My research with Professor Taback lies at the intersection of computer science and mathematics. We use automata theory, which is central to computer science, to analyze the structure of groups, a fundamental object of study in mathematics. Roughly speaking, an automaton is a machine that reads an input and outputs either “accept” or “reject” based on certain criteria and a group is a collection of elements together with an operation that takes two of the elements to form a third. A common example of a group is the set of integers under the operation of addition. In what follows, I first briefly discuss automata theory, then introduce the particular group we are focusing on, and finally show how we are using automata theory to study it.

Automata are basically simple computers. They consist of states and transition rules. An automaton takes a string of symbols as input and as it reads the string, it moves between states. When the last symbol is read, if the automaton is in an accept state, it accepts the string; otherwise, it rejects it. Together, all the strings accepted by an automaton form a language. Computers use automata to model language. The simplest automata only have states and read the input once. More complex automata have other features: e.g., some have counters (called stacks) and others can read both left and right along the input string. As an example, one can make a simple automaton that reads strings of the letters $a$ and $b$ and accepts strings with the same number of $a$’s and $b$’s.

The specific group we are studying is called Thompson’s group $F$. Thompson’s group $F$ was introduced by Richard Thompson in the 1960s in connection with questions in logic. It has since found applications in many areas of mathematics including algebra, logic, and topology. Elements of Thompson’s group $F$ are pairs of binary trees with the same number of edges and satisfying certain reduction conditions. An important element is shown in Figure 1.

A caret is defined as a node together with the two edges emanating downward from it (an upside-down vee), as seen in Figure 2. In the example above, each tree has 2 carets. We expanded on an existing system for classifying these carets based on their location in the tree. A caret is of type $I_r^0$ if it is the left child of its parent caret and has a right child; a caret is of type $I_r^0$ if it is the left child of its parent caret and has no right child; a caret is of type $I_r$ if it is the not the left child of a parent and does have a right child; and a caret is of type $I_0$ if it is not the left child of a parent and it has no right child. We then number the carets in each tree from left to right and let the string of caret types define the tree. For example, in Figure 1 the tree on the left is defined by the string $I_r^0, I_r^0$ and the tree on the right by the string $I_r, I_0$.

Note that some strings of caret types do not really correspond to trees. For instance, the string $I_r, I_r$ cannot correspond to a tree because every tree must have at least one caret with no children and both carets in the string have right children. Our first task was to construct an automaton that accepts all and only strings of caret types that correspond to a binary tree. This automaton is quite simple and is shown in Figure 4. The automaton has a stack which counts the number of $I_r^0$ and $I_0$ caret types.

Our next task is to construct an automaton that takes as input two strings and recognizes if they represent legitimate pairs of trees that satisfy the reduction condition, that is, if they represent elements of Thompson’s group $F$. Finally, we must construct an automaton that takes as input four strings and recognizes if the first two and second two define legitimate elements of Thompson’s group $F$ and recognizes the group operation between the two elements.
Figures:

**Figure 1.** The element $x_0$ in Thompson's group $F$.

**Figure 2.** A caret.

**Figure 3.** Caret $B$ is the right child of caret $A$.

**Figure 4.** The automaton accepting a string of caret types corresponding to a single binary tree.

**Faculty Mentor:** Jennifer Taback

**Funded by the** National Science Foundation