URA Midterm Report - "Refining the Lattice Package"
Joseph Thomas
Faculty Adviser: Klaus Lux
10/24/08

In the previous semester, we considered some fundamental ideas in the theory of group representations over finite fields. Specifically, we studied Norton's Irreducibility Criterion and Parker's Meat-Axe algorithm, two tools for investigating the reducibility of a representation. Given these tools, we wish to consider the more complicated problem of finding all subrepresentations of a group representation over a finite field.

In our previous report, we said that if G is a group, a group representation $\varphi : G \to GL(n, \mathbb{F})$ is *reducible* if there exists a nontrivial proper subspace of $\mathbb{F}^n$ that is invariant under the elements of $\varphi(G)$. We also saw that for every proper, nontrivial subspace S invariant under $\varphi(G)$, we could devise two simpler representations from φ, one describing the action of $\varphi(G)$ on S and the other describing the action of $\varphi(G)$ on the quotient space $\mathbb{F}^n/S$. Thus, one way to find all subrepresentations of a given group representation φ is to find all of the subspaces of $\mathbb{F}^n$ invariant under the elements in $\varphi(G)$.

Since the vector spaces in question are over finite fields, one might naïvely try to solve this problem by repeatedly applying the generators of $\varphi(G)$ to each of the vectors in $\mathbb{F}^n$. Because many of the representations we want to study involve vector spaces of dimension 100 or more, this approach is computationally infeasible. A more tractable approach, implemented by Lux et al. is to deduce a set of spaces P such that each member is invariant under $\varphi(G)$ and contains maximally a unique space invariant under $\varphi(G)$. The spaces in P can then be used to deduce all of the other spaces invariant under $\varphi(G)$. Crucially, once we determine P, the theory tells us all other invariant subspaces can be expressed as the sums of spaces in P. These calculations, which amount to determining subspace relationships, are considerably less time consuming than determining the images of vectors under the elements of $\varphi(G)$.

Before we discuss this procedure in more detail, it may be interesting to consider two applications of this algorithm. In coding theory, many encoding schemes describe a code as a subspace of a vector space $\mathbb{F}^n$, where $\mathbb{F}$ is a finite field. Given a group G and a representation $\varphi : G \to GL_n(\mathbb{F})$, we can consider the subspaces of $\mathbb{F}^n$ invariant under $\varphi(G)$ as codes. G is then an automorphism group of these codes. By making a clever choice of G and then using our program to find the subspaces of $\mathbb{F}^n$ invariant under $\varphi(G)$, one can produce codes with useful properties. This is famously true of the Golay code, which has the sporadic Mathieu group M_{23} as its automorphism group.

Our program can also be applied to the problem of determining whether a group representation is *induced*, a property we define as follows: Suppose H is a subgroup of a finite group G, and $[G : H] = m$. We say that a representation $\varphi : G \to GL_n(\mathbb{F})$ is induced if there exists a subspace $S \subset \mathbb{F}^n$ with the following properties:

- S is invariant under $\varphi(H)$
- $S^G = \{\varphi(g)(S) | g \in G\}$ has cardinality $[G : H]$
- The direct sum of the spaces in S^G is isomorphic to $\mathbb{F}^n$
- The elements of $\varphi(G)$ permute the elements of S^G in the same way that the elements of G permute the cosets of H.

Thus, one feasible way to determine whether a representation is induced from a subgroup H is to calculate all of the subspaces invariant under $\varphi(H)$, and analyze these to determine whether they satisfy the properties of S described above.

The Lattice Procedure

We will begin by outlining our program's procedure for calculating the lattice of invariant subspaces. Once we have described the full procedure, we will examine each step individually in greater detail. As before, suppose G is a finite group, $\mathbb{F}$ is a finite field, and $\varphi : G \to GL_n(\mathbb{F})$ is a representation.
The Lattice Procedure:

1. Using Parker's Meat-Axe algorithm, we find all of irreducible constituents of φ. This gives us a list $\varphi_1, \varphi_2, \ldots, \varphi_k$ of irreducible representations of G; we call these composition factors. Further information about the Meat-Axe can be found in our Spring report.

2. For each irreducible representation φ_i, we find a member m_i of the group ring, $\mathbb{F}[G]$, with the properties that $\varphi_i(m)$ is singular and $\varphi_j(m)$ is regular for all $j \neq i$. We refer to these elements as the "peakwords."

3. We calculate the nullspaces of $\varphi(m_1), \varphi(m_2), \ldots, \varphi(m_k)$. For each vector v in these nullspaces, we calculate the smallest subspace of F_n invariant under $\varphi(G)$ that contains v. The resulting subspaces $M_1, M_2, \ldots, M_l$ are called "mountains" and are precisely the members of the set P we described earlier. Each space M_i has the property that there exists a unique subspace V in $\mathbb{F}^n$ that is invariant under $\varphi(G)$ and maximally contains M_i.

4. We compute the containments between mountains. We also compute several relations called "dotted lines" which describe subspaces that result when we add two of the mountains derived from the same composition factor.

5. Using the basic incidences between mountains and the dotted lines, we compute the lattice of all subspaces invariant under $\varphi(G)$.

In our previous report we discussed Parker's Meat-Axe algorithm, which depended heavily upon the theory provided by Norton's Irreducibility Criterion. The procedure we consider in this paper is similarly built around the Benson-Conway theorem. This theorem tells us that if we can determine a the mountains in the lattice of spaces invariant under $\varphi(G)$, the basic containments between mountains, and the dotted lines, then we can describe the entire lattice of invariant subspaces *without having to calculate each subspace explicitly.*

Finding Peakwords and Maximal Spaces

Having already described peakwords, we will now give a formal definition: **Definition:** Let G be a group, and $\varphi : G \to GL_n(\mathbb{F})$ be a representation of G, and let φ_i denote an irreducible constituent of φ for $i = 1, 2, \ldots, k$. Extend $\varphi : \mathbb{F}[G] \to M_{n\times n}(\mathbb{F})$ to the group-ring $\mathbb{F}[G]$. We say $w \in \mathbb{F}[G]$ is a *peakword* if there exists an i such that $\varphi_i(w)$ is singular and for all $j \neq i$, $\varphi_j(w)$ is regular.

Finding a peakword for each irreducible constituent is a not a procedure that is very well understood. Practically speaking, one generally creates a "word-generator" program that produces elements of $\mathbb{F}[G]$, and then tests each resulting word to see if it possesses the peakword property. We have used two such word-generators one modeled after the deterministic word-generator used in the original C Meat-Axe, the other modeled after the probabilistic product-replacement procedure used to choose an element randomly from a finite group.

From the paper of Lux, Müller, and Ringe, we take the following facts:

- A peakword exists for each irreducible constituent of a reducible representation.

- When a peakword is evaluated on the original representation, the resulting matrix M has a nontrivial nullspace. Each vector v in the nullspace has the property that the smallest space invariant under $\varphi(G)$ that contains v is a mountain.

As described in Step 2 of the procedure, once we have obtained the peakwords, we can evaluate each peakword over the original representation to obtain a matrix with a nontrivial nullspace. For each vector v in that nullspace, we calculate the smallest subspace of $\mathbb{F}^n$ invariant under $\varphi(G)$ that contains v. The resulting set of spaces is P, the mountains of φ.

Having calculated all of the mountains of the representation, we determine whether there exists a containment relationship between each pair of mountains, and record this information in a table of basic incidences.

Finding Dotted Lines

To make use of the Benson-Conway theorem, we also need additional information about the sums of mountains. Specifically, we must determine an equivalence relation among mountains derived from the same irreducible representation. We say that a set of mountains $\{M_1, M_2, \ldots, M_d\}$ form an equivalence class provided that no mountain in the set contains another mountain in the set, and the sum of any two mountains in the set contains every element in the class. We refer to the resulting equivalence class as a dotted line.

Calculating the Lattice

Using the Benson-Conway theorem (in the form described in the paper of Lux, Müller and Ringe), we can represent each invariant subspace by the mountains it contains. The table of basic incidences gives us information about how the mountains are related in terms of containment, while the dotted lines gives us information about the spaces contained by the sums of mountains.

Consequently, given a set of mountains S, we can use the incidence table to verify that for each mountain $M \in S$, all of the mountains M contains are also in S. Likewise, for each dotted line D, we can check that whenever two members of D are in S, all members of D are in S. If S passes both of these tests, we conclude S describes a subspace invariant under $\varphi(G)$. Since each representation has finitely many mountains, it is not difficult to test all possible sets of mountains to find all of the invariant subspaces.

Refining the Lattice Algorithm

Algorithmically speaking, we have two ways of enhancing our implementation of the lattice procedure. First, there is great flexibility in the way one chooses elements from the group ring to test as peakwords. We originally used a non-deterministic procedure provided by the `chop` package. However, we found this method too slow for our purposes, and switched to a deterministic system similar to the one implemented in the C Meat-Axe. Unfortunately, there is no known way to show that a procedure for selecting elements from the group ring is the "most optimal" despite the fact that one's choice plays a significant role in the runtime of the program. Consequently, this is one area of the program users may need to tailor to suit their own purposes.

A second optimization that we intend to implement is peakword-condensation. We will briefly sketch the procedure here. Recall that each dotted line is a relation among mountains derived from the same peakword, and that each peakword is derived from a particular constituent. The same procedure we used to calculate mountains for the original representation can be used to find mountains of a smaller representation for the so-called "condensed algebra." The mountains of the condensed algebra correspond explicitly to mountains of the original representation. However, the new mountains are considerably smaller than the corresponding mountains of the original representation. As a result, one can more efficiently compute dotted lines for the original representation by computing dotted lines for the "condensed algebra" and then translating them into relations among mountains of the original representation.

Peak-word condensation is the largest remaining feature for us to implement in our refined Lattice package. Additionally, there are several small tasks which must be completed. Specifically, we would like to systematically profile our program (so that we can understand how much time each step of the program requires), create a manual, and format the scripts we have accumulated into a cohesive GAP package. As an application of our package, we would like to create scripts for computing self-dual codes.

References:

K. Lux and M. Wiegelmann, Condensing Tensor Product Modules, In “The ATLAS ten years on”, editors R.A. Wilson and R.T. Curtis. Cambridge University Press, 1998.

K. Lux and H. Pahlings, Computational aspects of representation theory of finite groups II, in Algorithmic Algebra and Number Theory. Matzat, B.H, Greuel, G.-M., Hiss, G., Eds. Springer, Berlin, 1999.

K. Lux and M. Wiegelmann, An Application of Condensation, The PIM Structures of the Mathieu Group M23. J. Symbolic Computation, 31:163-178,2001.

K. Lux, J. Müller, and M. Ringe, Peakword condensation and submodule lattices: An application of the Meat-axe, J. Symbolic Computation, 17:529-544, 1994.

Joseph Thomas, Computational Approaches to Finding Irreducible Representations, Undergraduate Research Assistantship Report, 16 May 2008.