CREATING A DYNAMIC DATABASE OF FINITE GROUPS

LEWIS COMBES, JOHN JONES, JENNIFER PAULHUS, DAVID ROE, MANAMI ROY, AND SAM SCHIAVONE

ABSTRACT. A database of abstract groups has been added to the *L-functions and Modular Forms Database* (LMFDB), available at https://www.lmfdb.org/Groups/Abstract/. We discuss the functionality of the database and what makes it distinct from other available databases of abstract groups. We describe solutions to mathematical problems we encountered while creating the database, as well as connections between the abstract groups database with other collections of objects in the LMFDB.

CONTENTS

Date: October 2, 2023.

2020 *Mathematics Subject Classification.* 20-04, 20-08, 20Dxx.

1. Introduction

Finite groups have played a profound role in mathematics for close to two centuries and, almost since their inception, mathematicians have asked classification questions about them. The quest to classify finite simple groups took up most of the 20th century. In the last 30 years as computational power became prominent, various databases of groups have been developed. The *ATLAS of Finite Groups* which contains information on interesting groups, notably character tables, began as a book [CCN+85] and is now available online [WWT+]. The computer algebra programs GAP [GAP21, BEO02] and Magma [BCP97] both have complete databases of small groups up to order 2000 (except for order 1024, for which there are almost 49.5 billion distinct isomorphism classes). These computer algebra programs also include other databases such as perfect groups (up to orders $2 \cdot 10^6$ and 50,000, respectively) and transitive groups up to degrees 48 [Hul], [BCH+]. The *GroupNames* project [Dok] includes groups up to order 500 (skipping orders 256 and 384) as well as additional information such as lattice of subgroups and character tables for the groups.

1.1. Features of the database.

As we note above, there are several other databases of groups already available, so why do we need yet another one? Some of the useful features of our database are the following.

- *The database is dynamically searchable.* We have precomputed and stored many invariants and properties of the groups in the collection, and created a search interface which allows users to search on these. For instance, if a user wants to find all groups of order 256 with nilpotency class 6, this can be accomplished in seconds using the search interface. (There are 38 such groups.) While in principle it may be possible to perform these sorts of searches in a given computer algebra system equipped with a database of groups, such a computation would likely require looping over many groups, computing the desired invariants or properties, and then filtering out those not meeting the search criteria, which will often be very time-consuming and computationally intensive.
- *The database is free and easy to use.* Use of our database requires no prior knowledge of any computer language or computer algebra package, making it accessible to a broad audience, from students just learning the basics of group theory, to experts using groups in their research. While most of our computations were done in Magma, which is closed-source and requires purchase of a license, the data we have computed is now publicly available on the internet through the LMFDB.

- *The database fosters connections between other collections of objects in the LMFDB.* The *L-functions and Modular Forms Database* (LMFDB) is a huge international collaboration to collect, curate, and connect computational mathematical work in number theory, particularly as it relates to the Langlands program. Groups are attached to many mathematical objects throughout the LMFDB, from Sato-Tate groups to automorphism groups of curves, and our database of groups facilitates more connections between these various collections. See Figure 3 in subsection 6.1 for a diagram illustrating these connections.
- *The database aggregates groups drawn from a variety of sources.* The database is flexible, in that it can accommodate groups of various types drawn from multiple sources, such as polycyclic, permutation, and matrix groups. See section 2 for a list of sources used to compute the data.
- *The database supports searches on subgroups.* This is in some sense a special case of the first point, but for each group in the database, we have computed and stored information on its subgroups. Users can search for all groups containing a subgroup with a given property; for instance, one can easily find the 12 groups of order 96 that contain a normal subgroup isomorphic to A_4. (See subsection 3.3 for more details on the subgroup information stored.)
- *The database provides a deterministic labeling of groups, their subgroups, and their conjugacy classes.* This labeling allows users to connect a particular conjugacy class or subgroup to the corresponding one in our database in a deterministic way, regardless of the given presentation of a group

1.2. **Structure of the paper.** We begin in Section 2 with information about sources for the groups currently included in the database, and how the data about them is generated. In Section 3 we describe the web interface and demonstrate a number of features of the database. In the process of designing the database and computing the corresponding data, we encountered a series of challenges which required mathematical solutions. The next two sections discuss these challenges: Section 4 describes the method by which we computed presentations for solvable groups and Section 5 details the procedures we used to deterministically label subgroups, conjugacy classes, and characters. In Section 6 we give examples of connections between the Abstract groups database to other collections of objects in the LMFDB.

1.3. **Acknowledgments.** We owe a large debt of gratitude to Tim Dokchitser, whose *GroupNames* website was the initial inspiration for our database and who generously shared code and database expertise with us (as well as giving us permission to re-post many of the definitions on his page as knowls on the LMFDB). We are also appreciative of the advice and many suggestions Andrew Sutherland gave us. We are grateful for various conversations and code from Michael Bush, Derek Holt, Alexander Hulpke, Bjorn Poonen, and David Roberts.

Thank you to the American Institute of Mathematics for hosting a mini-workshop where much of the initial planning happened, and the Institute for Computational and Experimental Research in Mathematics for a workshop where the project took off.

Roe and Schiavone were supported by the Simons Collaboration in Arithmetic Geometry, Number Theory, and Computation via Simons Foundation grant 550033. Paulhus was partially supported by a Frank and Roberta Furbush Scholarship from Grinnell College. Combes was supported by the Engineering and Physical Sciences Research Council grant EP/R513313/1.

2. Data overview

The database currently contains over 500,000 groups together with roughly 200 million subgroups and 50 million of their irreducible complex characters.

The code used to generate the data is written in Magma [BCP97] and Python [VRD09], and it may be found at the GitHub repository `https://github.com/roed314/FiniteGroups`. The database, as with the rest of the LMFDB, uses PostgreSQL as its database management system. Computations were carried out via a combination of an AMD Epyc 7713 server with 256 2.0GHz cores and 2TB of RAM, and a distributed computation on Google Compute Engine, with a substantial use of GNU parallel [Tan11] in both cases.

Source	Total	Solvable	Perm.	Matrix	OptimizedPC	MinPerm
Small Groups	257936	257500	257746	68042	257500	257746
Transitive Groups	235919	211279	235919	218	14499	161656
Intransitive Groups	5444	2739	5444	16	2330	5378
Classical Lie Type	2201	2	1509	2201	0	1509
CARAT	189	185	186	189	174	186
$\mathrm{GL}_n(\mathbb{F}_q)$ Subgroups	3018	2456	3000	3018	2397	3000
$\mathrm{GL}_2(\mathbb{Z}/N)$ Subgroups	29771	28819	25319	29771	24323	25254
Perfect	123	0	123	1	0	123
Chevalley	13	0	7	13	0	7
Sporadic	9	0	9	7	0	9
Small Group Auto.	283	283	283	0	73	111
Transitive Group Auto.	498	438	498	0	51	201
Auto. Groups of Curves	530	527	530	0	526	530

Table 1. Number of groups by source. See Section 2 for more information.

We compute and store data on groups from various sources. Counts of groups currently in the database from these different sources are given in Table 1. The "Total" column represents the total number of groups in the database from that source, excluding those already shown

in a previous line. The "Solvable", "Perm.", and "Matrix" columns give the number of groups from each source that are solvable, the number of groups for which we store a permutation representation, and the number of groups for which we store a matrix representation (over $\mathbb{Z}$, $\mathbb{F}_p$, $\mathbb{F}_q$, or $\mathbb{Z}/N$), respectively. The "OptimizedPC" column counts how many groups from that particular source include an optimized polycyclic presentation (see Section 4), meaning a presentation guaranteed to have a minimal number of generators, and "MinPerm" gives the number of the groups for which we know a minimal degree permutation representation.

Roughly half of the data is initiated from the small groups database in Magma [BE99a, BE99b, BEO01, O'B90, BE01, O'B91, NOVL04, OVL05, DE05, DE12, DEP22]. All groups of order up to 2000, except those whose order is larger than 500 and divisible by 128, are included. Most of the remaining groups come from the transitive groups database [Hul05, CH08, HR20], from which we include all groups of degree up to 47 except those of degree 32 with order between 512 and 40 billion. Note that our notion of equivalence is abstract isomorphism rather than conjugacy within S_n; the work to divide the transitive groups into isomorphism classes is described briefly in subsection 2.2. In addition to the small group and transitive group databases, we use the following sources:

- classical Lie groups up to particular bounds, and Chevalley groups that don't already show up as classical Lie groups [Tay87, RT98, HRT01];
- additional intransitive groups which are subgroups of S_{15} not currently in the database, computed using the Magma `Subgroup` command;
- all integer matrix groups of dimension up to 6 from CARAT [OPS08];
- subgroups of $\mathrm{GL}_n(\mathbb{F}_q)$ computed via Magma for $n = 2$ ($q < 1000$), $n = 3$ ($q < 16$), $n = 4$ ($q < 7$) and $n = 5$ ($q = 2$);
- subgroups of $\mathrm{GL}_2(\mathbb{Z}/N\mathbb{Z})$ for N up to 125 (skipping 80, 96, 104, 112 and 120);
- all perfect groups of order up to 50,000 from the corresponding database in Magma;
- the sporadic groups J_1, J_2, HS, J_3, McL, He, Ru, Co_3, and Co_2 with permutation and matrix representations taken from the Atlas of Sporadic Groups [WWT+];
- automorphism groups of curves up to genus 48 [Bre00], and "large" automorphism groups (of order $> 4(g-1)$) of curves up to genus 101 [Con10].

Since the automorphism group of small groups can be much larger than the groups themselves, we also include all automorphism groups of groups of order up to 255 and of transitive groups of degree up to 23.

2.1. **Data computed for each group.** For each group, we compute as many attributes as possible. Some use commands directly from Magma, such as determining if a group is abelian, while others require special functions we wrote (for example, determining if a group is metacyclic). We compute a "reasonable" presentation for the group (see Section 4), a minimal degree permutation representation, the lattice of subgroups up to automorphisms

of the group, and up to conjugacy when possible (Section 3.3), and conjugacy classes and the character tables when feasible. We also determine special subgroups such as the center, commutator, Frattini, and Fitting subgroups, as well as various series for the group.

♠♠♠ Sam: [Move this paragraph to §3.3?] We aim to compute the full lattice of subgroups, but there are many groups in our database where it is infeasible to compute subgroups up to conjugacy. For example, C_2^{10} has 229,755,605 subgroups, none of which are conjugate. However, it only has 11 classes of subgroups up to automorphism. We implemented the computation of the subgroup lattice up to automorphism and improved on Magma's built-in subgroup lattice methods up to conjugacy. In some cases, such as S_{47}, working up to automorphism does not sufficiently reduce the quantity of subgroups, so we restrict our attention to normal subgroups and their complements, maximal subgroups, Sylow subgroups, subgroups with small index and/or trivial core due to their importance in permutation representations.

The subgroup lattice is one example of the general challenge that many of our quantities of interest become more difficult to compute as the size or degree of the group increases. Our general approach is to try to compute everything for all groups, setting appropriate timeouts and omitting data if the computation does not finish or if we encounter errors in Magma. We have developed additional code for some problems where Magma's built-in methods were insufficient for our purposes; notable examples include computing quotients of permutation groups and isomorphism testing.

2.2. **The isomorphism problem.** When adding transitive permutation groups to our abstract group database, we had to ensure that groups with multiple transitive permutation representations were only added once each. For small groups, Magma will compute its small group identification number. In a larger range, the LMFDB's transitive group section already had the needed information. However, there were multiple cases where neither approach sufficed.

Running isomorphism tests can be time consuming, so we used the following strategy. We took multiple isomorphism invariants of a group which are quick to compute and combined them into a hash. This could be precomputed for large numbers of groups. Groups with different hashes were clearly non-isomorphic. If groups produced identical hashes, we then had Magma perform a slower, but conclusive test for isomorphism. Our hash was highly effective in distinguishing non-isomorphic groups. Full details will be given in the forthcoming article [Roe].

3. Features of the database

The database is viewable through the LMFDB website at https://www.lmfdb.org/Groups/Abstract/. In this section we describe how to navigate the web interface, and highlight some particular features.

3.1. **Features of the search interface.** Groups can be searched through numerous query types, for example: the isomorphism type of a group's automorphism group, commutator, center, abelianization, Frattini subgroup, etc.; the order of any of these and the group itself; and many boolean properties such as nilpotency, simplicity, and solubility. With the search interface, it is easy to quickly answer questions such as *"What are the nilpotent groups of order 36?"* and *"Are there any groups of order 256 with abelianization $C_2 \times C_2$?"* In this way, the database provides a useful service to researchers looking for groups with particular properties, as well as students coming to grips with the basics of group theory. It is also possible to search for groups whose orders factor in a particular way. For example, groups whose order is of the form p^3q^2r for primes p, q, r can be queried with the string $[3, 2, 1]$.

There is a curated list of some interesting groups that one can quickly access, and one can view the home page of a group picked randomly from the database as well. Each abstract group has a unique label attached to it (see Section 5 for more details), and one may search for a specific abstract group using its label or by a name such as "S5".

3.2. **Features of a group page.** On individual group pages, information is organized roughly by topic. Constructions of the group via a presentation (see section 4), and direct, semidirect and non-split product are given when possible, as well as representations as a matrix or permutation group. Homology information like Schur multiplier and commutator length are provided. Special subgroups like the center, commutator and socle are displayed. For many groups we provide a subgroup diagram up to conjugacy and up to automorphism (see section 3.3). We also display the derived series, chief series, and upper and lower central series of the group. Groups for which the given group is a maximal subgroup or a maximal quotient are listed under Supergroups.

Groups are also represented pictorially, in line with other sections of the LMFDB. Each disc in a group picture represents a conjugacy class of elements, arranged in an annulus around the middle class (of the identity), with distance according to number of prime factors in the class's order. A disc's size and color are chosen based on its order. The decisions on how to display a group in picture form are essentially a matter of taste, with some group pictures showing striking symmetries within the conjugacy classes, such as 480.60 in Figure 1.

For many groups we provide character tables for both complex and rational characters. When the data has been computed but the table is too large, the user is provided with a link to display the table. Rows correspond to irreducible characters, columns correspond to conjugacy classes for the complex character table and divisions for rational character table.

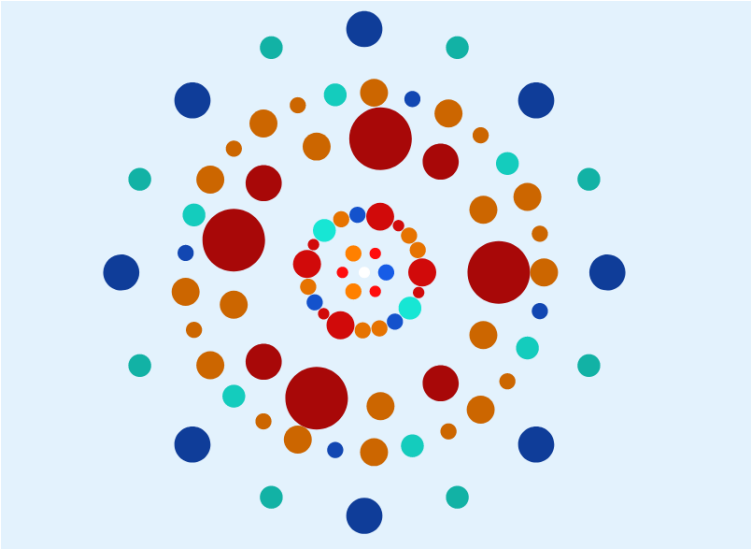

FIGURE 1. Group picture of 480.60

For ease of terminology, in this discussion we refer to columns as corresponding to "classes" to cover both cases simultaneously.

As is standard, the row(s) above the table contain power map information on classes. For each prime p dividing the order of the group and class represented by an element g, the entry above that column in the row labeled "pP" gives the label for the class of g^p. For each complex character, in the 2nd column of the table we designate whether the representation is real, complex, or quaternionic.

3.3. **Subgroups.** One of the main features of our work is a companion database of subgroups. Subgroups have their own search functionality and labeling system (see subsection 5.4). Given a group H, it is possible to search the database for all groups G containing H as a subgroup, subject to numerical constraints such as the index of H in G and the order of G. Cyclic subgroups, normal subgroups, or Sylow or Hall subgroups of a group can all be searched for. Each subgroup also has its own page. These pages give information about the ambient group and the quotient group structure (when normal). Related subgroups such as the centralizer, normalizer, normal closure and core of the subgroup are given.

The subgroup database allows us to display the subgroup lattice for many groups. This lattice is algorithmically very useful, allowing the computation of the rank of a group, its expressions as a semidirect product, etc. Magma has built-in methods for computing both the subgroup lattice and the list of subgroups up to conjugacy (without inclusions). We found that the second was much faster than the first, and that we could recover the inclusions more quickly by computing a vector counting the intersections with each conjugacy class: if $H_1 \subseteq H_2$ then the vector of counts for H_2 dominates that for H_1, dramatically decreasing the number of calls to `IsConjugateSubgroup`. When there are many subgroups up to conjugacy

(e.g., abelian p-groups), we instead compute subgroups up to automorphism. The first algorithm for doing so uses the holomorph of G, i.e., the semidirect product $G \rtimes \mathrm{Aut}(G)$. Conjugacy within the holomorph translates to automorphism in the group itself, and a code snippet kindly provided by Derek Holt using lifting through an elementary abelian series allowed for the computation of representatives of subgroups up to automorphism in the solvable case without computing the list up to conjugacy.

However, the holomorph is implemented in Magma as a permutation group of degree equal to the order of G, and as the size of G grows, computations using the holomorph bog down. For nonsolvable G and G with order at least 5000, we switch to a graph theoretic algorithm. First we compute a list of subgroups up to conjugacy, together with a list of automorphisms that generate the outer automorphism group of G. Taking subgroups up to conjugacy as vertices, we add an edge between H_1 and H_2 if there is an outer generator σ with $\sigma(H_1)$ conjugate to H_2. The components of the resulting graph give the subgroups up to automorphism.

The subgroup lattice can be viewed on a group page when it has fewer than 100 subgroup classes; see Figure 2 for an example. Inclusion is indicated with lines, and clicking a particular subgroup reveals information about its properties, e.g., whether it is maximal, solvable, its normalizer, its core, etc. Subgroups are separated into levels vertically based on their orders or by the number of prime factors in their order (the user can toggle between these two options), and can be rearranged by clicking and dragging. When the diagram is too large to conveniently display on the page, the full diagram can often be viewed on a separate linked page.

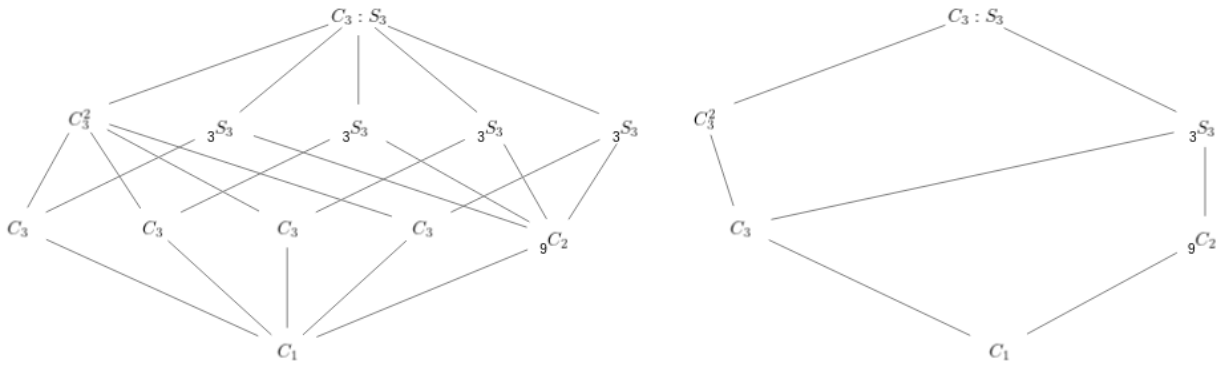

FIGURE 2. Subgroup diagrams of 18.4 up to conjugacy and autjugacy, as they appear on the LMFDB

In addition, we provide profiles of the subgroups up to automorphism and possibly up to conjugation. For each order of subgroup, the profile lists the isomorphism types and multiplicities of classes of subgroups with that type. For example, for the elementary abelian

group C_2^5, the profile for subgroups up to conjugation would list 155 conjugacy classes of subgroups of order 8 (all isomorphic to C_2^3), and one class of subgroups of the same order up to automorphism.

3.4. **Dynamic constructions.** While the database is finite, we still provide some functionality for some groups outside the database. For two classes of groups, we do computations on the fly to provide information to the user. Since pages need to be rendered quickly, these pages have less data than the group home pages for groups stored in the database.

The first type of group which is handled dynamically are groups in GAP's small group database, but which are not in the LMFDB collection. For these groups, we use GAP to compute data while rendering the page, since GAP is incorporated into SageMath. As an example, the group 256.501, one of the seven nontrivial split extensions of C_2 by OD_{128}, has automorphism group 512.402873. This group of order 512 is not in the database, but there is still a dynamically generated page with some basic information about the group such as statistics on the number of elements and conjugacy classes of each order, counts of irreducible characters of each degree, the list of maximal subgroups up to conjugacy and basic subgroups like the center and derived subgroup.

The second class of groups which are handled dynamically are abelian groups. These can be accessed by the special url of the form https://www.lmfdb.org/Groups/Abstract/ab/label where label gives the orders of the cyclic factors of the group, separated by dots, and where one can optionally specify multiplicities using underscores. So, the group $C_{12}^3 \times C_6$ can be given by label 12_3.6 or 12.12.12.6 or 12.6.12.12. Exceptionally large groups can be input using such a label since computations for them are straightforward starting from a decomposition as a product of cyclic groups. As with the first type, the pages for abelian groups give more limited information than those groups pulled directly from the database.

3.5. **Statistics.** Since the data is stored in a database structure, we are able to calculate many statistics for different order types: whether the order of a group is a prime, a product of two distinct primes, a power of one prime times another prime, etc. We collect statistics of broad interest here: https://www.lmfdb.org/Groups/Abstract/stats. For orders of groups in our database, we list distributions of different solvability types as a function of order. As two examples, we give the percentage of groups of various order types which are abelian and metacyclic but not cyclic, or supersolvable but not nilpotent nor metabelian. We also list distributions of nilpotency class and rank as a function of order, and for nonabelian groups we give distributions of automorphism group and outer automorphism group orders as a function of order. In all cases we give a numerical value as well as a percentage, and the counts shown in the statistics tables link to searches for groups of the corresponding type.

4. Computing presentations for solvable groups

Every finite solvable group is polycyclic [HEO05, §8.1], and the polycyclic presentations in Magma use one generator for each prime dividing the order of the group (counted with multiplicity). So, for example, the cyclic group of order $2^2 \cdot 3 \cdot 5$ would have a presentation with 4 generators instead of the simpler one using a single generator. Here we describe a process for finding a more human-readable polycyclic presentation for a solvable group by using chains of subgroups. One can pass between polycyclic presentations and subnormal filtrations with cyclic quotients:

(1) given a polycyclic presentation with generators $\{g_1, \ldots, g_m\}$ then

$$1 \leq H_1 \leq \cdots \leq H_m = G$$

will be a filtration of G with cyclic quotients, where $H_i = \langle g_1, \ldots, g_i \rangle$;

(2) given a filtration of the form above, any choice of elements g_i generating H_i/H_{i-1} will give generators for a polycyclic presentation.

Define the *relative order* of g_i to be the order of the quotient H_i/H_{i-1}. We approach the search for a presentation by first finding a filtration and then arbitrarily choosing generators.

We first construct all minimal length chains of subgroups, each normal in the next, with cyclic relative quotients. We do this via a "top-down" approach, building them down from the top in layers until reaching the trivial subgroup. This process guarantees a presentation with a minimal number of generators; we then compare all such minimal presentations according to the following criteria.

(1) Maximize the number of generators with order equal to their relative order.
(2) Maximize the number of generators that commute with each other.
(3) Aim for relative orders that are non-increasing.
(4) Conjugacy relations should be "deeper" (within smaller groups in the filtration).

This process cannot feasibly be made canonical: at some points we make arbitrary choices, both for the filtration and for the choice of generators.

There are some groups for which the process of constructing and comparing all minimal length chains is too time-consuming. In these cases we instead use two other algorithms, both of which run much faster, but may not give as good a presentation. The first proceeds by greedily building the chains by starting with $H = G$ and iteratively choosing a random normal subgroup $K < H$ with H/K cyclic. The second algorithm simply refines the derived series by taking an abelian basis at each step in order to fill in the series to one with cyclic quotients.

It is important to pick the presentation at the beginning of our computations for each solvable group, as this presentation impacts certain attributes of the group, such as how we represent elements of the group.

5. Labels

A foundational principle of the LMFDB is to display everything with a unique label, an identifier that carries mathematically relevant information which can ideally be computed in a deterministic way. One advantage of having labels for a type of object is that one can definitively order them. Conversely, a sequential ordering of all objects of a type trivially allows one to label them, if by nothing else, by their position in the sequence. We will talk of having labels and an ordering interchangeably. When sorting ordered tuples, we always mean lexicographically, and treat complex numbers as ordered pairs of real numbers for this purpose.

For small groups, we label them as `N.i` corresponding to the GAP ID encoded as a string, where `N` is the order of the group and `i` distinguishes groups of the same order (as determined in GAP). If a group is not in GAP Small Groups database, we replace `i` with an incrementing letter code, assigning labels to groups as they are added to our database.

The existence of automorphisms precludes having a definitive labeling *de novo*. Consequently, the labels for conjugacy classes, characters, and subgroups depend on fixing an ordered list of generators, which we do, and a specific realization of the group, i.e., as a permutation group, a polycyclic group, or as a matrix group. The latter is important since in each case, one can easily construct an injective set map from the group to $\mathbb{Z}$, which in turn orders elements of the group. We then need to have a reproducible method for generating elements of a group. In the applications below, we found that picking pseudo-random group elements worked efficiently. We first describe the pseudo-random number generator we use.

5.1. **A deterministic sequence of pseudo-random group elements.** We use an exponential pseudo-random number generator given by $a_j = a_0\, b^j \bmod p$ where

$$\begin{aligned} a_0 &= 123 \\ b &= 250962815189121053421918519178387186 29 \\ p &= 340282366920938463463374607431768211297. \end{aligned}$$

Here p is the largest prime less than 2^{128}.

To produce a pseudo-random sequence of group elements we follow a standard method. Starting with k generators, $g_1, \ldots, g_k$ we initialize the process by forming the $(k+5)$-tuple v with $v_i = g_i$ (taking subscripts modulo k). We then have 20 rounds of picking distinct indices i and j and replace v_i with $v_i \cdot v_j$. After this initialization step, we generate the sequence by again picking random subscripts i and j according to our pseudo-random sequence of integers, replacing v_i with $v_i \cdot v_j$ and returning the new v_i.

5.2. **Conjugacy classes and divisions.** If G is a group and $g_1, g_2 \in G$, we say that g_1 and g_2 are in the same *division* if there exists $h \in G$ such that $h^{-1} g_1 h = g_2^i$ for some $i \in \mathbb{Z}$ such

that $\gcd(i, |g_1|) = 1$. Divisions are unions of conjugacy classes, and rational-valued characters are constant on divisions. (In fact, the irreducible rational-valued characters form a basis for the space of functions that are constant on divisions, as complex characters do for class functions.) We say that a division is *maximal* if a representative generates a maximal cyclic subgroup of G.

Divisions are labeled with `nA` where n is a positive integer giving the order of a representative element and `A` is a capital letter which functions as a counter. They are ordered first by the size of a conjugacy class within the division, and then by the number of conjugacy classes within a division. Remaining ties are broken based on which division is seen first using a combination of the following two strategies.

The maximal divisions are partitioned into *small* and *large* divisions. For large divisions, we produce a pseudo-random sequence of group elements g_j and order those divisions based on the smallest j such that g_j is an element of the division. The conjugacy class of g_j also gives a first conjugacy class within the division, which is used below. A problem with using this method generally is that some groups contain maximal divisions which are still very small in comparison to the size of the group, so it may take a prohibitive amount of time to hit all of the divisions.

For small divisions, the basic approach is to enumerate elements and pick the smallest according to the total order on the elements of the group. If the division is small enough, we can do precisely this, but for medium-sized divisions we first attempt to narrow the number of elements considered by intersecting with non-normal subgroups of G. In order to make this process canonical, we use a modification of the big division process to choose a subgroup to intersect with. Note that Magma can quickly compute the maximal subgroups of the group G. Using the pseudo-random sequence of group elements, we record whether or not the elements are in each maximal subgroup as a vector of 0's and 1's (1 for when the element is in the subgroup). We generate enough elements so that the maximal subgroups are distinguished, sorting the maximal subgroups by these vectors, largest first. Then looking recursively at maximal subgroups of maximal subgroups, we can build part of the subgroup lattice of G from the top down, always having an ordering for the newly constructed subgroups. In this process, we ignore subgroups where the relative index is greater than 1000 as that would slow the computation. Paths down this lattice are then also ordered.

For small divisions, we intersect successively with maximal subgroups until the number of elements is relatively small (but positive). We can then enumerate elements and find the first one, based on the conversion of group elements to positive integers mentioned above. This both orders the divisions and picks a first conjugacy class within the division. Note that it is possible to track the number of elements in these intersections by using appropriate centralizers, so no enumeration is required until the number of elements drops below a threshold.

To distinguish large from small divisions, consider a division consisting of n conjugacy classes, each of size m with elements of order r. The division is considered large if $n \geq |G|/2000$ and either $n \geq 20000$ or $n^2 \geq |G|$. For divisions that don't meet these criteria, we compute the intersections down chains of maximal subgroups, greedily minimizing the intersection. If we are able to decrease the size of the intersection below a threshold we consider the division small; if the process terminates in a step where the remaining elements do not intersect nontrivially with any of the next steps in the subgroup tree, and we're still above the threshold, we also consider the original division large.

Once labels for maximal divisions are computed, we can compute labels and first elements for other classes. Working through the maximal divisions in order, we pick an element from its first conjugacy class g and compute in sequence the classes of elements g^i with $0 < i < |g|$ and $\gcd(i, |g|) > 1$. The order of the non-maximal divisions is simply the order in which they appear from this sequence (looping over divisions, and within that, powers of an element).

The label for a conjugacy class starts with the label of its division. If there is only one class within the division, the two labels are the same. When there is more than one class in a division `nA`, the conjugacy classes are labeled `nAj` where j is an integer. The first class encountered will then be `nA1`. Let g be an element representing this class. We then consider the conjugacy classes $[g^{-1}]$, $[g^2]$, $[g^{-2}]$, ... These correspond to conjugacy class labels `nA-1`, `nA2`, `nA-2`, ... The label for the class is the first time it appears in this sequence. We note that conjugacy classes which land in the same division have the same prefix, and will be grouped together in the complex character table for the group.

5.3. **Characters.** When feasible, we give information on irreducible rational-valued and complex-valued characters. The group $\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$ acts on the complex characters. The Galois orbits correspond to the rational characters, with each rational character being simply the sum of the complex characters in an orbit. We note that a rational-valued character may not arise from a rational-valued representation. The Schur index gives the smallest multiplier for a rational character so that the resulting character arises from a rational-valued representation.

Permutation representations of a group G can be realized using permutation matrices, and are thus sums of irreducible representations. Transitive permutation representations are classified by their degree n and *T-number* t [CHM98]. For a given irreducible representation, we can consider the "first" transitive permutation containing it to be the smallest pair (n, t).

The labels for rational characters are of the form `G.na` where `G` is the label for the group, `n` is the degree, and `a` is a lower-case letter which acts as a counter. The label for a complex character takes the form `G.nak` where `G.na` is the label for the corresponding rational character and `k` is positive integer which serves as a counter. In the character tables, characters are sorted by their labels viewed as a tuple (n, a) or (n, a, k), respectively. To

completely determine the labels, we just need to know the ordering among characters with the same degree.

For rational characters, we sort by (d, m, n, t) where d is the degree (which is explicitly given in the label), m is the size of the Galois orbit of complex characters giving rise to it, and n and t refer to the smallest containing permutation representation. Any remaining ties are broken by sorting on the vector of character values using our ordering of conjugacy classes.

Complex irreducible characters are given in the same order as their corresponding rational characters, with characters within a Galois orbit ordered by their vectors of values.

5.4. **Subgroups.** Most[1] labels for subgroups $H \leq G$ take the form `n.i.m.a.c`. The piece `n.i` is the label of the ambient group G, `m` is the index of H in G, and `a` and `c` distinguish H up to automorphism and conjugacy, respectively. For some ambient groups we only compute and display subgroups up to automorphism, in which case the label takes the reduced form `n.i.m.a`. To determine a and c we use the idea of Gassmann equivalence classes to further order subgroups of the same index. (For more on Gassmann equivalence and related notions, see [Sut21].)

Definition 5.1. Two subgroups H_1 and H_2 of G are *Gassmann equivalent* if they intersect each G-conjugacy class with the same cardinality. Equivalently, H_1 and H_2 have the same index m and the permutation representations $\pi_{H_1} : G \to S_m$ and $\pi_{H_2} : G \to S_m$ have the same character (which counts the number of cosets fixed by each conjugacy class).

In order to label Gassmann classes we first fix an ordering of the conjugacy classes of G as in Section 5.2. Having fixed such an ordering, it is easy to compute a unique identifier for each Gassmann class: we count the intersection of H with each G-conjugacy class. We enumerate H-conjugacy classes of elements of H in any order, then determine which G-conjugacy class they belong to, adding the size of the H-conjugacy class to a running tally of the intersection. An analogous process works for Gassman vectors up to automorphism, where we collect together G-conjugacy classes that are related by an automorphism of G.

If there are multiple subgroups up to automorphism (resp., conjugation) within a given Gassmann class, we use the subgroup lattice to further order those remaining subgroups. We first order subgroups H in the same Gassmann class using the lex ordering on sorted list of labels of all (proper) supergroups of H (one can and often does have Gassmann equivalent subgroups for which the lists of supergroups differ). Note that here "supergroup" refers to inclusions in the poset of subgroups up to automorphism (resp., conjugation), meaning we consider K to be a supergroup of H if it contains any subgroup equivalent to H up to automorphism (resp., conjugation).

[1]The exceptions are detailed at the end of this section.

In these cases where two or more Gassmann equivalent subgroups have the same set of supergroups, we resort to computing permutation representation signatures. We can, in fact, compute with any supergroup K of H instead of G. This is much more efficient when the index $[K : H]$ is much smaller than $[G : H]$, but does require some care to account for the fact that there may be two or more G-conjugates of H contained in K that are not K-conjugate.

Once we have an ordering inside the Gassmann classes, we assign a letter label to the Gassmann class and concatenate a number from the ordering of subgroups inside the Gassmann class to create an alpha-numeric value which we assign to a and (whenever we can compute up to conjugation) we similarly assign an alpha-numeric value to c. In order to keep labels small, we assume that we compute all the Gassmann classes for a given index so that we can assign ordinals to them.

In some cases, we are unable to compute the whole subgroup lattice, or there are enough subgroups that we do not want to store all of them. When possible, we compute subgroups up to a certain index bound, and the labeling scheme described above still works in that setting. But there are often some subgroups above the index bound that we want to keep, which fall into certain categories.

(1) Since Sylow subgroups are the unique subgroup of their order up to conjugacy, we can compute their labels a priori.
(2) We aim to compute the lattice of normal subgroups even when we cannot find all subgroups. In this case, we can use the same scheme described above applied to the lattice or normal subgroups to obtain a label with the letter `.N` appended; such labels are used for non-Sylow normal subgroups above the index bound.
(3) We store complements of normal subgroups even above the index bound so that we can describe groups as semidirect products. Given a normal subgroup with label `n.i.m.a.c.N`, we label its complements `n.i.m.a.c.N1`, `n.i.m.a.c.N2`, ... Unfortunately, these labels are not canonical since they depend on the order that complements are produced. Also note that a subgroup can appear as a complement of multiple normal subgroups; in this case we just label it based on the first normal subgroup where it arises.
(4) For non-Sylow, non-normal maximal subgroups above the index bound, we apply the original labeling scheme restricted to maximal subgroups, and append the letter `.M`.
(5) We also keep core-free subgroups above the index bound rather than discarding them, since these yield transitive permutation representations. The labels here are just `n.i.m.CFk`, where `k` is a counter arbitrarily running over the core-free subgroups of a given index.

6. Connections and examples

We anticipate that the group home pages will be a valuable tool for everyone from undergraduate students first learning abstract algebra to experts in the field. The layout of each page, the search features, and the knowls (the myriad of hyperlinks giving definitions of most of the words on each page) allow for exploration of many group theoretic concepts. Students can easily find examples of groups with common features like simple or solvable groups; they can explore Sylow subgroups (and more generally subgroup lattices); and they can learn about advanced topics such as series or character theory. Students can also explore relationships between different attributes of groups. A quick search demonstrates examples of nontrivial groups that are perfect but not simple or A-groups (a group with all its Sylow subgroups abelian) that are not solvable.

For researchers, the groups database is intimately connected with many other mathematical objects in the LMFDB. We highlight some of those connections below.

6.1. **Connections with other objects.** One of the integral parts of the LMFDB is demonstrating links between different mathematical objects. The LMFDB universe gives the big picture of various connections, which fall under the *Langlands program*. Moreover, there are many related objects and links between them are displayed in the database as well. In this section we list other pages in the LMFDB which are linked with Abstract Groups page. We describe the various connections to other LMFDB pages with the group page in Figure 3.

In Figure 3, the boxes representing LMFDB sections Belyi maps, Hypergeometric motives over $\mathbb{Q}$, Lattices, and Modular curves are only available on the beta version of the LMFDB (https://beta.lmfdb.org/). Artin representations connect to Abstract groups via both their image and their projective images. One can also get from Artin representations to Galois groups (i.e., transitive permutation groups) by taking a stem field for the field cut out by the image of the representation to get a number field, and then taking the Galois group of its Galois closure. For a number field, one has the Artin representations of the Galois group of the normal closure of the field.

Example 6.1. Many objects in the LMFDB link to groups. Having a built-in group database makes it easy for the user to find more information about the groups in question. Some groups may be familiar to the user, such as those which arise as automorphism groups of genus 2 curves; others are more complicated. Consider the Artin representation https://www.lmfdb.org/ArtinRepresentation/4.5744.8t39.d.a which has the smallest conductor of all 4-dimensional irreducible Artin representations with group https://www.lmfdb.org/Groups/Abstract/192.1493 [JR17]. This group is displayed as $C_2^3 : S_4$, but since the semidirect product notation may not uniquely identify a group, the connection to a specific group page lets the user obtain lots of information about this group. Similarly,

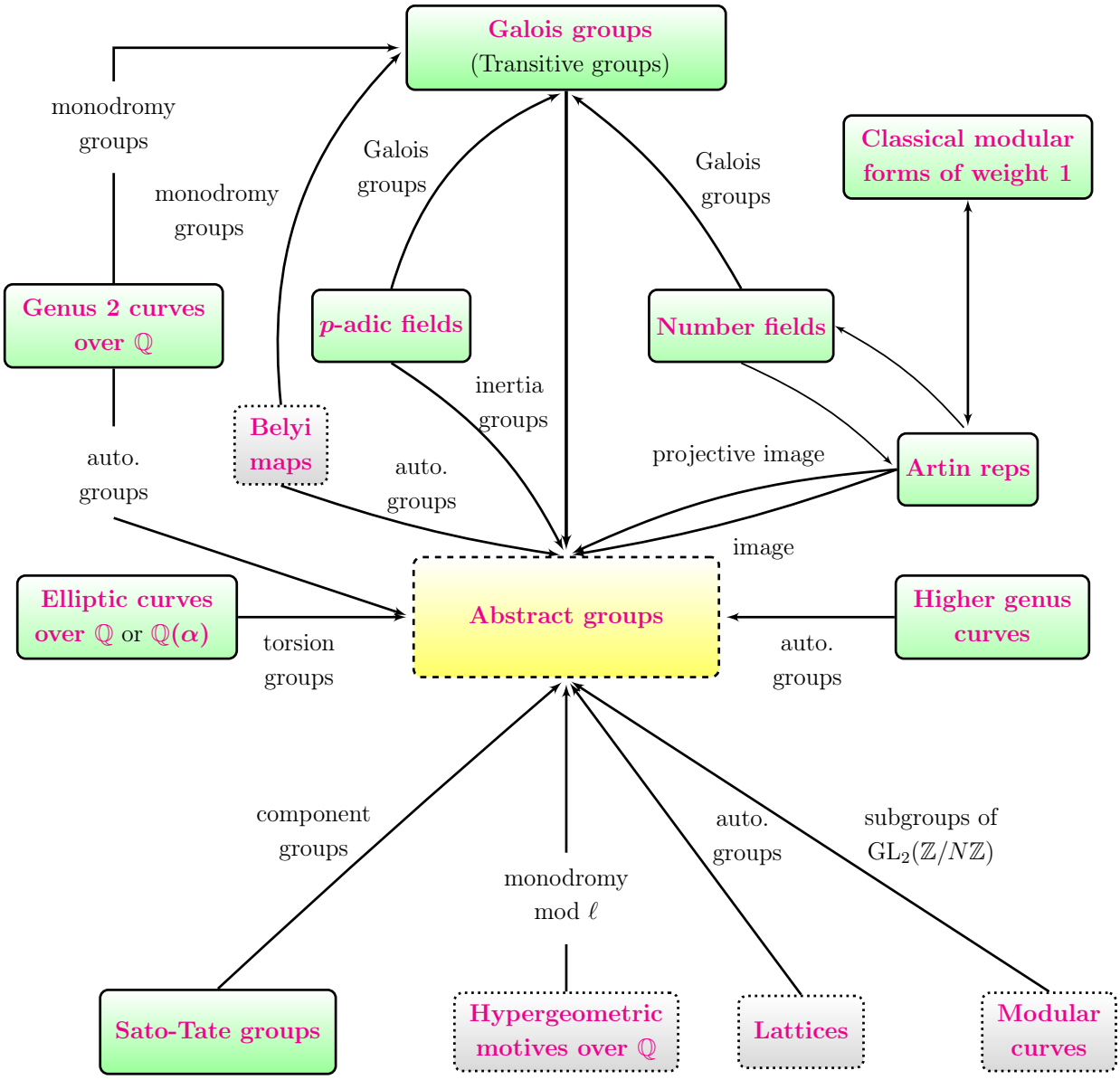

FIGURE 3. Diagram illustrating the connections between abstract group pages and other collections of objects in the LMFDB. Nodes in solid (green) and dotted (grey) rectangles are on the production and beta version of the LMFDB, respectively.

the projective image of this group is $C_2^2 : S_4$, to which the user is referred to the page https://www.lmfdb.org/Groups/Abstract/96.227 for more details.

6.2. **Applications via Galois theory.** Perhaps the most celebrated application of groups is their usage in studying field extensions as a part of Galois theory. Below we give examples

of how our database of finite groups, together with the Galois correspondence, can be used to study objects in other collections in the LMFDB.

Example 6.2. Let $F = \mathbb{Q}(\alpha)$ be the number field with LMFDB label 5.1.35152.1, where α has minimal polynomial $f := x^5 - x^4 + 2x^3 - 4x^2 + x - 1$. Then F has Galois closure $L = \mathbb{Q}(\beta)$ with LMFDB label 20.0.3354518684571451850752.1, where β has minimal polynomial

$$x^{20} - 2x^{19} + 10x^{17} - 15x^{16} + 40x^{14} - 64x^{13} + 46x^{12} + 8x^{11} - 32x^{10} + 8x^9 + 46x^8 - 64x^7 + 40x^6 - 15x^4 + 10x^3 - 2x + 1\,,$$

and $G := \mathrm{Gal}(L/\mathbb{Q}) \cong F_5$, the Frobenius group of order 20 (with LMFDB label 20.3.) Examining the lattice of subgroups up to conjugacy given on the 20.3 homepage, we see that the extension F_5 is solvable, so the roots of f can be expressed by radicals. Computing fixed fields, we obtain the subfield lattice shown in Figure 4, where K and E are the number fields

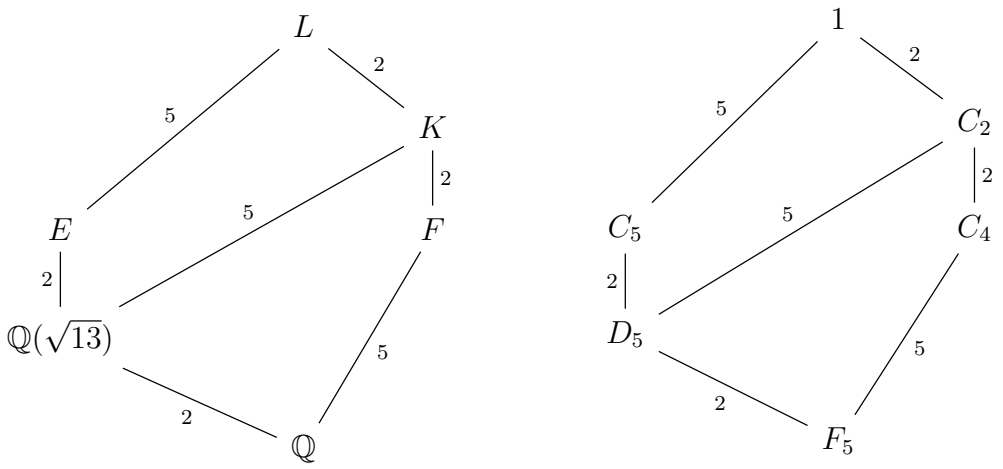

FIGURE 4. The subfield lattice of the number field with LMFDB label 20.0.3354518684571451850752.1 and the subgroup lattice of its Galois group F_5.

given by

$$x^{10} - x^9 - 3x^8 + 5x^6 + x^5 - 5x^4 + 3x^2 - x - 1$$

and

$$x^4 + x^3 + 2x^2 - 4x + 3\,,$$

respectively. We observe that the commutator subgroup of G is C_5, the subgroup corresponding to E. Thus E is the largest abelian subfield of L, with Galois group $\mathrm{Gal}(E/\mathbb{Q}) \cong G^{\mathrm{ab}} \cong C_4$.

Example 6.3. Let $K = \mathbb{Q}_2(\alpha)$ be the p-adic field with LMFDB label 2.4.6.7, where α has minimal polynomial $f := x^4 + 2x^3 + 2x^2 + 2$. Note that K is not Galois over $\mathbb{Q}_2$ and K has

Galois closure $K^{\mathrm{gal}} = \mathbb{Q}_2(\beta)$ with LMFDB label 2.12.18.59, where β has minimal polynomial

$$x^{12} - 2x^{11} + 6x^{10} + 4x^9 + 6x^8 + 12x^7 - 4x^6 - 8x^3 + 16x^2 - 8$$

and $G := \mathrm{Gal}(K^{\mathrm{gal}}/\mathbb{Q}_2) \cong A_4$, the alternating group of order 12 (with LMFDB label 12.3). The inertia group $I := I(K^{\mathrm{gal}}/\mathbb{Q}_2)$ of K^{gal} is the abelian group C_2^2 of order (with LMFDB label 4.2). The wild inertia group of K^{gal} (i.e., the unique 2-Sylow subgroup of I) in this case is equal to I.

Examining the lattice of subgroups up to conjugacy given on the 12.3 homepage, we obtain the subfield lattice of $K^{\mathrm{gal}}/\mathbb{Q}_2$, shown in Figure 5. It is clear from the subgroup lattice of 12.3 that $\mathrm{Gal}(K^{\mathrm{gal}}/K) = C_3$ and the extension $K/\mathbb{Q}_2$ is primitive. Also note that A_4 is solvable of length 2, so we find two intermediate fields E and F of K^{gal} such that $\mathrm{Gal}(K^{\mathrm{gal}}/E) = C_2^2$ and $\mathrm{Gal}(K^{\mathrm{gal}}/F) = C_2$. Computing fixed fields, we obtain the subfield lattice shown in Figure 5, where E and F are the p-adic fields given by $x^3 - x + 1$ and $x^6 + x^2 - 1$, respectively. The LMFDB labels for E and F are 2.3.0.1 and 2.6.6.1, respectively. We observe that the commutator subgroup of G is C_2^2, the subgroup corresponding to E. Thus E is the largest abelian subfield of K^{gal}, with Galois group $\mathrm{Gal}(E/\mathbb{Q}) \cong G^{\mathrm{ab}} \cong C_3$.

This is an interesting example since K is the only degree 4 extension of $\mathbb{Q}_p$, for any p, which has Galois group $\mathrm{Gal}(K^{\mathrm{gal}}/\mathbb{Q}_2) \cong A_4$.

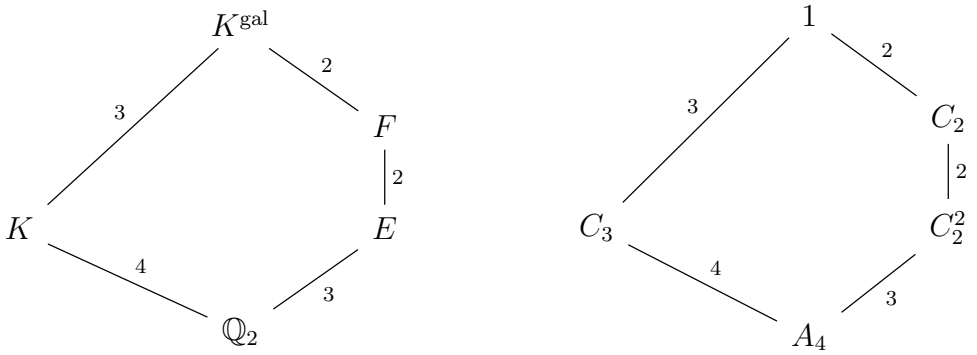

FIGURE 5. The subfield lattice of the Galois closure of the p-adic field with LMFDB label 2.4.6.7 and the subgroup lattice of its Galois group A_4.

Example 6.4. Let G be the Sato-Tate group $J(O)$ with LMFDB label 1.4.F.48.48a. Let G^0 be the connected component of the identity and G/G^0 be the component group of G. As noted on its homepage, $J(O)$ has the largest component group ($C_2 \times S_4$, LMFDB label 48.48, with order 48) among Sato-Tate groups of abelian surfaces over number fields.

Let A be an abelian variety of dimension $g \leq 3$ defined over a number field k. Let K be the minimal extension of k over which all endomorphisms of A are defined, i.e., such that $\mathrm{End}(A_K) = \mathrm{End}(A_{\overline{k}})$. By [FKRS12, Proposition 2.17], then $\mathrm{ST}_A \,/\, \mathrm{ST}_A^0 \cong \mathrm{Gal}(K/k)$, where ST_A is the Sato-Tate group of A.

Let $C : y^2 = x^6 - 5x^4 + 10x^3 - 5x^2 + 2x - 1$ considered over $\mathbb{Q}$, and let $A = \mathrm{Jac}(C)$. As shown in [FKRS12], the variety A has Sato-Tate group $J(O)$, realizing this group over $\mathbb{Q}$, and the endomorphisms of A are defined over the number field $K = \mathbb{Q}(\sqrt{-2}, \sqrt{-11}, a, b)$ where

$$a^3 - 7a + 7 = 0 \qquad \text{and} \qquad b^4 + 4b^2 + 8b + 8 = 0\,.$$

Applying the above proposition to this example, we have

$$C_2 \times S_4 \cong G/G^0 \cong \mathrm{Gal}(K/k)\,.$$

However, as shown in [FKRS12, Table 4], the curve C can also be used to realize 24 other Sato-Tate groups by varying the base field. In other words, by taking a number field L with $k \subseteq L \subseteq K$ and considering the base change A_L, we can obtain other Sato-Tate groups. For instance, examining [FKRS12, Table 8] we see that $J(T)$ (LMFDB label `1.4.F.24.13a`) is the unique Sato-Tate group occurring in genus 2 with component group $C_2 \times A_4$. Computing the fixed field L of $C_2 \times A_4 \leq C_2 \times S_4$, we find that $L = \mathbb{Q}(\sqrt{-11})$. Thus the base change A_L has Sato-Tate group $J(T)$, realizing another of the 52 possible Sato-Tate groups.

With further calculation, one can use other base changes of A to realize other Sato-Tate groups whose component groups are subgroups of $C_2 \times S_4$ by considering the lattice of subgroups given on the homepage for $C_2 \times S_4$ and computing fixed fields.

Galois theory can also be applied in a geometric context. The equivalence of categories between function field extensions in one variable and nonsingular projective curves (see [Sta18, Tag 0BXX], for instance) allows us to study nonconstant morphisms of curves by examining the associated extension of function fields. We consider an example coming from the Families of higher genus curves with automorphisms collection, available at `https://www.lmfdb.org/HigherGenus/C/Aut/`.

Example 6.5. Consider the refined passport with label `3.168-42.0.2-3-7`. It corresponds to a unique topological equivalence class of morphisms $X \to X/\operatorname{Aut}(X) \cong \mathbb{P}^1$, where X is the Klein quartic curve, a genus 3 curve with the largest possible number of automorphisms for its genus. (Such curves are known as *Hurwitz curves*.) It has automorphism group $\operatorname{Aut}(X) \cong \mathrm{PSL}_2(\mathbb{F}_7)$ (which has LMFDB label `168.42`), and all automorphisms are defined over the field $K = \mathbb{Q}(\zeta_7)$. By examining the lattice of subgroups of $\mathrm{PSL}_2(\mathbb{F}_7)$ given on its homepage and applying the Galois correspondence, we can find all intermediate covers Y with $X \to Y \to X/\operatorname{Aut}(X) \cong \mathbb{P}^1$.

Computationally, this can be accomplished using Magma's `CurveQuotient` command. Calling this on each subgroup of $\mathrm{PSL}_2(\mathbb{F}_7)$ in turn, we find three intermediate covers Y of genus 1, corresponding to the subgroups of $\mathrm{PSL}_2(\mathbb{F}_7)$ isomorphic to C_2, C_3, and C_4. (All other quotients by nontrivial subgroups of $\operatorname{Aut}(X)$ result in genus 0 curves.) Each of these curves can be equipped with the structure of an elliptic curve by taking as the origin of the group law the image of $(1 : 0 : 0)$ under the appropriate quotient map. With some further

computation (see [Elk99, equation 2.10]), one can show that each of these elliptic curves is isomorphic to the curve $E : y^2 = 4x^3 + 21x^2 + 28x$ over K. From this we observe that $\mathrm{Jac}(X)$ decomposes as E^3 up to isogeny.

6.3. **Modular curves and subgroups of $\mathbf{GL_2(\mathbb{Z}/N\mathbb{Z})}$.** The LMFDB currently has a preliminary database of modular curves, available at https://alpha.lmfdb.org/ModularCurve/Q/. We conclude this section by describing how properties of modular curves can be deduced from their corresponding subgroups of $\mathrm{GL}_2(\mathbb{Z}/N\mathbb{Z})$. For a more comprehensive exposition of modular curves and subgroups of $\mathrm{GL}_2(\widehat{\mathbb{Z}})$, see [RSZB22] or [Zyw22].

Let E be an elliptic curve over $\mathbb{Q}$ and $E[N]$ be its N-torsion subgroup for each $N \in \mathbb{Z}_{\geq 1}$. The absolute Galois group $G_{\mathbb{Q}} := \mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$ acts on $E[N]$, and since $E[N] \cong (\mathbb{Z}/N\mathbb{Z})^2$ as abelian groups, we obtain a representation

$$\rho_{E,N} : G_{\mathbb{Q}} \to \mathrm{Aut}(E[N]) \cong \mathrm{GL}_2(\mathbb{Z}/N\mathbb{Z})\,.$$

By choosing compatible bases, we can take the inverse limit and package these together as a single representation

$$\rho_E : G_{\mathbb{Q}} \to \varprojlim_N \mathrm{GL}_2(\mathbb{Z}/N\mathbb{Z}) = \mathrm{GL}_2(\widehat{\mathbb{Z}})\,.$$

If E does not have complex multiplication, then Serre's Open Image Theorem [Ser72] implies that the image of ρ_E is an open subgroup of $\mathrm{GL}_2(\widehat{\mathbb{Z}})$, hence has finite index. Given an open subgroup H of $\mathrm{GL}_2(\widehat{\mathbb{Z}})$, one can define the modular curve X_H whose K-points parametrize elliptic curves E/K such that $\mathrm{img}(\rho_E) \subseteq H$ (up to conjugation). (For a precise definition of X_H, see [RSZB22, §2.3] or [Zyw22, §3].)

For each $N \in \mathbb{Z}_{\geq 1}$, let $\pi_N : \mathrm{GL}_2(\widehat{\mathbb{Z}}) \to \mathrm{GL}_2(\mathbb{Z}/N\mathbb{Z})$ be the projection map. Every open subgroup $H \leq \mathrm{GL}_2(\widehat{\mathbb{Z}})$ contains $\ker(\pi_N)$ for some N, and the smallest such $N \in \mathbb{Z}_{\geq 1}$ is called the *level* of H. If H contains $\ker(\pi_N)$, then H can be recovered from $\pi_N(H)$ as $H = \pi_N^{-1}(\pi_N(H))$. Thus to store H on a computer, we can simply store generators for $\pi_N(H)$ where N is the level of H.

Although this is a database of modular curves, much of the geometric data is computed group theoretically. Let $H \leq \mathrm{GL}_2(\widehat{\mathbb{Z}})$ be an open subgroup of level N. Letting $\Gamma_H := \pm\pi_N(H) \cap \mathrm{SL}_2(\mathbb{Z}/N\mathbb{Z})$, then one can determine the number of elliptic points and cusps of X_H by studying the action of the matrices

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

on the coset space $\Gamma_H \backslash \mathrm{SL}_2(\mathbb{Z}/N\mathbb{Z})$. This data can in turn be used to compute the genus of X_H.

Example 6.6. Consider the modular curve $X_0(6)$ of level 6 with LMFDB label 6.12.0.a.1. As a containment $H \subseteq H'$ of open subgroups of $\mathrm{GL}_2(\widehat{\mathbb{Z}})$ induces a morphism $X_H \to X_{H'}$ of

modular curves, we have morphisms $X_0(6) \to X_0(2)$ and $X_0(6) \to X_0(3)$. On the homepage for $X_0(6)$, it is further claimed that $X_0(6)$ is the fiber product of $X_0(2)$ and $X_0(3)$ over $X(1)$—we explain how this can be determined group theoretically.

Let H_2, H_3, H_6 be the subgroups of $\mathrm{GL}_2(\mathbb{Z}/2\mathbb{Z})$, $\mathrm{GL}_2(\mathbb{Z}/3\mathbb{Z})$, $\mathrm{GL}_2(\mathbb{Z}/6\mathbb{Z})$ corresponding to $X_0(2)$ (with label 2.3.0.a.1), $X_0(3)$ (with label 3.4.0.a.1), and $X_0(6)$, respectively. Note that $\mathrm{GL}_2(\mathbb{Z}/6\mathbb{Z})$ has label 288.851 and the subgroup $H_6 \cong C_2^2 \times S_3$ has subgroup label 288.851.12.c1.a1. Taking the inverse image of H_2 and H_3 under the projection maps $\pi_{6,2} : \mathrm{GL}_2(\mathbb{Z}/6\mathbb{Z}) \to \mathrm{GL}_2(\mathbb{Z}/2\mathbb{Z})$, $\pi_{6,3} : \mathrm{GL}_2(\mathbb{Z}/6\mathbb{Z}) \to \mathrm{GL}_2(\mathbb{Z}/3\mathbb{Z})$, we obtain subgroups $\widetilde{H_2} := \pi_{6,2}^{-1}(H_2)$ with subgroup label 288.851.3.a1.a1 and $\widetilde{H_3} := \pi_{6,3}^{-1}(H_3)$ with subgroup label 288.851.4.b1.a1.

Either by examining the (very large) subgroup lattice diagram of $\mathrm{GL}_2(\mathbb{Z}/6\mathbb{Z})$ or by computing directly, we find that $\widetilde{H_2} \cap \widetilde{H_3} = H_6$ and $\langle \widetilde{H_2}, \widetilde{H_3} \rangle = \mathrm{GL}_2(\mathbb{Z}/6\mathbb{Z})$. This shows that H_6 is the fiber product of $\widetilde{H_2}$ and $\widetilde{H_3}$ over $\mathrm{GL}_2(\mathbb{Z}/6\mathbb{Z})$, and hence $X_0(6)$ is the fiber product of $X_0(2)$ and $X_0(3)$ over $X(1)$, as depicted in Figure 6.

$$\begin{array}{ccc} X_0(6) & \longrightarrow & X_0(2) \\ \downarrow & \ulcorner & \downarrow \\ X_0(3) & \longrightarrow & X(1) \end{array} \qquad \begin{array}{ccc} H_6 & \longrightarrow & \widetilde{H_2} \\ \downarrow & \ulcorner & \downarrow \\ \widetilde{H_3} & \longrightarrow & \mathrm{GL}_2(\mathbb{Z}/6\mathbb{Z}) \end{array}$$

FIGURE 6. The fiber product diagram for the modular curve $X_0(6)$, and the fiber product diagram of the corresponding subgroups of $\mathrm{GL}_2(\mathbb{Z}/6\mathbb{Z})$.

REFERENCES

[BCH+] Greg Butler, John Cannon, Derek Holt, Alexander Hulpke, John McKay, and Gordon Royle. Database of transitive groups. https://magma.maths.usyd.edu.au/magma/handbook/text/781. 2

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993). 2, 4

[BE99a] Hans Ulrich Besche and Bettina Eick. Construction of finite groups. *J. Symbolic Comput.*, 27(4):387–404, 1999. 5

[BE99b] Hans Ulrich Besche and Bettina Eick. The groups of order at most 1000 except 512 and 768. *J. Symbolic Comput.*, 27(4):405–413, 1999. 5

[BE01] Hans Ulrich Besche and Bettina Eick. The groups of order $q^n \cdot p$. *Comm. Algebra*, 29(4):1759–1772, 2001. 5

[BEO01] Hans Ulrich Besche, Bettina Eick, and E. A. O'Brien. The groups of order at most 2000. *Electron. Res. Announc. Amer. Math. Soc.*, 7:1–4, 2001. 5

[BEO02] Hans Ulrich Besche, Bettina Eick, and E. A. O'Brien. A millennium project: constructing small groups. *Internat. J. Algebra Comput.*, 12(5):623–644, 2002. 2

[Bre00] Thomas Breuer. *Characters and automorphism groups of compact Riemann surfaces*, volume 280 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 2000. 5

[CCN+85] J. H. Conway, R. T. Curtis, S. P. Norton, R. A. Parker, and R. A. Wilson. $\mathbb{ATLAS}$ *of finite groups*. Oxford University Press, Eynsham, 1985. Maximal subgroups and ordinary characters for simple groups, With computational assistance from J. G. Thackray. 2

[CH08] John J. Cannon and Derek F. Holt. The transitive permutation groups of degree 32. *Experiment. Math.*, 17(3):307–314, 2008. 5

[CHM98] John H. Conway, Alexander Hulpke, and John McKay. On transitive permutation groups. *LMS J. Comput. Math.*, 1:1–8 (electronic), 1998. 14

[Con10] Marston Conder. Group actions on surfaces. https://www.math.auckland.ac.nz/~conder/BigSurfaceActions-Genus2to101-ByGenus.txt, 2010. 5

[DE05] Heiko Dietrich and Bettina Eick. On the groups of cube-free order. *J. Algebra*, 292(1):122–137, 2005. 5

[DE12] Heiko Dietrich and Bettina Eick. Addendum to "On the groups of cube-free order" [J. Algebra 292 (1) (2005) 122–137] [mr2166799]. *J. Algebra*, 367:247–248, 2012. 5

[DEP22] Heiko Dietrich, Bettina Eick, and Xueyu Pan. Groups whose orders factorise into at most four primes. *J. Symbolic Comput.*, 108:23–40, 2022. 5

[Dok] Tim Dokchitser. GroupNames. http://groupnames.org. 2

[Elk99] Noam D. Elkies. The Klein quartic in number theory. In *The eightfold way*, volume 35 of *Math. Sci. Res. Inst. Publ.*, pages 51–101. Cambridge Univ. Press, Cambridge, 1999. 22

[FKRS12] Francesc Fité, Kiran S. Kedlaya, Víctor Rotger, and Andrew V. Sutherland. Sato–Tate distributions and Galois endomorphism modules in genus 2. *Compositio Mathematica*, 148(5):1390–1442, 2012. 20, 21

[GAP21] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.11.1*, 2021. 2

[HEO05] Derek F. Holt, Bettina Eick, and E. A. O'Brien. *Handbook of Computational Group Theory*. Discrete Mathematics and Its Applications. CRC Press, 2005. 11

[HR20] Derek Holt and Gordon Royle. A census of small transitive groups and vertex-transitive graphs. *J. Symbolic Comput.*, 101:51–60, 2020. 5

[HRT01] R. B. Howlett, L. J. Rylands, and D. E. Taylor. Matrix generators for exceptional groups of Lie type. *J. Symbolic Comput.*, 31(4):429–445, 2001. 5

[Hul] Alexander Hulpke. Transitive groups library - a GAP package, version 3.0. https://www.gap-system.org/Packages/transgrp.html. 2

[Hul05] Alexander Hulpke. Constructing transitive permutation groups. *J. Symbolic Comput.*, 39(1):1–30, 2005. 5

[JR17] John W. Jones and David P. Roberts. Artin L-functions of small conductor. *Res. Number Theory*, 3:Paper No. 16, 33, 2017. 17

[NOVL04] M. F. Newman, E. A. O'Brien, and M. R. Vaughan-Lee. Groups and nilpotent Lie rings whose order is the sixth power of a prime. *J. Algebra*, 278(1):383–401, 2004. 5

[O'B90] E. A. O'Brien. The p-group generation algorithm. volume 9, pages 677–698. 1990. Computational group theory, Part 1. 5

[O'B91] E. A. O'Brien. The groups of order 256. *J. Algebra*, 143(1):219–235, 1991. 5

[OPS08] J. Opgenorth, W. Plesken, and T. Schulz. Carat, crystallographic algorithms and tables, Version 2.1b1. https://github.com/lbfm-rwth/carat/, Jul 2008. 5

[OVL05] E. A. O'Brien and M. R. Vaughan-Lee. The groups with order p^7 for odd prime p. *J. Algebra*, 292(1):243–258, 2005. 5

[Roe] David Roe. A fast nonisomorphism test for finite groups. In preparation. 6

[RSZB22] Jeremy Rouse, Andrew V. Sutherland, and David Zureick-Brown. ℓ-adic images of Galois for elliptic curves over $\mathbb{Q}$ (and an appendix with John Voight). *Forum Math. Sigma*, 10:Paper No. e62, 63, 2022. With an appendix with John Voight. 22

[RT98] L. J. Rylands and D. E. Taylor. Matrix generators for the orthogonal groups. *J. Symbolic Comput.*, 25(3):351–360, 1998. 5

[Ser72] Jean-Pierre Serre. Propriétés galoisiennes des points d'ordre fini des courbes elliptiques. *Invent. Math.*, 15(4):259–331, 1972. 22

[Sta18] The Stacks Project Authors. *Stacks Project*. https://stacks.math.columbia.edu, 2018. 21

[Sut21] Andrew V. Sutherland. Stronger arithmetic equivalence. *Discrete Anal.*, pages Paper No. 23, 23, 2021. 15

[Tan11] O. Tange. GNU parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, Feb 2011. 4

[Tay87] Don Taylor. Pairs of generators for matrix groups. I. Cayley Bulletin 3, 1987. 5

[VRD09] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. 4

[WWT+] Robert Wilson, Peter Walsh, Jonathan Tripp, Ibrahim Suleiman, Richard Parker, Simon Norton, Simon Nickerson, Steve Linton, John Bray, and Rachel Abbott. Atlas of finite group representations - version 3. http://brauer.maths.qmul.ac.uk/Atlas/v3/. 2, 5

[Zyw22] David Zywina. Explicit open images for elliptic curves over $\mathbb{Q}$. https://arxiv.org/abs/2206.14959, 2022. 22

Lewis Combes, School of Mathematics and Statistics, University of Sheffield, UK, S3 7RH
Email address: lmcombes1@sheffield.ac.uk

John Jones, Department of Mathematics and Statistical Sciences, P.O. Box 871804, Arizona State University, Tempe, AZ, 85287
Email address: jj@asu.edu

Jennifer Paulhus, Department of Mathematics and Statistics, Grinnell College, Grinnell, IA, 50112
Email address: paulhus@math.grinnell.edu

David Roe, Massachusetts Institute of Technology, Department of Mathematics, 77 Massachusetts Ave., Bldg. 2-336 Cambridge, MA 02139, United States of America
Email address: roed@mit.edu

Manami Roy, Department of Mathematics, Fordham University, Bronx, New York, USA
Email address: mroy17@fordham.edu

Sam Schiavone, Massachusetts Institute of Technology, Department of Mathematics, 77 Massachusetts Ave., Bldg. 2-336 Cambridge, MA 02139, United States of America
Email address: sam.schiavone@gmail.com