Summer 2013 NSERC USRA Report
Orderability of Knot Groups

Geoff Naylor

August 29, 2013

During the summer I have been investigating the orderability of knot groups under the supervision of Professor Dale Rolfsen. A group is orderable when we can assign an order relation on it that is invariant under applications of the group multiplication. For example if $x < y$ it would then follow for every z in the group that $zx < zy$ and $xz < yz$.

The key theorem that we used to investigate this problem was the result that every group was orderable if and only if every ball of finite radius was preorderable. For every group that is finitely generated we can associate a ball of any given radius to be the subset of all words whose length is less than or equal to the given radius. Thus these are finite sets and thus are ready for computational methods to be applied them.

A preordering for a ball of radius k B_k is a subset $P \subset B_k$ such that we have:

$$B_k = \{1\} \bigsqcup P \bigsqcup P^{-1} \tag{1}$$

$$P \cdot P \bigcap B_k \subset P \tag{2}$$

$$x \in B_k \Rightarrow xPx^{-1} \bigcap B_k \subset P \tag{3}$$

So we just need a method that determines for a given B_k if it is the case that a P satisfying all three properties exists. To do so we attempt to generate the set P and we either successfully construct it or demonstrate that it cannot exist in the attempt. The initial step is to pick an element of B_k that is not the identity and assume without loss of generality that it is in P. We can then use properties 2 and 3 to add new elements to P that follow by means of multiplication and conjugation. Eventually we either get all three properties at the same time and have a preorder, or we end up having $1 \in P$ in which case we cannot, or neither. In the case of neither we have to choose some element that neither it nor its inverse are in P. It is at this point we break it into cases where we first try augmenting P with the original element and then try to augment P with the inverse. If one works we get a preorder but if both fail we can demonstrate that a preorder is impossible. This branching process can occur many times and is one of the many factors that cause this to be a computationally complex problem.

The generation of P in this procedure and the verification for properties 2 and 3 require a method for multiplying words in the group. It is not enough to merely concatenate two words. We need a way of compressing them to their shortest length so we can determine their membership in the ball or determine if they are in fact the empty string corresponding to the identity. We need some way of creating an automatic procedure for reducing words to their shortest length form. To do this we use the Knuth-Bendix procedure. Essentially what it does is it starts with a small set of rewrite rules that identify the group. These are as simple as a rewrite rule corresponding to the relation of the group as well as the rewrite rules of any generator next to its inverse being equivalent to the empty string. The application of any of these rules always shorten a word, but they may not be sufficient for bringing the word to a canonical form. That's where rule confluence comes in. We can add new rewrite rules by looking at which of our current rules overlap. This allows us to make more rules from our initial set. These new rules may be confluent with each other as well which allows us to generate even more rules. This procedure does not in general terminate for most groups, but because we are only concerned with balls or some finite radius we are only concerned with reducing a finite collection of words. Only a finite subset of the complete set of rules would be necessary for reducing these words so after applying this procedure long enough we can be confident that our word reducer will be sufficiently strong for our purposes.

The execution of this algorithm was done in Python. The first part of my research project involved self-teaching myself the language. It was a great experience getting some hands on computer programming in mathematics. Upon computing a presentation for a knot group we can then begin to generate some ball of finite radius. Balls of radius up to 5 can be computed fairly easily. Radius 6 balls can take quite a bit longer but are still feasible to deal with in less than a day. Depending on the number of generators and how complicated the word reducer is a ball of radius 7 can take about a week of computer time. Radius 8 and above balls are really not feasible unless the group has a particularly simple word reducer.

I generated preorders for many knot groups for as high of a radius that was computationally viable at the time. While this isn't definitive proof on the orderability of any of these it is evidence to suggest that they are indeed orderable. Though for many groups whose relations are long this evidence is fairly weak. After all groups whose relations relate very long words to other long words will behave essentially like free groups on balls of small radius and free groups are orderable. In a sense, I find it much more desirable to get the result that something is not orderable because that result is definitive. Most of the groups shown to be not orderable through this procedure were ones already shown to be not orderable through other methods. Though one knot in particular did not fall into this category and was a non-trivial result. I did prove that the 5_2 knot is not orderable. Also, by logging all the multiplications and conjugations done by the algorithm and then only looking at the ones that lead directly to the identity, I was able to extract a short human readable proof that the 5_2 knot is not orderable. This proof would have been nearly impossible to do without the algorithm, as the chain of multiplications and conjugations that lead to the identity would have been nearly impossible to stumble upon manually. Thus I have had at least some success getting a positive result with this procedure. Also, outside the realm of knot groups I have studied the Baumslag-Solitar groups. I have discovered using this method that quite a few non-trivial cases of non-orderability exist within this family.

Since I have already built a program that tests for orderability, I plan to continue testing the orderability of groups for the foreseeable future. This has been an interesting problem to work on and it has been very rewarding to get some results. I am incredibly grateful to Professor Rolfsen and UBC for the opportunity to participate in this program.