THE WORD PROBLEM FOR AUTOMATA GROUPS

ARDEN RASMUSSEN

1. 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*. Commonly ␣ $\in \Sigma^*$ is defined as always mapping to the identity. Then the free monoid generated by Σ will be denoted Σ^*, and the elements of Σ^* are commonly called *words* or *strings*, and it possesses an identity as the *empty word* which will be denoted as ϵ. The free monoid Σ^* can be though of as the set of all possible combinations of letters in Σ. For example consider the alphabet

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

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.

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.

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 $A : \Sigma^* \to S$.

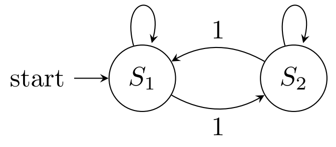

Figure 1. Very simple Finite state automata

Date: February 20, 2020.

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

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

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_1	S_2
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 $w = 111␣␣1$. Running the automaton with this input of w we see that each stage is described below

(1) $s = S_1$, $w = 111␣␣1$.
(2) $s = S_2$, $w = 11␣␣1$.
(3) $s = S_1$, $w = 1␣␣1$.
(4) $s = S_2$, $w = ␣␣1$.
(5) $s = S_2$, $w = ␣1$.
(6) $s = S_2$, $w = 1$.
(7) $s = S_1$, $w = \epsilon$.

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 $S = \{e, a, a^2, a^3, b, ab, a^2b, a^3b\}$. 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;

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.

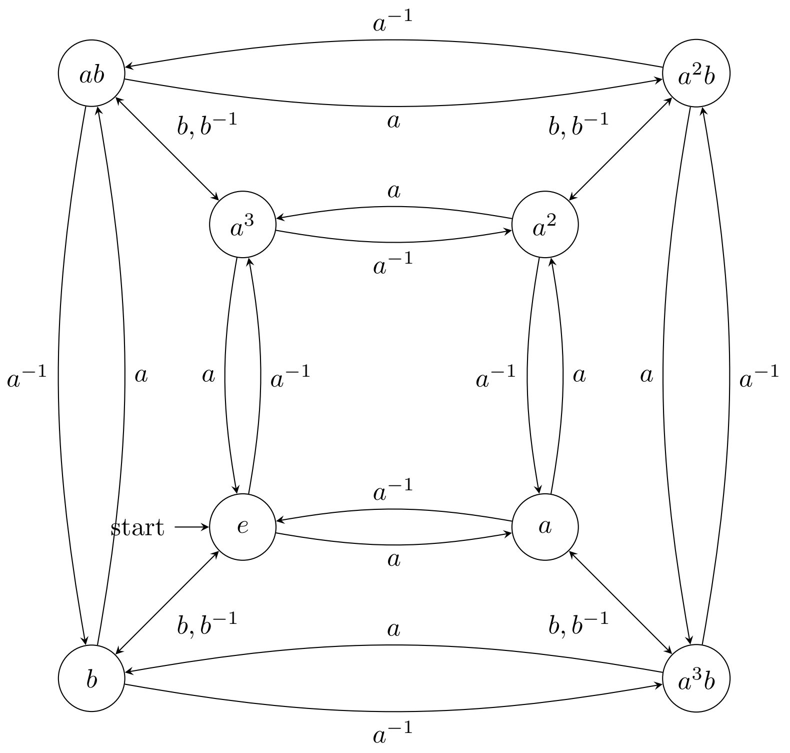

Figure 2. Finite state automaton for D_4

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 $w = a^3b$.

2. 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

Theorem 2.1. *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 $x, y, z, \ldots$ 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.

3. 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.

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

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 by the empty string ϵ, x, or xx. And thus the word problem is solvable for Z_3. Let us test this by considering $w = xxx^{-1}xx^{-1}x^{-1}xxxx$.

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

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

4. Automatic Group Structure

In group theory the Automatic groups are groups with additional structure applied to them in the form of finite state automata. This additional structure is required for the Knuth-Bendix method to be guaranteed to succeed, which will be discussed further in section 5.

Right now I need to read into more information about the structure associated with automatic groups. And why it is necessary for the Knuth-Bendix method to work. From the Internet:

> Let G be a group and A a finite set of generators. Then an *automatic structure* of G with respect to A is a set of finite state automata:
> - the *word-acceptor*, which accepts for every element of G at least one word in A^* representing it.
> - *multipliers*, one for each $a \in A \cup \{1\}$, which accept a pair (w_1, w_2) for words in w_i accepted by the word-acceptor, precisely when $w_1 a = w_2$ in G

From a paper:

> The group G is said to be automatic with respect to the generating set A if there exist finite state automata W, $M_\$$ and M_a for each $a \in A$, with the following properties.
> (1) W has input alphabet A and, for each $g \in G$, there is at least one element $w \in A^*$ with $w \in L(W)$ (the language accepted by W) and $\bar{w} = g$.
> (2) For $a = \$$ and $a \in A$, M_a has input alphabet $A^\dagger \times A^\dagger$, M_a accepts only padded words over $A \times A$ and, for all $v, w \in A^*$, $(v, w)^\dagger \in L(M_a)$ if and only if $v, w \in L(W)$ and $\bar{v}\bar{a} = \bar{w}$.

5. 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.

This seems more understandable, but has some really funky notation.

- Presentation
- Relations
- Reduction Ordering
- Shortlex Order
- Deductive Closure
- Rewrite Closure
- Equivalence Closure
- Converse
- Relation Composition
- Reflexive Transitive Closure

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.