Automating Sylow's Theorems

March 8, 2024

Nearly every introductory group theory course has a problem set with a series of problems of the form "prove that there are no simple groups of order n," for various values of n. Nearly all of these problems fall to a relatively small set of techniques, generally based on applications of Sylow's theorems. Once the list of techniques is established, these problems generally do not require new insights to solve. Generally, one starts with a list of the possible numbers of Sylow subgroups and then gradually rule out various numbers until hitting a contradiction. This process feels similar to working through logic grid and sudoku puzzles. Since computers are excellent sudoku solvers, it's reasonable to hope that they also make excellent Sylow solvers. We make a first step in this direction by implementing several of the common Sylow tricks, and developing a simple theorem prover which can apply these tricks. We hope that this first proof of concept will inspire other more sophisticated attempts towards Sylow automation.

1 Sylow Theory Techniques

We illustrate a list of techniques, which are useful in proving that there are no simple groups of a given order. In each section we will present a technique, and then illustrate a typical use of the technique within the context of an example.

1.1 Directly Applying Sylow

We start by recalling the following variant of Sylow's theorem.

Theorem 1. *Let* G *be a finite group. For a prime* p *dividing the order of the* G*, let* n_p *denote the number of Sylow* p*-subgroups of* G*. Then*

1. $n_p \equiv 1 \pmod{p}$
2. n_p *divides* $|G|$
3. *If* $n_p = 1$ *then the unique Sylow-*p *subgroup is normal in* G

Since this result is critical in all of our applications, we will briefly recall the proofs of these statements. We will take as given the fact that Sylow p-subgroups exist for all prime p (this follows from fact 1 above, but the reasoning is circular). See [ISAACS] or [DUMMIT] for the two most common proofs.

Proof. Since G's conjugation action on the set of Sylow p-subgroups is transitive, the second fact follows directly from the Orbit-Stabilizer Theorem.

If $n_p = 1$ then the Sylow p-subgroup is characteristic, since it may be described as the largest p-subgroup of G. Hence the unique Sylow p-subgroup is normal.

To prove the first fact, we fix a Sylow p-subgroup P, and consider the conjugation action of P on the remaining Sylow p-subgroups. The key observation is that this action has no fixed points. Suppose that P fixes a Sylow p-subgroup $Q \neq P$. Then the subgroup $\langle P, Q \rangle$ generated by P and Q contains both P and Q as Sylow p-subgroups. If P fixes Q under conjugation, then $\langle P, Q \rangle$ must also fixe Q under conjugation. But then Q is normal in $\langle P, Q \rangle$, which contradicts the third fact. Now by the Orbit-Stabilizer Theorem, P's conjugation action on the remaining $n_p - 1$ Sylow p-subgroups breaks up into orbits, each of size a power of a p greater than 1. Hence $n_p \equiv 1 \pmod{p}$. □

We now illustrate the use of this technique on a simple example.

Theorem 2. *There are no simple groups of order* 40.

Proof. If G is a group of order 40, then by Theorem 1 then n_5 must divide 40, and be congruent to 1 mod 5. This immediately forces $n_5 = 1$, from which we conclude that the unique Sylow 5-subgroup of G is normal. Hence G is not simple. □

For many group orders, applying Sylow's theorems is just this straightforward.

1.2 Embedding into S_n

Let G be a simple group which acts nontrivially on a set T of size n. If we identify T with the alphabet $1, 2, \ldots n$ then this action naturally gives rise to a map $\phi : G \to S_n$. Since G is simple, ϕ must be either injective or trivial, but the latter is ruled out since G's action on T was assumed to be nontrivial. So G must embed into S_n. In fact simplicity of G forces G to embed in A_n. To see this, consider the sequence of maps $G \to S_n \to \mathbb{Z}/2$, where the second arrow corresponds to the quotient of S_n by A_n. Unless G

We may apply the above line of reasoning either to G's action on its Sylow p-subgroups, or to G's action on the cosets of a particular subgroup. We conclude the following two rules.

Theorem 3. *Suppose that* G *is a simple group.*

- *If* n_p *is the number of Sylow* p*-subgroups of* G*, then* G *is isomorphic to a subgroup of* A_{n_p}.

- *If* G *has a subgroup of index* n*, then* G *is isomorphic to a subgroup of* A_n

Theorem 4. *There are no simple groups of order* 24*.*

Proof. Suppose that G is a simple group of order 24. By Sylow's theorems, the number of Sylow 3-subgroups of G is either 1 or 4. If there is only one Sylow 3-subgroup, then it is normal in G, and hence G is not simple. Otherwise G has exactly 4 Sylow 3-subgroups, and so G is isomorphic to subgroup of A_4. Therefore $|G|$ divides $|A_4| = 4!/2 = 12$ which is an immediate contradiction. □

We may enhance the utility of this trick slightly by recalling that A_n is simple as long as $n \geq 5$. We illustrate this with another example.

Theorem 5. *There are no simple groups of order* 90*.*

Proof. Suppose that G is a simple group of order 90. By Sylow's theorems, the number of Sylow 5-subgroups of G must be 6. Hence G embeds into A_6. Since the order of A_6 is $6!/2 = 360$, this implies that G has index 4 in A_6. Applying the same trick again, we see that A_6 embeds into A_4, which is a contradiction since $\#A_6 \nmid \#A_4$. □

1.3 Counting

In some cases it is possible to show that a simple group of order n must have more than n elements, yielding an immediate contradiction. To apply this trick, one first shows that any two Sylow p-subgroups have small intersection. If there are a large number of Sylow p-subgroups then, the elements of order p cover a large number of group elements, sometimes taking up enough space that the other Sylow subgroups cannot fit. This trick is most easily applied when the Sylow p-subgroups are cyclic of prime order; in that case each pair of Sylow p-subgroups is guaranteed to overlap in only a single element.

The smallest group order for which a counting argument is natural is 30. We illustrate the argument below.

Theorem 6. *There are no simple groups of order* 30*.*

Proof. Suppose that G is simple of order 30. By Sylow's theorems G must have exactly 6 Sylow 5-subgroups. Since the Sylow 5-subgroups are cyclic of prime order, each pair intersects in exactly one element. Therefore G has $(5-1)\cdot 5 = 21$ elements of order 5. However G must also have exactly 10 Sylow 3-subgroup, each of which is also cyclic of prime order. So G must have at least $(3-1)\cdot 10 = 20$ elements of order 3. Since $21 + 20$ is greater than the order of G, we conclude that G cannot be simple. □

1.4 Normalizers of Sylow Intersection

The tricks discussed so far are sufficient to deal with all the non-simple orders up to 100. Unfortunately, these tricks quickly begin to break down for much larger orders. Pushing forward requires a more sophisticated collection of tricks. Most notably, it is often useful to consider the normalizer of the largest intersection of Sylow p-subgroups. If we can force the normalizer to have small index in G, then we obtain a contradiction by considering an embedding into A_n.

First, recall the following fact which lets us find Sylow p-subgroups with large intersection.

Theorem 7. *Let G be a group whose Sylow p-subgroups have order p^n. Suppose that each pair of distinct Sylow p-subgroups intersects in no more than p^k elements. Then the number of Sylow p subgroups of G is congruent to 1 mod p^{n-k}.*

The proof of this fact is nearly identical to the proof of Theorem 1. The normalizer of intersection technique is best illustrated with an example.

Theorem 8. *There are no simple groups of order 144.*

Proof. Suppose G is simple of order 144. By Sylow's theorems, the number of Sylow 3-subgroups of G is either $1, 4$, or 16.

By the previous result, there must be two Sylow 3-subgroups P_1 and P_2 which intersect in a subgroup H of order 3. Since p-groups are nilpotent, the normalizer of $N_G(H)$ must strictly contain both P_1 and P_2. So $N_G(H)$ must have order $18, 36$, or 72. If $N_G(H)$ has order either 36 or 72, then G must embed into either A_2 or A_4, each of which is an immediate contradiction. Therefore $|N_G(H)| = 18$. But by Sylow's theorems, $N_G(H)$ must have a unique Sylow 3-subgroup. Therefore $N_G(H)$ contains a unique subgroup of order 9, which is a contradiction since $N_G(H)$ contains both P_1 and P_2. □

2 A simple abstraction for simple theorem proving

The goal of this work was to design an environment in which we could automatically prove that there are no simple groups of order n for various values of n.

One approach to this problem is a fully ad-hoc solution, where we specify a specific series of steps to be applied in order. Such a program might start by directly applying Sylow's theorems to obtain a list of the possible numbers of Sylow p-subgroups for each p dividing n. Several of these numbers can often be ruled out immediately if they force G to embed into an alternating group which is too small. Following that, the program can obtain a lower bound for the number of elements of p-power order for various values of p. If the total number of prime power order exceeds n, then we derive an immediate contradiction and conclude that there are no simple groups of order n. This approach works

well for many small values of n. In fact the first composite n for which this sequence of steps fails (other than $60 = |A_5|$) is 90. Unfortunately this approach fails to extend easily to other more sophisticated techniques. To add a new technique, a substantial amount of the code base must be changed in order to keep track of the various quantities needed to implement the new method.

At the other end of the spectrum, one could imagine a program that is fully-automatic. Such a program would take as input the definition of a group, along with some basic facts about finite groups and integers, and use those facts in order to prove that there are no simple groups of a given order. This would require encoding the rules of first-order logic, which is a challenge in itself. But even more, it would require the computer to prune its search space intelligently, in order to avoid combinatorial explosion when looking for a proof.

We opt for a hybrid of these extremes. Our approach is to provide the computer with a list of techniques which are known to be useful when proving that there are no simple groups of order n. In our framework, a technique (or theorem) is simply an object that takes in a collection of facts, performs some well-defined series of operations, and then outputs a collection of facts. This as an advantage over the fully ad-hoc approach of being extensible. Implementing a new method simply involves adding a new object to the list of techniques; none of the core theorem-proving code needs to be changed. Similarly, it is straightforward to remove a particular method, as this only involves deleting a single line of code.

On the other hand, we avoid the combinatorial explosion faced by the purely automatic approach by carefully selecting a collection of techniques which are fairly course-grained. For example, our implementation does not need to know the definition of a group. Instead it is programmed to work with higher level techniques, such as Sylow's theorems, and the counting method. Our techniques are sufficiently targeted that each applies only rarely. This allows us to take a brute-force approach to theorem-proving, while avoiding combinatorial explosion. At the same time our implementation is sufficiently general that it can handle additional techniques, as well as well as other types of theorems.

2.1 Facts and Techniques

Within our framework, a *simple fact* consists simply of a name (which is a string), followed by an (ordered) list of strings which are thought of as a set of arguments for a given type of fact. For example the fact, `(group, [G])` expresses the logical statement "G is a group". In a similar way, the fact `(order, [G,n])` expresses the statement "the order of G is n" (while n is formally a string within our framework, we generally convert it to an integer before working with it). We make note that the logical description associated to a fact is only for the sake of human understanding, and is not used by our program.

It is useful to be able to describe more complicated facts which are connected by a

logical or. To express such facts, we define a *disjunction* type which simply consists of a list of facts. The interpretation is that a disjunction is true if at least one of its component facts is true (though formally, our program does not manipulate truth values). For convenience we will generally write `OR(fact1, fact2)` as shorthand for the disjunction object containing `fact1` and `fact2`.

A *technique* or theorem is a map that takes as input certain lists of facts, outputs a new list of facts. Each theorem contains a template which describes the collections of facts which form a valid input to the theorem. Formally a template is simply a list of facts. We say that a list of facts $\mathcal{F}$ matches the template if $\mathcal{F}$ is equivalent to the template up to a relabelling of its arguments. For example if `[(foo, [A, B]), (foo[A, C])]` is our template, then `[(foo,[X,Y]), (foo,[X,Z])]` and [(foo,[X,X]),(foo,[X,X])] both match the template, while [(foo,[W,X]), (foo,[Y,Z])] and [(bar,[X,Y]), (bar,[X,Z])] do not. A list of facts is a valid input for a theorem if and only if it matches the theorem's template.

Given a valid list of input facts, a theorem may output a list of output facts in an arbitrary way, and the output facts may consist of either simple facts or disjunctions (or both). Of course we generally define our theorems in a way that aligns with standard mathematical results. For example, the theorem "sylow" takes a list of inputs matching the template [(group, [G]), (order, [G], n)] and outputs a list of disjunctions corresponding to the possible numbers of Sylow subgroups for each prime dividing n. When $n = 10$, such an output would read [OR((numSylow, [2, G, 1]), (numSylow, [2, G, 5])), OR((numSylow, [5, G, 1])). The fact (numSylow, [2,G,1]) is interpreted as "the number of Sylow 2-subgroups of G is 1." Note that the output list of facts contains the character G. This is intended to match the template. If the facts [(group, [H]), (order, [H], 10)] were input into the theorem instead, the result would be the same, however G would be replaced by H.

We complicate this model slightly for implementation purposes. A fact output by a theorem may contain an argument which is a string beginning with a question mark (e.g. ?T). This means that ?T is a wildcard character whose value will be decided later. The reason for this is that we would like theorems to be able to state the existence of new mathematical objects. Within our framework such an object must be given a name, but that name should not collide with a string that is already in use. A theorem object does not have a broad enough scope to see the other strings that are currently in use, so the wildcard character is our way of delegating the task of choosing names elsewhere. A typical use of the wildcard character is to express the fact "there exists a group of order n". A theorem that wants to return such a fact might return `[(group, [?G]), (order,[?G,n])]` The string `?G` will be replaced a new name later on.

We also allow the strings in our templates to start with the special character '*'. This indicates that we do not allow this string to be renamed in the usual way when deciding template matching. The typical use of this is for an input fact such as (order, H, *1), where a specific number is important. The *1 indicates that the string 1 may not be replaced with any other value when template matching.

2.2 Proving Theorems

Given our abstractions for facts and theorems, searching for a proof is actually quite easy. We build an environment, which we call a *proof environment* which initially consists of a list of facts, together with a list of theorems, and a goal. The initial list of facts are our hypotheses. For the problem of showing that there is no simple group of order n, our initial fact list `(group, [G])`, `(order, [G, n])`, `(simple, [G])`. In other words we start by hypothesizing that there is a simple group G, of order n. Our goal is to prove the fact `(false, [])` under these hypotheses. The fact `(false,[])` is no different from any other fact, but we generally interpret it to indicate that we have arrived at a contradiction. The theorems in our case correspond to the usual tricks for applying Sylow's theorems.

We first explain our method of theorem proving without disjunctions. In this case, our approach is as simple as possible. We maintain a collection of established facts, which is initially consists our hypotheses. Then we scan through the list of theorem templates, looking all possible ways to match an ordered list of established facts to one of the theorem templates. Then for each match, we apply the relevant theorem resulting in more established facts which we add to the collection. We iterate this procedure, growing the list of established facts until either we hit a threshold number of iterations, causing us to declare failure, or we add the goal to the established facts in which case we declare success. As a minor optimization, we only apply a theorem to a set of facts if at least one of those facts was concluded from the previous round. Otherwise the theorem must have been applied to the same set of facts in an earlier round, so there is no need to apply it again.

To allow for disjunctions we need to complicate this model slightly. Instead of adding a disjunction directly to the list of established facts, we add each of its component facts, and make a note that each added fact is related to its parent disjunction. So if applying a theorem results in the disjunction D = OR(X,Y), we add X and Y to our list of established facts, and set a value in each of X and Y, indicating that they are dependent on D. In general, each fact holds a list which contains all of the disjunctions that the fact depends on from anywhere in its ancestry. In our example we would have X.disjunctionDependencies = [(D,0)] indicating that X depends on the 0-index fact of the disjunction D. Similarly Y.disjunctionDependencies = [(D,1)]. In general, when a fact F_1 is used to conclude another fact F_2, the disjunction dependencies of F_1 are appended to the list of disjunction dependencies of F_2. In this way we maintain a complete list of the disjunctions that each fact depends on anywhere in its ancestry. Note that disjunctions are themselves treated as facts, and so each disjunction also maintains a list of the disjunctions that it depends on, which it passes to its children.

With this new setup we can no longer conclude that we have a proof of our goal if it shows up in this collection of established facts. The reason is that our fact may depend on one more facts which come from a disjunction. For example, suppose we begin with the hypothesis D = OR(A,B), and use A to prove our goal G. Then we have shown that A

implies G, but this is not enough to conclude that D implies G. On the other hand, if we show that both that $A \longrightarrow G$ and $B \longrightarrow G$, then we have

$$\begin{aligned}(A \longrightarrow G) \text{ and } (B \longrightarrow G) &\equiv (A \text{ or } B) \longrightarrow G \\ &\equiv D \longrightarrow G.\end{aligned}$$

Therefore in this case it is valid to conclude G. Within our proof environment, this situation occurs when the fact G shows up twice, once with disjunctionDependencies equal to [(D,0)] and once with disjunctionDependencies equal to [(D,1)]. In general, there is a rule which lets us decide whether a fact G has been proven within our environment.

Suppose that the fact G occurs k time with the i^{th} occurrence of G having disjunctionDependencies equal to d_i. For each i, let $d_i = \{(f_1, x_1), \ldots (f_{i,t}, x_{i,t})\}$. Let S be a set of integers indexing a subset of the d_i. Suppose that each element of the product $\prod_{i \in S} f_1$ is a subset of some d_i. Then G is true. In other words to test if a fact G is true, we may look at all of the disjunctions that all instances of G depend on. For each disjunction D, at least one of its facts is true. We consider each possible identity of the true statement, for every disjunction. If in each such case we can conclude G, then G must in fact be true.

Note that for this to work, it is important that disjunctions also maintain a list of their dependencies. Otherwise consider the following situation. Let $X = \text{OR}(A, B)$, and let $Y = \text{OR}(C, D)$, with both C and D implying the goal G. If Y depends on fact $B = (X, 1)$, then it is not correct to say that

$$(A \text{ and } C) \text{ or } (A \text{ and } D) \text{ or } (B \text{ and } C) \text{ or } (B \text{ and } D) \longrightarrow G,$$

because C and D both implicitly depend on B. The problem is fixed when we add B to the list of dependencies for Y, and hence both C and D.

Thus we have a method which allows us to decide when a fact G has been proven. While our method is crude, it works for many instances of problems which can be solved by applying Sylow's theorems.

3 Conclusions

We have demonstrated a flexible framework for automating the application of Sylow's theorems to prove that there are no simple groups of a given order. Our approach encodes various Sylow techniques as "theorem" objects that can be applied to a collection of established facts to derive new facts. By combining these theorems in different ways, we can handle a wide variety of group orders, including many challenging cases.

While our current implementation is fairly basic, combining just a handful of standard Sylow techniques, it demonstrates the potential for further automating arguments in finite group theory. By encoding more sophisticated techniques as theorem objects, one could likely handle many larger and more challenging group orders. Furthermore, the same

framework could be adapted to automate proofs in other areas of abstract algebra or even different fields of mathematics.

Overall, this project serves as a proof of concept, showing how relatively simple programs can leverage the power of modern computers to tackle problems that have traditionally relied on human intuition and insight. We hope this work will inspire further research into automated theorem proving, particularly in areas like finite group theory where many problems share a common set of techniques.

4 Appendix: The Interesting Group Orders

We give several lists summarizing the first few challenging group orders, for various definitions of challenging. These lists were generated by our code. Each list begins with the techniques that were used. The numbers list the non-prime power group orders up to 1000 for which the techniques are not sufficient. Fortunately the orders of the non-abelian simple groups do show up in these lists.

Only Sylow's Theorem: 12, 24, 30, 36, 48, 56, 60, 72, 80, 90, 96,105,108,112,120,132, 144,150,160,168,180, 192, 210, 216, 224, 240, 252, 264, 270, 280, 288, 300, 306, 315, 320, 324, 336, 351, 360, 380, 384, 392, 396, 400, 420, 432, 448, 450, 480, 495, 504, 520, 525, 528, 540, 546, 552, 560, 576, 600, 612, 616, 630, 640, 648, 660, 672, 702, 720, 728, 735, 750, 756, 760, 768, 784, 792, 800, 810, 840, 858, 864, 870, 896, 900, 918, 924, 945 960, 972, 990, 992

Sylow/Embedding into A_n: 30, 56, 60, 105, 132, 144, 168, 180, 210, 240, 252 264, 280, 288, 306, 315, 336, 351, 360, 380, 396, 400, 420, 432, 480, 495, 504, 520, 525, 528, 540, 546, 552, 560, 576, 612, 616, 630, 660, 672, 702, 720, 735, 756, 760, 792, 800, 810, 840, 858, 864, 900, 918, 924, 960, 990, 992

Sylow/Embedding into A_n/Counting: 60, 144, 168, 180, 210, 240, 252, 264, 280, 288, 306, 315, 336, 360, 396, 400, 420, 432,480, 495, 504, 525, 528, 540, 560, 576, 612, 630, 660, 672, 702, 720, 735, 756, 760, 792, 800, 810, 840, 864, 900, 918, 924, 960, 990

Our implementation of the normalizer of intersection trick knocks off the orders: 144, 306, 315, 400, 525, 735, 800, and 918 leaving the following as our list of hard orders less than 1000.

Sylow/Embedding into A_n/Counting/Normalizer of Intersection: 60, 168, 180, 210, 240, 252, 264, 280, 288, 336, 360, 396, 420, 432, 480, 495, 504, 528, 540, 560, 576, 612, 630, 660, 672, 702, 720, 756, 760, 792, 810, 840, 864, 900, 924, 960, 990

For reference, a simple group of non-prime order less than 1000 must have order either

60, 168, 360, 504, or 660.

5 Appendix: An Uninteresting Conjecture

Say that a group order n is *uninteresting* if it falls to a direct application of Sylow's theorems. In other words, the integer n is uninteresting if it has a prime factor p, for which there is only a single divisor of n which is congruent to 1 mod p. Let $U(x)$ be the number of uninteresting n which are most x.

Numerical data strongly suggests that $\lim_{x\to\infty} U(2x)/U(x)$ exists and is approximately equal to 1.68. This suggests that perhaps there is a positive constant $\alpha < 1$ such that $U(x) \in \Theta(x^{\alpha})$. Is this true? Even a heuristic justification would be nice.