1}r^{-1}s$ the same, and given any two elements in the group expressed as a product of the groups generators, can we determine if they are equivalent?

For an arbitrary group, this is not necessarily possible to determine this for all elements. However, for automata groups, it has been proven that it is always possible. When it is possible, we say that the word problem is solvable for the group. It has been proven that the word problem is solvable for automata groups, through the use of the Knuth-Bendix method.

2. Notation

Before we can begin into the structure of the proof of the solvability of the word problem, we first need to define some key concepts that will be used throughout the reset of the paper.

First, we are only focusing on finitely generated groups, which are the groups where every element can be written as a combination of a finite number of elements from a finite set of generators. This means that every group that we will be considering can be presented by generators and relations. For some group G, the set of generators will be denoted X, and the set of relations will be denoted R. First we will denote the set of generators union their inverse the *alphabet*, and denote it as Σ, so

$$\Sigma = X \cup X^{-1}.$$

Elements of the alphabet are typically called *letters*.

The second thing to define is the *language*. This is the free monoid of the alphabet, under the group operation.

> **Definition 1.** *The free monoid of a group is the set, whose elements are all finite sequences of elements from the group, including the sequence of zero elements, commonly denoted by ε.*

Elements of the language, will be called words. Note that we primarily use the notation of a group operation that is multiplicative, but the language can also be constructed using an additive operation, for example

$$\Sigma^* = \{\varepsilon, r, s, r+r, r+s, r+r+r, \ldots\}.$$

Date: April 25, 2020.

Example. Consider D_4, whose generators are $X = \{r, s\}$. Then the alphabet Σ would be

$$\Sigma = X \cup X^{-1} = \left\{r, s, r^{-1}, s^{-1}\right\}.$$

Finally the language would be the free monoid of the alphabet

$$\Sigma^* = \{\varepsilon, r, s, rr, rs, rrr, rrs, \ldots\}.$$

A word in this language could be $rrrsr^{-1}ss^{-1}$.

Since every element of a group can be defined as a product of the groups generators, then we will call some word $w \in \Sigma^*$, a representation of an element $a \in G$, if $w = a$.

The last notation that we must define is *shortlex ordering.* This is a method to apply ordering to the elements of a language.

Definition 2. *Words are sorted by length with the shortest words first, and words of the same length are sorted into lexicographical order (alphabetical).*

Generally this is just alphabetical ordering, similar to how a dictionary will order the words in it.

Example. For our example of D_4, the shortlex ordering is given by

$$\varepsilon < r < s < rr < rs < rrr < rrs < \cdots$$

3. The Word Problem

The word problem generally stated is to determine if two words are equivalent. More commonly discussed is the solvability of the word problem, as actually solving it usually results in large computations, it is beneficial to determine if the word problem is solvable before actually solving it. More commonly discussed is the solvability of the word problem, as actually solving it.

Definition 3. *The word problem for a finitely generated group G is solvable if for two representations of elements in that group $w, u \in \Sigma^*$, then it is possible to determine if $w = u$.*

At first this may seem 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, in specific groups it has been shown to be solvable.

Returning to the previous example of $rrsrrsr^{-1}ss^{-2}rrs$, and $rsr^{-1}r^{-1}srrrs^{-1}r^{-1}r^{-1}s$ in D_4. These elements appear very different, however upon evaluation, they are actually equivalent, and both of these represent the identity element in D_4.

4. Solvability of the Word Problem for Z_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.

$$Z_3 = \langle x | x^3 = 1 \rangle \tag{1}$$

In this group, our alphabet consists of $\Sigma = \left\{x, x^{-1}\right\}$, and so Σ^* is the set of all words 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.

(1) $xxx = e$
(2) $xx^{-1} = e$
(3) $x^{-1}x = e$
(4) $x^{-1}x^{-1}x^{-1} = e$
(5) $x^{-1} = xx$
(6) $x^{-1}x^{-1} = x$

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 are are left with xx and x respectively.

With these relations we can reduce any string to either be the empty string e, x, or xx. And thus the word problem is solvable for Z_3.

Example. Let us test this by considering $w = xxx^{-1}xx^{-1}x^{-1}xxxx$.

$$\begin{aligned} w &= xxx^{-1}xx^{-1}x^{-1}xxxx \\ (2) &\to xxxx^{-1}x^{-1}xxxx \\ (1) &\to x^{-1}x^{-1}xxxx \\ (3) &\to x^{-1}xxx \\ (3) &\to xx \end{aligned}$$

Thus we see that $w = xx = x^2$, and our relations successfully reduced the word to the element of the group which it represents. This process can be done for any word in Σ^*.

So we can consider two arbitrary elements $w, u \in \Sigma^*$. By the relations, and the process outlined above, we can reduce w, u to be one of e, x, xx, let us call the results of this reduction w', u' respectively. Then if $w' = u'$, then we can conclude

$$w = w' = u' = u.$$

5. Knuth-Bendix Method

6. Introduction

The word problem in group theory is the problem of deciding if two representations consisting 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 the groups.

7. Finite State Automata

We will first begin with some definitions that are used in the description of automatic groups. Firstly Σ is the finite set of *letters*, this is commonly called the *alphabet*. The free monoid generated by Σ will be denoted as Σ^*, and the elements of Σ^* are commonly called words. The free monoid Σ^* can be thought of as the set of all possible combinations of letters in Σ. We will also include the identity element of Σ^* to be denoted as ϵ, this is simply the word with no characters in it. For example consider the alphabet

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

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 Σ^* is called a *language*, and a subset of $\Sigma^* \times \Sigma^*$ is called a *relation*.

In our case we can commonly view the alphabet Σ 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 Σ^* 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 Σ is the alphabet of symbols, S is a non-empty finite set of states, s_0 is the initial state such that $s_0 \in S$, δ is the state-transition function $\delta : S \times \Sigma \to S$, 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 $A : \Sigma^* \to S$.

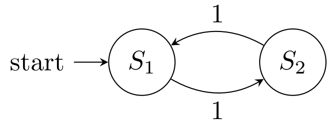

Figure 1. Very simple Finite state automata

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 δ 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.

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

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, ad 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 $S = \left\{e, a, a^2, a^3, b, ab, a^2b, a^3b\right\}$. Now we consider our $X = \{a, b\}$. Then we find $\Sigma = \left\{a, a^{-1}, b, b^{-1}\right\}$. And finally the transition table for D_4 is given in table 2;

Table 2. Transition table for D_4

δ_{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

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

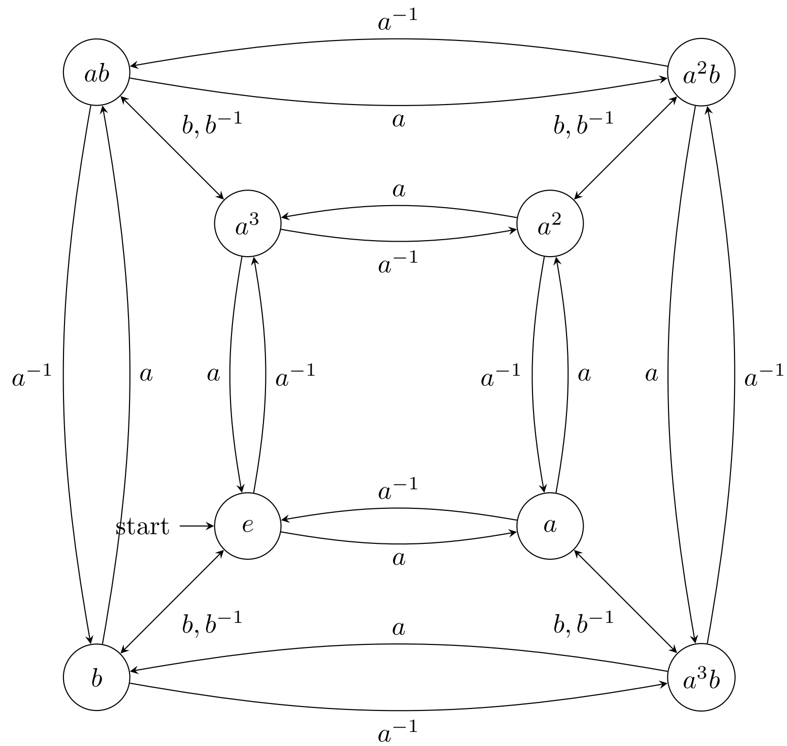

FIGURE 2. Finite state automaton for D_4

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 $w = a^3b$.

8. 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 Σ^* 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 given in Theorem 1.

Theorem 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 as a problem of rewriting elements in some group G. For some set of generators $x, y, z, \ldots$ for G, we introduce one letter for x and another for x^{-1}. We will then call these letters the alphabet Σ for the problem. With this definition every element in G is represented in some way by a product $abc \cdots pqr$ of symbols from Σ 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 weather 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.

9. 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.

$$Z_3 = \langle x | x^3 = 1 \rangle \tag{2}$$

In this group, our alphabet consists of $\Sigma = \left\{x, x^{-1}\right\}$, and so Σ^* 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.

(1) $xxx = e$
(2) $xx^{-1} = e$
(3) $x^{-1}x = e$
(4) $x^{-1}x^{-1}x^{-1} = e$
(5) $x^{-1} = xx$
(6) $x^{-1}x^{-1} = x$

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 are are left with xx and x respectively.

With these relations we can reduce any string to either be the empty string ϵ, x, or xx. And thus the word problem is solvable for Z_3. Let us test this by considering $w = xxx^{-1}xx^{-1}x^{-1}xxxx$.

$$\begin{aligned} w &= xxx^{-1}xx^{-1}x^{-1}xxxx \\ (2) &\to xxxx^{-1}x^{-1}xxxx \\ (1) &\to x^{-1}x^{-1}xxxx \\ (3) &\to x^{-1}xxx \\ (3) &\to xx \end{aligned}$$

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

10. 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. The next few sections demonstrate the process for constructing the relations that the automaton will utilize.

11. 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 ε is the empty string.

12. 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 set of relations which encapsulate all the information of the provided relations, but in a more simplified format. 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 $\left\{e, x, x^2\right\}$. 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 key compoenent that the Knuth-Bendix method resolves is the lack of confluence for a set of relations. The confluence of a set of relations, describes the fact that there are multiple ways to achieve the same result. Particullarly once can apply the relations in different orders and still achive the same result. The Knuth-Bendix method develops a set of relations such that confluence is always preserved. This is crutial to make the problem more computationally friendly, and thus applicable to automata.

Consider the finitely presented monoid $M = \langle \Sigma | R \rangle$, where Σ is the set of generators, and R is the set of relations. First we will consider the set of all posible words denoted Σ^*. Now we apply the concept of shortlex ordering to Σ^* to define an ordering on Σ^*, 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 $P_i \to Q_i$, for all i.

The next step is to progressivly 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 entierly 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

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

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.

12.1. **Example.** Let us consider a very simplistic example and use the Knuth-Bendix algorithm to rewritte 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

$$x^3 \to 1 \tag{1}$$
$$y^3 \to 1 \tag{2}$$
$$xyxyxy \to 1 \tag{3}$$

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

$$x^3yxyxy \xrightarrow{(1)} yxyxy \qquad x^3yxyxy \xrightarrow{(3)} x^2$$

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

$$yxyxy \to x^2 \tag{4}$$

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

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

Thus resulting in the relation

$$xyxyx \to y^2 \tag{5}$$

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

$$x^3 \to 1 \tag{1}$$
$$y^3 \to 1 \tag{2}$$
$$yxyxy \to x^2 \tag{4}$$
$$xyxyx \to y^2 \tag{5}$$

Now we repeate the entire process again.

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

$$x^3yxyx \xrightarrow{(1)} yxyx \qquad x^3yxyx \xrightarrow{(5)} x^2y^2$$
$$xyxyx^3 \xrightarrow{(1)} xyxy \qquad xyxyx^3 \xrightarrow{(5)} y^2x^2$$
$$yxyx \to x^2y^2 \tag{6}$$
$$y^2x^2 \to xyxy \tag{7}$$

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

$$y^3xyxy \xrightarrow{(1)} xyxy \qquad y^3xyxy \xrightarrow{(4)} y^2x^2$$
$$yxyxy^3 \xrightarrow{(1)} yxyx \qquad yxyxy^3 \xrightarrow{(4)} x^2y^2$$
$$yxyx \to x^2y^2 \tag{6}$$
$$y^2x^2 \to xyxy \tag{7}$$

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

$$x^3 \to 1 \tag{1}$$
$$y^3 \to 1 \tag{2}$$
$$yxyx \to x^2y^2 \tag{6}$$
$$y^2x^2 \to xyxy \tag{7}$$

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

$$yxyx^3 \xrightarrow{(1)} yxy \qquad yxyx^3 \xrightarrow{(6)} x^2y^2x^2$$

$$x^2y^2x^2 \to yxy \tag{8}$$

$$y^3xyx \xrightarrow{(2)} xyx \qquad y^3xyx \xrightarrow{(6)} y^2x^2y^2$$

$$y^2x^2y^2 \to xyx \tag{9}$$

$$y^2x^3 \xrightarrow{(1)} y^2 \qquad y^2x^3 \xrightarrow{(7)} xyxyx$$

$$yxyxyx \to x^2 \tag{10}$$

$$y^3x^2 \xrightarrow{(2)} x^2 \qquad y^3x^2 \xrightarrow{(7)} yxyxy$$

$$yxyxy \to x^2 \tag{11}$$

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

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

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

$$yxyx \to x^2y^2 \tag{6}$$

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

13. 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.

13.1. **Reasoning.** The reason for the necessity of the Knuth-Bendix method, consider the group from section 12.1. 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 Councile Atlas Computer Labratory*, pages 263–297, 1970.