Algebra and Computation Course Instructor: V. Arvind

Lecture 5: More on Subgroups and GROUP-INTER

Lecturer: V. Arvind *Scribe: Ramprasad Saptharishi*

1 Overview

Last lecture we saw that membership in a permutation subgroup can be done efficiently and inspected the recognizability of some group properties. This lecture we shall look at other properties of groups that can be detected efficiently.

2 The General Setting

Over all examples that we have seen, there is a central structure to all recursive algorithms that we are doing; we want small generating sets for recursive calls. In general, we are provided a group $G = \langle A \rangle$ and a subgroup H that is only given as a black box, and we wish to find a generating set for H to inspect its properties. The general idea was to find a tower of subgroups between G and H, like

$$G = G_0 \geq G_1 \geq G_2 \geq \cdots \geq G_r = H$$

and we want certain "nice" properties to be satisfied. This can infact be generalized to non-permutation groups as well. Assume that our groups are embedded into a larger group, where multiplication, inverses etc are known.

1. Given a generating set for G_i, we should be able to obtain a generating set for G_{i+1} quickly
2. The index $[G_i : G_{i+1}] \leq m$ is not too large
3. Quick membership algorithms in G_i
4. A REDUCE procedure to bring down the generating set.

The *quick membership algorithm* requirement has a definition by its own.

Definition 1. *A group H is said to be polynomial time recognizable if there exists a polynomial time algorithm that solves membership in H.*

Though H is not explicitly given, this is a very reasonable definition. For example, look at H being the automorphism group of some graph. Finding what H is very unlikely, but certainly given a permutation of vertices, one can easily check if it is infact an automorphism or not.

The properties stated before are precisely what we required for the recursion to go through.

Claim 2. *With the four properties, we can find a generating set for H quickly*

Proof. The first step is to find coset representatives of H in G.

$$G = Hx_1 \cup Hx_2 \cup \cdots \cup Hx_r \qquad r \leq m$$

In the permutation group case, just the orbit of $\{1\}$ would have finished it, but what about this case? Note that G acts transitively on the set of right cosets! Hence one can find a set of unique coset representatives by makign G act on the $id \in G$ and checking of two elements x and y of the orbit belong to the same coset of H (this happens if and only if $xy^{-1} \in H$).

Once we have the coset representatives, Schreier's lemma gives a generating set for the next element of the tower and recursion happens. Crowding of generators around H can be prevented by the REDUCE algorithm. Putting them all together, we would have a small generating set for H. ☐

3 Subnormality and GROUP-INTER

Definition 3. *A subgroup H of G is said to be subnormal in G if there exists a tower of subgroups chained by normality (i.e)*

$$H = G_r \lhd G_{r-1} \lhd \cdots \lhd G_0 = G$$

This is denoted by $H \lhd\lhd G$.

SUBNORMAL: Given $H = \langle B \rangle$ and $G = \langle A \rangle$. Check if $H \lhd\lhd G$.

The following claim is the core of the algorithm to detect subnormality.

Claim 4. $H \lhd\lhd G \iff H \lhd\lhd \left\langle H^G \right\rangle$

Proof. One way is obvious, if $H \lhd\lhd \left\langle H^G \right\rangle$, then immediately we have the tower $H \lhd\lhd \left\langle H^G \right\rangle \lhd G$.

The other direction is also fairly straightforward. Suppose we had a tower for $H \lhd\lhd G$,

$$H = G_r \lhd G_{r-1} \lhd \cdots \lhd G_0 = G$$

then clearly, just intersecting the entire tower with the normal closure $\left\langle H^G \right\rangle$, we get

$$H = H \cap \left\langle H^G \right\rangle \lhd G_{r-1} \cap \left\langle H^G \right\rangle \lhd \cdots \lhd G \cap \left\langle H^G \right\rangle = \left\langle H^G \right\rangle$$

which is a tower for $H \lhd\lhd \left\langle H^G \right\rangle$. □

Hence if H were subnormal, you are guaranteed to find a series through the normal closure. Hence the algorithm is then immediate:

Compute the normal closure $\left\langle H^G \right\rangle$. If $\left\langle H^G \right\rangle = H$, then we are done since $H \lhd G$ is our tower. If $\left\langle H^G \right\rangle = G$, then we know that H cannot be subnormal in G since there is no way by which you can obtain the tower if $\left\langle H^G \right\rangle = G$. Hence if $\left\langle H^G \right\rangle$ was a strict subgroup of G, recurse on this.

```
1: $K_1 = G$;  $K_2 = H$
2: while $\left\langle K_2^{K_1} \right\rangle \neq K_2$ do
3:     if $\left\langle K_2^{K_1} \right\rangle = K_1$ then
4:         output false
5:     else
6:         $K_1 = \left\langle K_2^{K_1} \right\rangle$
7:     end if
8: end while
9: output true
```

We will now go on to show that if $H \lhd\lhd G$, then we can compute $G \cap H$ efficiently.

Theorem 5. *Let $H = \langle B \rangle$ and $G = \langle A \rangle$ be subgroups of* Sym_n. *Given the promise that $H \lhd\lhd \langle G, H \rangle$, we can compute a small generating set for $G \cap H$ in polynomial time.*

Proof. Using the subnormality algorithm, we can infact compute the normal series of H. Let

$$\begin{aligned} H = G_r \lhd G_{r-1} \lhd &\ \cdots \ \lhd G_0 = \langle G, H \rangle \\ G \cap H \lhd G \cap G_{r-1} \lhd &\ \cdots \ \lhd G \cap G_0 = G \cap \langle G, H \rangle = G \end{aligned}$$

Now there are two questions to ask before we apply the recursion procedure, is the index between adjacent indices small? Given a generating set for $G \cap G_i$, can we find a generating set for $G \cap G_{i+1}$?

As for the second question, note that $G \cap G_i$ normalizes G_{i+1}! And hence we can use the previous algorithm to get a generating set for the intersection $(G \cap G_i) \cap G_{i+1} = G \cap G_{i+1}$

As for the first question, exercise! $\square$

3.1 Recognizing Solvability

Definition 6. *For any group G, the commutator subgroup G' of the group, also denoted by $[G, G]$, is defined as*

$$[G, G] = \left\{ g_1 g_2 g_1^{-1} g_2^{-1} \ : \ g_1, g_2 \in G \right\}$$

Also, it's easy to see that $G' \lhd G$ and G/G' is abelian, infact it's the smallest group that satisfies this property!

Definition 7. *A group G is said to be solvable if we can find a tower*

$$G \rhd G_1 \rhd G_2 \rhd \cdots \rhd \{1\}$$

such that for each i, G_i/G_{i+1} is abelian.

Equivalently, a group G is solvable if the you can hit $\{1\}$ by looking at successive commutator subgroups.

SOLVABLE: Given a group $G = \langle A \rangle$, check if G is solvable.

Here's the solution: Take $[A, A] = \left\{ a_1 a_2 a_1^{-1} a_2^{-1} \ : \ a_1, a_2 \in A \right\}$ and consider its normal closure. The claim is that, this is is the commutator subgroup of G! Once we prove this, we can descend by computing successive normal closures, and if we get stuck, then G can't be solvable.

We leave the proof of the claim and details of this algorithm as an exercise.

4 Jerram's Filter

Recall the MEMBERSHIP algorithm, the REDUCE procedure was crucial in keeping the generating set in control through the recursion. Our algorithm made sure that the generating set does not grow beyond n^2. However, there are much stronger results, and reduce algorithms.

Theorem 8 (Neumann)**.** *For $n > 3$, every subgroup of* Sym_n *can be generated by atmost $\lfloor \frac{n}{2} \rfloor$ elements.*

Infact, the above theorem is tight. Consider $\Omega = \{a_1, a_2, \cdots, a_m, b_1, \cdots, b_m\}$ and let G be the group generated by the transpositions $\{(a_i, b_i) \ : \ 1 \le i \le m\}$. This is a generating set for G and has size $\frac{n}{2}$. And since G is isomorphic to $\mathbb{F}_2^m$, this matches the lower bound as well.

However, to algorithmically compute the generators, we need slightly weaker bound.

Theorem 9 (Jerram)**.** *Any subgroup G of* Sym_n *has a generating set of size $(n-1)$. Infact, given $G = \langle A \rangle$, we can compute an $(n-1)$ sized generating set in polynomial time.*

Proof. Define the graph $X_A = ([n], E_A)$ as follows: For each $g \in A$, let $i_g \in [n]$ be the least point moved by g. Add the edge $e_g = (i_g, i_g^g)$ to E_A.

We now have an undirected graph on n vertices. We shall go on to show that we can keep modifying A to get to get to a graph $X_{A'}$ that is acyclic (and of course, $G = \langle A' \rangle$), and hence the theorem follows.

Before we go into the algorithm, we need one more terminology. For any $T \in \mathrm{Sym}_n$, the weight of the graph X_T (denoted by $w(T)$) is defined as $\sum_{g \in T} i_g$. An obvious upper bound on $w(T)$ is $|T|\, n$.

The algorithm is an online algorithm, it maintains a set A such that X_A is acyclic and as a new element g comes in, it changes A a little to accomodate g and still maintain an acyclic graph.

At any stage assume that we have a set S such that X_S is acyclic; enter g. If even on addition of e_g, the graph remains acyclic, there is nothing to be done, just add that edge and add g to the set S.

Otherwise, in $X_{S \cup \{g\}}$, there exists a unique cycle containing the edge e_g. Now each edge of the cycle is labelled with an element of S (the direction gives an ambiguity of whether it is g_k or g_k^{-1}, this will be usually denoted by g_k^ϵ where ϵ can be 1 or -1).

Let i be the least point on this cycle. Since i is a part of the cycle, one of the two edges incident on i in the cycle must be labelled as g_0 such that $i = i_{g_0}$. In that direction, walk from i round the complete cycle back to i; this naturally corresponds to a product of the form $g_0 g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_k^{\epsilon_k} = h$. Replace g_0 by h in the set S (unless h is trivial, in which case just discard it).

Note that this transformation doesn't change the group generated by the set, but on the other hand, the choice of h demands that $j^h = h$ for all

$1 \leq j \leq i$ and hence fixes sometime beyond i. And since we chose i to be the least element of the cycle, the weight of the graph increases when we replace g_0 by h.

The size of our set remained the same, but the choice of h forced the weight to go up. Hence in a polynomially many steps, we are sure to hit the upper bound and hence will end up with an acyclic graph (with some h becoming trivial in the process).

Repeating the process with all elements of A, we have a generating set A' with an acyclic graph $X_{A'}$, and hence $|A| \leq n - 1$. $\square$

Let us revisit the original REDUCE algorithm that restricted our generating set to n^2, infact that can also be seen in this setting. In that algorithm, we were looking at collisions and doing modifications based on that, those precisely correspond to 2 cycles in X_A.

Hence Jerram's filter can also be thought of as the original REDUCE algorithm but being more mindful about the entire graph rather than just 2 cycles.