Inequalities Involving Generalized Matrix Functions

by

Alexander Byaly

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
1 August 2015

Keywords: Immanant, Generalized Matrix Function, Laplacian Matrix, Brauer Character, Linear Representation, Modular Character

Approved by

Randall R. Holmes, Professor of Mathematics
Stewart L. Baldwin, Professor of Mathematics
Thomas H. Pate, Professor of Mathematics
Michel Smith, Professor of Mathematics

Abstract

Given a set F of functions that have their domain and codomain in common, if the codomain is ordered we may partially order F by choosing a subset S of their domain and saying that for $f, g \in F$, $f \leq g$ means that $f(x) \leq g(x)$ for every $x \in S$. We call such an ordering a dominance ordering of F on S. This dissertation is concerned with dominance orderings of (normalized) generalized matrix functions on the set of positive semidefinite matrices and on select subsets. We consider variants of Shur's theorem and Lieb's conjecture with different subsets of the group algebra and a different dominance order. In particular, we show that neither holds for Brauer characters with the usual ordering. We suggest a subset of the positive semidefinite matrices on which a modified Schur's theorem may hold for Brauer characters. It holds for Brauer characters of p-solvable groups and for many specific Brauer characters of symmetric groups. We provide a computational approach for generating dominance inequalities involving Brauer characters of symmetric groups.

Acknowledgments

I would like to express my appreciation for my thesis adviser, Randall R. Holmes, whose advice, criticism, patience, and encouragement were essential to the development of this work. I am also indebted to my thesis committee, Kenneth Noe, Michel Smith, Stewart L. Baldwin, and Thomas H. Pate, for keeping me honest.

I would like to thank Tin-Yau Tam, who often saved me when I got myself into trouble, as well as Carolyn Donegan and Gwen Kirk, who helped me stay out of it. I would also like to thank my colleagues, Daniel Brice, Steven X. Clontz, and Zachary Sarver, for helping me get back to work and sometimes the opposite.

I am grateful to Gordon G. Johnson, who asks a lot of questions.

Finally, I would like to thank my parents for their constant support and for the frequency with which they asked when I would be done.

List of Symbols

$\mathbb{C}G$ The group algebra of G over $\mathbb{C}$.

$\{\!\{\pi\}\!\}$ The irreducible Brauer character of S_n corresponding to the partition π.

$[\![\pi]\!]$ The irreducible ordinary character of S_n corresponding to the partition π.

α' The restriction of $\alpha \in \mathbb{C}G$ to the p-regular elements of G.

d_c The generalized matrix function corresponding to $c \in \mathbb{C}S_n$.

$\bar{d}_c$ In the event that $c(\mathrm{id}) \neq 0$, $\bar{d}_c = \frac{1}{c(\mathrm{id})} d_c$.

G' The set of p-regular elements of the group G.

$\mathrm{IBr}(G)$ The irreducible Brauer characters of G with respect to p.

$\mathrm{Irr}(G)$ The irreducible characters of the group G.

$\leq_S$ $f \leq_S g$ iff $g(\mathrm{id})f \preceq_S f(\mathrm{id})g$.

$\preceq_S$ $f \preceq_S g$ iff $d_f(A) \leq d_g(A)\ \forall A \in S$.

$\mathcal{P}_n(p)$ The set of p-regular matrices in $\mathbb{C}^{n\times n}$.

S_n The symmetric group of degree n.

Table of Contents

Abstract ii

Acknowledgments iii

List of Symbols iv

1 Introduction 1

2 Laplacian Matrices 6

3 Brauer Characters 11

4 Linear Programming 29

4.1 Setting up the linear program 31

4.2 The Decomposition Matrix 40

4.3 Producing a solution 45

4.4 Two complete examples 51

4.5 Linear programming appendix 54

5 Further Work 59

A Generated Results 63

Bibliography 65

Chapter 1

Introduction

Let H be a set. Then $\mathbb{C}H$ is the set of functions from H to $\mathbb{C}$. This forms a vector space using pointwise addition and the usual scalar multiplication. There is a natural embedding $\iota_H : H \to \mathbb{C}H$ such that $\iota_H(\sigma)$ is the indicator function on $\{\sigma\}$. When it is clear from context we will use the same symbol to refer to $\sigma \in H$ and the corresponding element $\iota_H(\sigma) \in \mathbb{C}H$. The image $\iota_H(H)$ forms a basis for $\mathbb{C}H$.

If G is a group then the group multiplication on $\iota_G(G)$ uniquely extends to a bilinear map $* : \mathbb{C}G \times \mathbb{C}G \to \mathbb{C}G$. The pair $(\mathbb{C}G, *)$ is called the **group algebra** of G over $\mathbb{C}$.

Given two functions f and g with codomain $\mathbb{C}$, we call f and g **equivalent** if they agree on the intersection of their domains and they are both zero outside the intersection.

Let f be a partial function from the symmetric group S_n to $\mathbb{C}$ (a function with codomain $\mathbb{C}$ and with domain any subset of S_n).
Define the **generalized matrix function** $d_f : \mathbb{C}^{n\times n} \to \mathbb{C}$ corresponding to f by

$$d_f(A) = \sum_{\sigma \in \operatorname{dom} f} f(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)},$$

where $A = (a_{ij}) \in \mathbb{C}^{n\times n}$.

Note that if f and g are equivalent partial functions from S_n to $\mathbb{C}$, then $d_f = d_g$. Since this dissertation is primarily about these generalized matrix functions, it will not be necessary to distinguish between equivalent partial functions. From here on we will consider $\mathbb{C}G$ to be a set of equivalence classes. Given an equivalence class f in $\mathbb{C}S_n$, d_f is the generalized matrix function corresponding to any function in the class.

Let H be a subset of G. Then $\mathbb{C}H$ is a set of equivalence classes to which functions from H to $\mathbb{C}$ belong. But every function f from H to $\mathbb{C}$ can be extended to an equivalent function $\bar{f} : G \to \mathbb{C}$ by setting $\bar{f}(x) = 0$ for $x \in G \backslash H$. So $\mathbb{C}H$ is a subset of $\mathbb{C}G$.

Given $S \subseteq \mathbb{C}^{n\times n}$, we partially order $\mathbb{C}S_n$ by putting

$$f \preceq_S g \quad \Leftrightarrow \quad d_f(A) \leq d_g(A) \ \forall A \in S.$$

If $f \in \mathbb{C}S_n$ and $f(\mathrm{id}) > 0$, we define $\bar{d}_f(A) = \frac{d_f(A)}{f(\mathrm{id})}$. We partially order $\mathbb{C}S_n$ a second way by putting

$$f \leq_S g \quad \Leftrightarrow \quad g(\mathrm{id})d_f(A) \leq f(\mathrm{id})d_g(A) \ \forall A \in S,$$

which, when $\bar{d}_f$ and $\bar{d}_g$ are both defined, is equivalent to

$$f \leq_S g \quad \Leftrightarrow \quad \bar{d}_f(A) \leq \bar{d}_g(A) \ \forall A \in S.$$

We may think of $\leq_S$ as the analogue of $\preceq_S$ on the projective space $(\mathbb{C}S_n \backslash \{0\})/\approx$, where $v \approx w$ if and only if v and w are parallel.

Theorem 1.1. *Let $f, g \in \mathbb{C}S_n$ such that $f(\mathrm{id}) > 0$ and $g(\mathrm{id}) > 0$. Then*

$$f \leq_S g \quad \Leftrightarrow \quad \frac{f}{f(\mathrm{id})} \preceq_S \frac{g}{g(\mathrm{id})}$$

Proof. This follows by the linearity of $f \mapsto d_f$. □

When $f \leq g$ is written with the subscript omitted it will mean $f \leq_{\mathcal{H}} g$, where $\mathcal{H}$ is the set of positive semidefinite matrices in $\mathbb{C}^{n\times n}$. Similarly, $f \preceq g$ means $f \preceq_{\mathcal{H}} g$.

Remark 1.2. *If $f, g \in \mathbb{C}S_n$ and $S \subseteq \mathbb{C}^{n\times n}$, the statement $f \preceq_S g$ is also an assertion that $d_f(A)$ and $d_g(A)$ are real for every $A \in S$. Furthermore, if S is nonempty the set $\{x \in \mathbb{C}S_n : \exists y \in \mathbb{C}S_n (x \preceq_S y)\}$ is not closed under scalar multplication unless we restrict*

ourselves to real scalars. For $\alpha \in \mathbb{C}$ *we have* $d_{\alpha f} = \alpha d_f$*, so if* α *is not real then the sets* $\{A \in \mathbb{C}^{n\times n} : d_f(A) \in \mathbb{R}\}$ *and* $\{A \in \mathbb{C}^{n\times n} : d_{\alpha f}(A) \in \mathbb{R}\}$ *are disjoint.*

Compare this to the set $\{x \in \mathbb{C}S_n : \exists y \in \mathbb{C}S_n(x \leq_S y)\}$*, which is all of* $\mathbb{C}S_n$*.*

Fix a group G. A **matrix representation** of G over $\mathbb{C}$ (or $\mathbb{C}$-representation of G) is a homomorphism from G into $\mathrm{GL}_m(\mathbb{C})$ for some positive integer m. If ρ is a $\mathbb{C}$-representation of G, $tr \circ \rho$ is called a $\mathbb{C}$-**character** of G. A $\mathbb{C}$-character of a group is necessarily constant on conjugacy classes, since if $\sigma, \tau, \alpha \in G$ with $\sigma = \alpha^{-1}\tau\alpha$ then for every $\mathbb{C}$-representation ρ of G we have $\rho(\sigma) = \rho(\alpha)^{-1}\rho(\tau)\rho(\alpha)$ and similar matrices have the same trace.

A $\mathbb{C}$-character that cannot be expressed as the sum of two other $\mathbb{C}$-characters is called **irreducible**. The set of irreducible $\mathbb{C}$-characters of G is abbreviated $\mathrm{Irr}\, H$. If χ is a $\mathbb{C}$-character of G then $\chi(\mathrm{id}) \geq 0$ [11, p. 8.5], so $\bar{d}_\chi$ is defined.

For a symmetric group S_n we can generate the set of irreducible characters using a construction on Young tablueaux. If π is a partition of n, a **Young diagram** of shape π is a collection of boxes arranged in rows such that the number of boxes in row i is π_i. Given a Young diagram with n boxes, assembling the integers $\{1, \ldots, n\}$ within them yields a **Young tableau**.

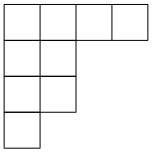


Figure 1.1: A Young diagram of shape $[4, 2, 2, 1]$.

3	9	5	7
1	8		
6	2		
4			

Figure 1.2: A Young tableau of shape $[4, 2, 2, 1]$.

Fix an integer n and let α be a Young tableau with size n. Define $R_\alpha = \{\sigma \in S_n : \forall i \in \{1, \ldots, n\}, \sigma(i)$ shares a row with i in $\alpha\}$ and $C_\alpha = \{\sigma \in S_n : \forall i \in \{1, \ldots, n\}, \sigma(i)$ shares a

column with i in $\alpha\}$. The **Young symmetrizer** $y_\alpha \in \mathbb{C}S_n$ is defined to be

$$y_\alpha = \sum_{\substack{\sigma \in R_\alpha \\ \tau \in C_\alpha}} \varepsilon(\tau)\sigma\tau,$$

where ε is the sign function. The action of S_n on $\mathbb{C}S_n y_\alpha$ forms a $\mathbb{C}$-representation of S_n and the corresponding character is irreducible. Furthermore, generating one such character for each partition of n yields the complete set of irreducible characters of S_n [3, p. 52]. We denote the irreducible character of S_n corresponding to the partition α by $[\![\alpha]\!]$.

The character $[\![1, 1, \ldots, 1]\!]$ is the sign function ε. The generalized matrix function d_ε is the determinant. The character $[\![n]\!]$ is the constant function $1 : S_n \to \mathbb{C}$ given by $(\sigma \mapsto 1)$. The corresponding function d_1 is called the permanent. It is known that among the functions $f \in \mathbb{C}S_n$ satisfying $f \succeq 0$, ε is a minimum.

Theorem 1.3 (Schur [6, thm 7.3, p. 214]). *If $G \leq S_n$ and $\chi \in \mathrm{Irr}(G)$ then $\chi \geq \varepsilon$.*

The minimality of ε is an immediate consequence of two other results regarding generalized matrix functions.

Theorem 1.4 ([6, ex 7.13, p. 219]). *If $G \leq S_n$ and $\chi \in \mathrm{Irr}(G)$ then $\chi \succeq 0$.*

Theorem 1.5 (Watkins [6, thm 7.6, p. 215]). *If $f \in \mathbb{C}S_n$ and $f \succeq 0$ then $f \geq \varepsilon$.*

Proof of Theorem 1.3. Let G be a subgroup of S_n and let $\chi \in \mathrm{Irr}(G) \subseteq \mathbb{C}S_n$. Then by Theorem 1.4 we have $\chi \succeq 0$ and therefore by Watkins' theorem we know $\chi \geq \varepsilon$. □

Since ε, which corresponds to the "smallest" partition of n, is also minimal in a different set, this suggests that the character corresponding to the "largest" partition of n may be maximal in another way.

Conjecture 1.6 (Lieb's conjecture, "Permanental Dominance conjecture") [6, p. 224]). *If $G \leq S_n$ and $\chi \in \mathrm{Irr}(G)$, then $\chi \leq 1$.*

Conjecture 1.7 (Soules' conjecture [6, p. 224])**.** *Let $G \leq S_n$ and $\chi \in \mathrm{Irr}(G)$. Let ρ be a $\mathbb{C}$-representation of G such that $\rho(\sigma)$ is unitary for every $\sigma \in G$ and $\chi = tr \circ \rho$. Let $\delta_i(\sigma)$ be the ith diagonal entry of $\rho(\sigma)$. Then*

$$\delta_i \leq 1.$$

Chapter 2

Laplacian Matrices

Let G be a simple oriented graph with vertices $\{v_1, \ldots, v_n\}$. This means the edges of G are directed, for every pair of vertices there is at most one edge connecting them, and there are no edges connecting a vertex to itself. The edge set $E(G)$ is a set of ordered pairs such that $(u, v) \in E(G)$ corresponds to the existence of an edge from u to v. The **laplacian matrix** of G is the matrix $L(G) = (a_{ij})$ where

$$a_{ij} = \begin{cases} \text{degree}(i) & \text{if } i = j, \\ -1 & \text{if } v_i \text{ and } v_j \text{ are adjacent,} \\ 0 & \text{otherwise.} \end{cases}$$

An **edge labeling** of a G is a function with domain $E(G)$. Let f be an edge labeling of G using complex numbers. The **generalized laplacian matrix** $L(G, f)$ is given by

$$L(G,f)_{ij} = \begin{cases} \sum_{e \in E(G), v_i \in e} |f(e)| & \text{if } i = j, \\ f(v_i, v_j) & \text{if } (v_i, v_j) \in E(G), \\ \overline{f(v_i, v_j)} & \text{if } (v_j, v_i) \in E(G), \\ 0 & \text{otherwise.} \end{cases}$$

Note that if f is the constant -1 function, this is the same as the laplacian matrix $L(G)$.

Theorem 2.1. *L(G,f) is positive semidefinite.*

Proof. For each $k, l \in \{1, \ldots, n\}$ with $k < l$, define the matrix M_{kl} by

$$[M_{kl}]_{ij} = \begin{cases} |L(G,f)_{kl}| & \text{if } i = j = k, \\ |L(G,f)_{kl}| & \text{if } i = j = l, \\ L(G,f)_{ij} & \text{if } i = k \text{ and } j = l, \\ L(G,f)_{ij} & \text{if } i = l \text{ and } j = k, \\ 0 & \text{otherwise.} \end{cases}$$

Then we have

$$L(G,f) = \sum_{1 \le k < l \le n} M_{kl}.$$

Each matrix M_{kl} is a direct sum of a zero matrix with a 2×2 matrix that has a non-negative diagonal and determinant zero. This means the matrices $\{M_{kl}\}$ are positive semidefinite. Since the sum of positive semidefinite matrices is positive semidefinite, this completes the proof. □

Corollary 2.2. *$L(G)$ is positive semidefinite.*

A matrix $A = (a_{ij}) \in \mathbb{C}^{n \times n}$ is called **diagonally dominant** if

$$\sum_{i \neq k} |a_{ik}| \le |a_{kk}|$$

for every $k \in \{1, \ldots, n\}$.

Theorem 2.3. *If A is a hermitian diagonally dominant matrix, then $A = D + \sum_{i \in I} L_i$, where D is a nonnegative diagonal matrix and the L_i, $i \in I$, are generalized laplacians of oriented graphs labeled with complex numbers.*

Proof. Suppose $A = (a_{ij}) \in \mathbb{C}^{n \times n}$ is hermitian and diagonally dominant. Put $I = \{(i,j) : 1 \le i < j \le n\}$.

Let $k = (i, j) \in I$. Set G_k to be the oriented graph with vertex set $V(G_k) = \{v_1, \ldots, v_n\}$ and edge set $E(G_k) = \{(v_i, v_j)\}$ (the set containing a single edge from v_i to v_j). Then the function $f_k : (v_i, v_j) \mapsto a_{ij}$ is a labeling of G_k.

The generalized laplacian matrix $L_k = L(G_k, f_k)$ is zero except possibly in four positions. The (i, i) and (j, j) entries are both $|a_{ij}|$. The (i, j) entry is a_{ij}, and the (j, i) entry is $\overline{a_{ij}} = a_{ji}$.

Thus the matrix

$$D = A - \sum_{k \in I} L_k$$

is zero off the diagonal. The ith diagonal entry is

$$\begin{aligned} d_{ii} &= a_{ii} - \sum_{k \in I} [L_k]_{ii} \\ &= a_{ii} - \sum_{1 \le j < i} \left[L_{(j,i)}\right]_{ii} - \sum_{i < j \le n} \left[L_{(i,j)}\right]_{ii} \\ &= a_{ii} - \sum_{1 \le j < i} |a_{ji}| - \sum_{i < j \le n} |a_{ij}| \\ &= a_{ii} - \sum_{j \ne i} |a_{ji}| \\ &\ge 0. \end{aligned}$$

Therefore D is a positive diagonal matrix and $A = D + \sum_{k \in I} L_k$. □

Corollary 2.4. *Hermitian diagonally dominant matrices are positive semidefinite.*

Proof. A sum of positive semidefinite matrices is positive semidefinite. □

We have identified S_n with the subset of $\mathbb{C}S_n$ consisting of the indicator functions of the singleton subsets of S_n in the natural way. So in $d_\sigma(A)$, the σ is the function that sends σ to 1 and every other element of S_n to 0 and $d_\sigma(A) = \prod_{i=1}^n a_{i\sigma(i)}$. Given any $c \in \mathbb{C}S_n$, we can then write

$$d_c(A) = \sum_{\sigma \in S_n} c(\sigma) \prod_{i=1}^n a_{i\sigma(i)} = \sum_{\sigma \in S_n} c(\sigma) d_\sigma(A).$$

Lemma 2.5. *Let $A \in \mathbb{C}^{n\times n}$ be a positive semidefinite matrix such that $\mathrm{d}_\sigma(A) \geq 0$ for every $\sigma \in S_n$. Then if $H \leq S_n$ and $\chi \in \mathrm{Irr}(H)$ we have $\mathrm{per}(A) \geq \bar{d}_\chi(A)$.*

Proof. Let $A \in \mathbb{C}^{n\times n}$ be positive semidefinite and assume that $\mathrm{d}_\sigma(A) \geq 0$ for every $\sigma \in S_n$. Let $H \leq S_n$ and let $\chi \in \mathrm{Irr}(H)$. We know that for each $\sigma \in H$, $\left|\frac{\chi(\sigma)}{\chi(e)}\right| \leq 1$ because $\chi(\sigma)$ is a sum of $\chi(e)$-many roots of 1 [11, 8.5(2) and proof, p. 18], so $|\chi(\sigma)|$ cannot be greater than $\chi(e)$.

We have

$$\begin{aligned}
\bar{d}_\chi(A) &= \frac{1}{\chi(e)} \sum_{\sigma\in \mathrm{H}} \chi(\sigma) d_\sigma(A) \\
&= \sum_{\sigma\in \mathrm{H}} \frac{\chi(\sigma)}{\chi(e)} d_\sigma(A) \\
&\leq \left| \sum_{\sigma\in \mathrm{H}} \frac{\chi(\sigma)}{\chi(e)} d_\sigma(A) \right| \\
&\leq \sum_{\sigma\in \mathrm{H}} \left| \frac{\chi(\sigma)}{\chi(e)} \right| d_\sigma(A) \\
&\leq \sum_{\sigma\in \mathrm{H}} d_\sigma(A) \\
&\leq \sum_{\sigma\in \mathrm{S}_n} d_\sigma(A) \\
&= \mathrm{per}(A).
\end{aligned}$$

□

Theorem 2.6. *Let $H \leq S_n$ and $\chi \in \mathrm{Irr}(H)$. Let B be the set of laplacian matrices of bipartite graphs with n vertices. Then*

$$\chi \leq_B 1.$$

Proof. We will show that every element of B satisfies the hypothesis of Lemma 2.5. Suppose G is a bipartite graph with vertices $\{v_1, \ldots, v_n\}$ and put $A = (a_{ij}) = L(G)$. Let $\sigma \in S_n$. We denote by $\mathrm{unfix}(\sigma)$ the set of elements of $\{v_1, \ldots, v_n\}$ that σ does not leave constant (with

the action $\sigma(v_i) = v_{\sigma(i)}$).

Then

$$d_\sigma(A) = \prod_{i\in\{1,\dots,n\}} a_{i,\sigma(i)} \neq 0 \quad \Leftrightarrow \quad \forall v \in \text{unfix}(\sigma),\ v \text{ is adjacent to } \sigma(v).$$

Suppose $d_\sigma(A) \neq 0$, and choose $v \in \text{unfix}(\sigma)$. Let J be the orbit of v under $\langle\sigma\rangle$. Either the elements of J form a cycle in G or J is a pair of adjacent vertices. Since G is bipartite, every cycle of G has an even number of elements. Thus $|J|$ is even. Let S be the set of orbits of $\text{unfix}(\sigma)$. Then we have

$$\begin{aligned} d_\sigma(A) &= \prod_{i\in\{1,\dots,n\}} a_{i,\sigma(i)} \\ &= \left[\prod_{v_i\in\text{fix}(\sigma)} a_{i,i}\right] \prod_{v_i\in\text{unfix}(\sigma)} a_{i,\sigma(i)} \\ &= \left[\prod_{v_i\in\text{fix}(\sigma)} a_{i,i}\right] \prod_{\text{J}\in\text{S}} \underbrace{\prod_{v_i\in\text{J}} a_{i,\sigma(i)}}_{(-1)^{|J|}=1}. \end{aligned}$$

Since the diagonal entries of $L(G)$ are just the degrees of the corresponding vertices, the product of the diagonal is positive. So $d_\sigma(A) > 0$. This happens for every $\sigma \in H$ for which $d_\sigma(A)$ is nonzero. The conclusion follows from Lemma 2.5. □

Chapter 3

Brauer Characters

Let $\mathbb{O}$ be the ring of algebraic integers in $\mathbb{C}$ and let p be a prime. The field $\mathbb{F}$ is constructed by choosing a maximal ideal M of $\mathbb{O}$ containing p and setting $\mathbb{F} = \mathbb{O}/M$. Denote by U the group of roots of unity of order relatively prime to p in $\mathbb{O}$, and denote by U' the group of roots of unity in $\mathbb{F}$.

Theorem 3.1. *For every $z \in U'$, the order of z is relatively prime to p.*

Proof. Let $z \in U'$. Suppose the order of z is kp for some positive integer k. Then

$$\begin{aligned} z^{kp} - 1 &= 0 \\ z^{kp} - 1^p &= 0 \\ (z^k - 1)^p &= 0 \\ z^k - 1 &= 0, \end{aligned}$$

but kp is the minimal positive integer such that $z^{kp} = 1$, so this is a contradiction. □

Theorem 3.2. *The quotient map $\pi : \mathbb{O} \to \mathbb{O}/M$, maps U isomorphically onto U'.*

Proof. First, observe that if $z \in \mathbb{O}$ is a zero of the polynomial $x^k - 1$ then $z + M$ is a zero as well. Thus $\pi(U) \subseteq U'$.

Next, we will show that $\pi|_U$ is injective. Suppose $z \in U$ with order $k > 1$, and $\pi(z) = 1$. For any integer n, $\pi(z^n) = \pi(z)^n = 1$, so $1 - z^n \in M$. Thus we have

$$k = k - 0 = k - \left[\sum_{0 \le i < k} z^i\right] = \left[\sum_{0 \le i < k} (1 - z^i)\right] \in M.$$

Since k is relatively prime to p, 1 can be written as a linear combination of k and p with integer coefficients. Because both k and p are in M, 1 is in M as well, which is a contradiction since M is a proper ideal. Therefore $\mathrm{Ker}(\pi|_U)$ is trivial.

To see why π is surjective, let k be an integer relatively prime to p and set R_k (resp. R'_k) to be the group of zeros of $x^k - 1$ in $\mathbb{O}$ (resp. in $\mathbb{O}/M$). Then π maps R_k injectively into R'_k, but since R_k has order k $\left(R_k = \{e^{\frac{2\pi i}{k}} : 0 \leq i < k\}\right)$, and R'_k has order no more than k (R'_k is a set of zeroes of a degree k polynomial), $\pi(R_k) = R'_k$. However, $U' = \bigcup_{gcd(p,k)=1} R'_k$ (by Theorem 3.1), so $\pi(U) = U'$ and this completes the proof. □

A **p-regular** member of a group G is an element of order relatively prime to p. We denote by G' the set of p-regular elements of G. Group elements that are not p-regular are called **p-singular**. A **matrix representation** of G over a field $\mathbb{F}$ is a homomorphism from G into $\mathrm{GL}_m(\mathbb{F})$ for some m.

Let $\rho : G \to \mathrm{GL}_m(\mathbb{F})$ be a matrix representation of G. If $\sigma \in G'$, then the eigenvalues of $\rho(\sigma)$ are all roots of unity (since $\rho(\sigma)^k = I_n$ for some k, every eigenvalue λ satisfies $\lambda^k = 1$).

Given ρ, define the **Brauer character** ϕ corresponding to ρ by

$$\phi : G' \to \mathbb{C},$$

$$\phi(\sigma) = \sum_{i=1}^{m} \iota(u_i),$$

where $\{u_i\}_{i=1}^m$ are the eigenvalues of $\rho(\sigma)$ counting multiplicity and ι is the inverse of $\pi|_U$. A Brauer character that cannot be written as the sum of other Brauer characters is called **irreducible**. The set of irreducible Brauer characters of G is denoted $\mathrm{IBr}\, G$. Given the above definition, it is natural to consider Brauer character analogues to Schur's theorem (1.3) and the permanental dominance conjecture (1.6).

Conjecture 3.3. *Let $G \leq S_n$ and let $\phi \in \mathrm{IBr}(G)$. Then $\phi \geq \varepsilon$.*

Conjecture 3.4. *Let $G \leq S_n$ and let $\phi \in \mathrm{IBr}(G)$. Then $\phi \leq 1$.*

However, neither of these hold. We will show this using the group S_3 and the prime 3.

	(.)	(..)	(...)
$[\![3]\!]$	1	1	1
$[\![2,1]\!]$	2	0	-1
$[\![1^3]\!]$	1	-1	1

	(.)	(..)
ϕ_1	1	1
ϕ_2	1	-1

Table 3.1: Character tables of S_3. The ordinary irreducible characters are on the left and the irreducible Brauer characters (for $p = 3$) are on the right. Since characters are constant on conjugacy classes, these tables may be used to evaluate the corresponding functions. Elements of the symmetric group are conjugate when they have the same cycle structure, so we use these structures to denote the classes.

Theorem 3.5. *Conjecture 3.3 is false.*

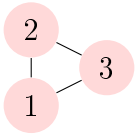
Proof. Let

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Then A is positive semidefinite, but $\det(A) = 1 - 1 - 1 - 1 + 1 + 1 = 0$ and $\bar{d}_{\phi_2}(A) = 1 - 1 - 1 - 1 + 0 + 0 = -2$. Thus $\phi_2 \not\geq \varepsilon$. □

Theorem 3.6. *Conjecture 3.4 is false.*

Proof. Let G be the following graph.



Then the laplacian matrix $L(G)$ is equal to

$$\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}.$$

According to Theorem 2.2, $L(G)$ is positive semidefinite, but $\mathrm{per}(L(G)) = 8+2+2+2-1-1 = 12$ and $\bar{d}_{\phi_1}(L(G)) = 8+2+2+2+0+0 = 14$. Thus $\phi_1 \not\leq 1$. □

Since Conjecture 3.3 and Conjecture 3.4 are false, it looks like a dominance relation on the entire set of positive semidefinite matrices is too much to ask for. However some of these inequalities may reappear if we restrict our attention to a subset.

Let p be a prime number. We define a positive semidefinite matrix $A \in \mathbb{C}^{n\times n}$ to be **p-regular** if it satisfies

$$\prod_{i=1}^{n} a_{i,\sigma(i)} = 0$$

for every p-singular $\sigma \in S_n$. We denote the set of p-regular matrices in $\mathbb{C}^{n\times n}$ by $\mathcal{P}_n(p)$. When n and p are clear from context, we will elide them and write $\mathcal{P}$ instead. We find this set of interest because if A is p-regular, then

$$d_\chi(\mathrm{A}) = d_{\chi'}(\mathrm{A}),$$

where χ' is the restriction of χ to p-regular group elements. As a consequence of this we have the following.

Theorem 3.7. *Let p be a prime and let $f, g \in \mathbb{C}S_n$ with $f \succeq g$. Then*

$$f' \succeq_{\mathcal{P}_n(p)} g'.$$

Proof. On the set $\mathcal{P}_n(p)$, the functions d_f and $d_{f'}$ are equal. □

The the statement that a matrix A is p-regular is a statement that "enough" of the off-diagonal entries of A are zero. If they are *all* zero or if there is a zero row or column the property is trivially satisfied. We observe that matrices of this nature are in fact p-regular for every prime p.

Theorem 3.8. *If p is a prime and $A \in \mathbb{C}^{n\times n}$ is a diagonal matrix with nonnegative entries then $A \in \mathcal{P}_n(p)$.*

Proof. For any p-singular $\sigma \in S_n$, we have $\sigma \neq \mathrm{id}$, so the product $\pi_{i=1}^n a_{i\sigma(i)}$ contains an off-diagonal (zero) entry of A. □

Theorem 3.9. *If p is a prime, $B \in \mathbb{C}^{(n-1)\times(n-1)}$ is positive semidefinite, and $P \in \mathbb{C}^{n\times n}$ is a permutation matrix then $A = P^{-1}(B \oplus 0)P \in \mathcal{P}_n(p)$.*

Proof. For $\sigma \in S_n$ the product $\pi_{i=1}^n a_{i\sigma(i)}$ has an entry from every row of A, one of which is a zero row. □

Addtionally, the set $\mathcal{P}_n(2)$ is somewhat degenerate. All 2-regular matrices are of one of these two types.

Theorem 3.10. *If $A \in \mathcal{P}_n(2)$ then either A is a diagonal matrix or $A = P^{-1}(B \oplus 0)P$, where B is positive semidefinite and P is a permutation matrix.*

Proof. Suppose $A \in \mathcal{P}_n(2)$ is not a diagonal matrix. Then for some $i \neq j$ there is an entry a_{ij} of A that is nonzero, and since A is Hermetian, $a_{ji} \neq 0$ as well. Put $\sigma = (ij) \in S_n$. Since σ is 2-singular, we have

$$\prod_{i=1}^{n} a_{i,\sigma(i)} = 0.$$

But this is a product of a_{ij} and a_{ji} and diagonal entries of A. Since we know a_{ij} and a_{ji} are nonzero, some diagonal entry a_{kk} must be zero.

We claim that every entry on row k or column k is zero as well. Since $A \geq 0$, for any integer $l \in 1, \ldots, n$ we have that the principal minor $a_{ll}a_{kk} - a_{kl}a_{lk} \geq 0$, so

$$-|a_{kl}|^2 = -a_{kl}a_{lk} = a_{ll}a_{kk} - a_{kl}a_{lk} \geq 0.$$

Thus there is a permutation matrix P such that $A = P^{-1}CP$, where the last column and last row of C are zero. We have $C = C(n|n) \oplus 0$, and $C(n|n)$ is positive semidefinite because it is a principal submatrix of C. □

We will now turn our attention to an analogue of Schur's conjecture involving $\leq_{\mathcal{P}}$.

Conjecture 3.11. *Let n be a positive integer and let p be a prime. If $G \leq S_n$ and $\phi \in \mathrm{IBr}(G)$, then $\phi \geq_{\mathcal{P}} \varepsilon$.*

We present some cases in which Conjecture 3.11 holds. In particular, we will see it holds if $p = 2$ or if the group G is p-solvable. (The group G is called **p-solvable** if the nonabelian composition factors of G have order relatively prime to p. In particular, this is true if the order of G is relatively prime to p.) We will eventually show that it also holds for $G = S_n$ with $n \leq 6$ and for several of the irreducible Brauer characters of larger symmetric groups.

Theorem 3.12 (Fong – Swan [1, thm 72.1])**.** *Let G be a p-solvable group. Let $\phi \in \mathrm{IBr}(G)$. There exists an irreducible character χ of G such that $\chi' = \phi$.*

Theorem 3.13. *Let G be a p-solvable group, and let $\phi \in \mathrm{IBr}(G)$. Then there is a $\chi \in Irr(G)$ such that $d_\chi(A) = d_\phi(A)$ for all p-regular matrices A.*

Proof. By the Fong – Swan theorem there exists a $\chi \in \mathrm{Irr}(G)$ with $\chi' = \phi$. If A is a p-regular matrix, $d_\chi(A) = d_{\chi'}(A) = d_\phi(A)$. □

Corollary 3.14. *If G is a p-solvable group and $\phi \in \mathrm{IBr}(G)$ then $\phi \geq_{\mathcal{P}} \varepsilon$.*

Proof. Let G be a p-solvable group and let $\phi \in \mathrm{IBr}(G)$. By Theorem 3.12, $\phi = \chi'$ for some $\chi \in \mathrm{Irr}(G)$. Then for $A \in \mathcal{P}$ we have $\bar{d}_\phi(A) = \bar{d}_{\chi'}(A) = \bar{d}_\chi(A) \geq \det(A)$, where the last step is from Theorem 1.3. □

So we see that for a p-solvable group G, the partial order $(\mathrm{IBr}(G), \leq_{\mathcal{P}})$ is order isomorphic to a subset of $(\mathrm{Irr}(G), \leq)$.

Given a Hermetian $A \in \mathbb{C}^{n\times n}$, define $G(A)$ to be the simple graph with vertex set $\{v_1, v_2, \ldots, v_n\}$ and edge set $\{\{v_i, v_j\} : a_{ij} \neq 0\}$.

Theorem 3.15. *Suppose $A \in \mathbb{C}^{n\times n}$ and $p \geq 3$ is a prime. The following are equivalent:*

1. *A is p-regular.*

2. *$G(A)$ has no cycles of length divisible by p and $A \geq 0$.*

3. *There is a simple graph G with vertex set $V(G) = \{v_1, \ldots, v_n\}$ that has no cycles of length divisible by p and $A = W^*W$ for some (possibly non-square) complex matrix $W = [w_1|w_2|\ldots|w_n]$ with the vectors w_i and w_j orthogonal exactly when v_i and v_j are not adjacent in G.*

Proof. $(1 \Rightarrow 2)$: (contrapositive) Suppose A is positive semidefinite and suppose $G(A)$ has a cycle c with length k divisible by p. Then $c = (v_{f(1)}, v_{f(2)}, \ldots, v_{f(k)})$ for some (injective) function $f : \{1, \ldots, k\} \to \{1, \ldots, n\}$. Let σ be the cycle $(f(1), f(2), \ldots, f(k))$ in S_n. This is a p-singular permutation, but the factors in the product

$$\prod_{i=1}^{n} a_{i,\sigma(i)} = \left[\prod_{i=1}^{k} a_{f(i),f(i+1 \bmod k)}\right] \prod_{i \notin \mathrm{im}(f)} a_{ii}$$

are either diagonal entries of A or entries of A corresponding to edges in $G(A)$. All of these are nonzero, so the product is nonzero and A is not p-regular.

$(2 \Rightarrow 3)$: Assume (2). Since $A \geq 0$, there is a matrix $W \in C^{n \times n}$ such that $A = W^*W$. Let $\{w_1, \ldots, w_n\}$ be the columns of W, and let $G = G(A)$.

$(3 \Rightarrow 1)$: Assume (3). Let $\sigma \in S_n$ be p-singular, and decompose σ into disjoint cycles. Since the degree of σ is the least common multiple of the cycle lengths, and since p divides the degree of σ, p must divide one of the cycle lengths. Let d be a component cycle with length divisible by p. Now, the vertices in G indexed by the integers of d do not form a cycle. Therefore there is an integer k such that v_k is not adjacent to $v_{d(k)} = v_{\sigma(k)}$. So $w_k \perp w_{\sigma(k)}$. This means that

$$a_{k,\sigma(k)} = \sum_{i=1}^{m} [W^*]_{ki} W_{i\sigma(k)} = \sum_{i=1}^{m} \overline{W_{ik}} W_{i\sigma(k)} = \langle w_{\sigma(k)}, w_k \rangle = 0,$$

so

$$\prod_{i=1}^{n} a_{i,\sigma(i)} = 0,$$

and A is a p-regular matrix. $\square$

Theorem 3.16. *Let G be a simple graph with vertices $v_1, \ldots, v_n$. Then there exists a basis $x_1, \ldots, x_n$ of $\mathbb{C}^n$ such that $E(G) = \{\{v_i, v_j\} : x_i \not\perp x_j, i \neq j\}$.*

Proof. We prove this by well ordering. Suppose there exists a graph for which this theorem is false, and the graph G is a counterexample with the number of vertices n minimal. Then G cannot be the empty graph. Let H be the graph induced by removing the vertex v_n from G. There exists a basis $X = \{x_1, \ldots, x_{n-1}\}$ of $\mathbb{C}^{n-1}$ such that $E(H) = \{\{v_i, v_j\} : x_i \not\perp x_j, i \neq j\}$. Put $B = \{v : \{v, v_n\} \in E(G)\}$ and $A = X \backslash B$. We will produce a vector $x \in \mathbb{C}^n$ such that $X \cup \{x\}$ is a basis for $\mathbb{C}^n$, $x \perp v$ for all $v \in A$, and $x \not\perp v$ for all $v \in B$.

Let m be the Lebesgue measure on $A^\perp$. We know that $X^\perp = (A \cup B)^\perp$ is a one dimensional subspace of $A^\perp$, so $\langle A \cup B \rangle \cap A^\perp$ has codimension one as a subspace of $A^\perp$. Thus $m(\langle X \rangle \cap A^\perp) = 0$.

Let $b \in B$. Consider the dimension of $b^\perp \cap A^\perp$. Clearly it is no greater than $\dim A^\perp$, and if they were equal we would have

$$b^\perp \cap A^\perp = A^\perp,$$
$$b^\perp \supseteq A^\perp,$$
$$(b^\perp)^\perp \subseteq (A^\perp)^\perp,$$
$$\langle b \rangle \subseteq \langle A \rangle.$$

However, $\langle B \rangle \cap \langle A \rangle = \emptyset$, so this is not the case. Therefore $\dim(b^\perp \cap A^\perp) < \dim(A^\perp)$ and $m(b^\perp \cap A^\perp) = 0$. This gives us that $S = \left(\langle X \rangle \cup \bigcup_{b \in B} b^\perp\right) \cap A^\perp$ is a finite union of measure zero sets. It cannot be all of $A^\perp$, so there is a vector $x \in A^\perp \backslash S$.

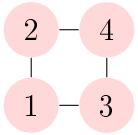
Since $x \in A^\perp$ we know that $x \perp v$ for every $v \in A$. Also $x \notin S$, which contains every vector of $A^\perp$ that is orthogonal to an element of B or is in the span of X. We have that $X \cup \{x\}$ is a basis of $\mathbb{C}^n$ with the desired property. $\square$

Using Theorem 3.15 and Theorem 3.16, we may now describe a procedure for creating p-regular matrices.

- Pick a graph G with n vertices and no cycles of length divisible by p.
- Pick vectors $v_1, \ldots, v_n \in \mathbb{C}^n$ such that v_i and v_j are orthogonal exactly when i and j are not adjacent in G.
- Set $V = [v_1|v_2|\ldots|v_n]$ and $A = V^*V$.

The matrix A is p-regular by Theorem 3.15.

Example for $p = 3$ and $n = 4$:



$$v_1 = \begin{pmatrix} i \\ 2 \\ 0 \\ 0 \end{pmatrix}, v_2 = \begin{pmatrix} 3 \\ 0 \\ 2 \\ 0 \end{pmatrix}, v_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, v_4 = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 3 \end{pmatrix}$$

$$A = \begin{pmatrix} 5 & -3i & 2 & 0 \\ 3i & 13 & 0 & 4 \\ 2 & 0 & 2 & 3 \\ 0 & 4 & 3 & 13 \end{pmatrix}$$

If we specialize Conjecture 3.11 to the prime 2, we see that it holds in a trivial way.

Theorem 3.17. *Let $\mathcal{P} = \mathcal{P}_n(2)$ for some positive integer n and let $f, g \in \mathbb{C}S_n$. Then*

$$f \leq_{\mathcal{P}} g.$$

Proof. By Theorem 3.10, $\mathcal{P}_n(2)$ is a set consisting of diagonal matrices and matrices with a zero row and column.

For a diagonal matrix D,

$$d_f(D) = \sum_{\sigma \in S_n} f(\sigma) \prod_{i=1}^{n} d_{i\sigma(i)} = f(\mathrm{id}) \prod_{i=1}^{n} d_{ii},$$

so we have

$$g(\mathrm{id}) d_f(D) = g(\mathrm{id}) f(\mathrm{id}) \prod_{i=1}^{n} d_{ii} = f(\mathrm{id}) d_g(D).$$

For a matrix C with a zero row or column, $d_f(C) = 0$, so once again $g(\mathrm{id}) d_f(C) = f(\mathrm{id}) d_g(C)$. Thus we have $f \leq_{\mathcal{P}} g$ as desired. □

Recall the role of Watkins' theorem (1.5) in the proof of Schur's theorem (1.3). We suspect a similar approach may work to prove Conjecture 3.11, but both the hypothesis and conclusion of Watkins' theorem are too strong for our needs. This leads us to the following alternative.

Theorem 3.18. *Let $c \in \mathbb{C}S_n$ such that $c \succeq_{\mathcal{P}} 0$. Then $c \geq_{\mathcal{P}} \varepsilon$.*

We will return to prove this after some technical lemmas.

Theorem 3.19. *If $A \in \mathcal{P}_n(p)$, $x \in \mathbb{R}$, and $A + xE_{nn} \geq 0$, then $A + xE_{nn} \in \mathcal{P}_n(p)$.*

Proof. Let $A \in \mathcal{P}_n(p)$, $x \in \mathbb{R}$ and assume that $A + xE_{nn} \geq 0$. Then $G(A)$ has no cycles of length divisible by p by Theorem 3.15. Since $G(A+xE_{nn}) = G(A)$, we have that $G(A+xE_{nn})$ also has no cycles of length divisible by p. With $A + xE_{nn} \geq 0$ it follows from Theorem 3.15 that $A + xE_{nn} \in \mathcal{P}_n(p)$. □

Lemma 3.20. $d_c(A + xE_{nn}) = d_c(A) + x d_c(A(n|n) \oplus 1)$.

Proof. Let $W = \{\sigma \in S_n : \sigma(n) \neq n\}$ and let $W' = S_n \backslash W$. We have

$$\begin{aligned} d_c(A + xE_{nn}) &= \sum_{\sigma \in S_n} c(\sigma) d_\sigma(A + xE_{nn}) \\ &= \sum_{\sigma \in W} c(\sigma) d_\sigma(A + xE_{nn}) + \sum_{\sigma \in W'} c(\sigma) d_\sigma(A + xE_{nn}). \end{aligned}$$

Now

$$\sum_{\sigma\in W} c(\sigma)d_\sigma(A+xE_{nn}) = \sum_{\sigma\in W} c(\sigma)d_\sigma(A),$$

while

$$\begin{aligned}
\sum_{\sigma\in W'} c(\sigma)d_\sigma(A+xE_{nn}) &= \sum_{\sigma\in W'} c(\sigma)\left(\prod_{i\neq n} a_{i\sigma(i)}\right)(a_{nn}+x) \\
&= \left[\sum_{\sigma\in W'} c(\sigma)\left(\prod_{i\neq n} a_{i\sigma(i)}\right)a_{nn}\right] + x\sum_{\sigma\in W'} c(\sigma)\left(\prod_{i\neq n} a_{i\sigma(i)}\right) \\
&= \left[\sum_{\sigma\in W'} c(\sigma)d_\sigma(A)\right] + x\sum_{\sigma\in W'} c(\sigma)d_\sigma(A(n|n)\oplus 1) \\
&= \left[\sum_{\sigma\in W'} c(\sigma)d_\sigma(A)\right] + xd_c(A(n|n)\oplus 1).
\end{aligned}$$

Therefore

$$\begin{aligned}
d_c(A+xE_{nn}) &= \left[\sum_{\sigma\in S_n} c(\sigma)d_\sigma(A)\right] + xd_c(A(n|n)\oplus 1) \\
&= d_c(A) + xd_c(A(n|n)\oplus 1).
\end{aligned}$$

□

Proof of Theorem 3.18. We want to show that $d_c(A) \geq c(e)\det(A)$ for every $A \in \mathcal{P} = \mathcal{P}_n(p)$. We proceed by induction on n. For $n = 1$, det is the identity function, so for $A = [a] \in \mathcal{P}$, we have

$$d_c(A) = c(e)a = c(e)\det(A).$$

Now assume $n > 1$. Fix $A \in \mathcal{P}$. If A is singular then $d_c(A) \geq 0 = c(e)\det(A)$ as desired. Otherwise, put

$$\begin{aligned} f(x) &= d_c(A + xE_{nn}) - c(e)\det(A + xE_{nn}) \\ &= d_c(A) - c(e)\det(A) + x\left[d_c(A(n|n) \oplus 1) - c(e)\det(A(n|n) \oplus 1)\right]. \end{aligned}$$

Now our goal is to prove that $f(0) \geq 0$. Put $r = \frac{-\det(A)}{\det(A(n|n))}$. Then $\det(A + rE_{nn}) = \det(A) + r(\det(A(n|n) \oplus 1) = 0$, and $A + rE_{nn}$ is still positive semidefinite since the principal minors that do not involve the nth row are unchanged. By Theorem 3.19, $A + rE_{nn} \in \mathcal{P}$, so $f(r) = d_c(A + rE_{nn}) \geq 0$. Since $r \leq 0$, showing that the slope of f is nonnegative would be sufficient to show $f(0)$ is as well.

The slope is $d_c(A(n|n) \oplus 1) - c(e)\det(A(n|n) \oplus 1)$. By identifying S_{n-1} with the elements of S_n that fix the last row and column of A we may define c' to be the restriction of c to S_{n-1}. Then $d_{c'}(A(n|n)) - c'(e)\det(A(n|n)) \geq 0$ by the induction hypothesis. However, the left-hand side of this inequality is equal to the slope of f, completing the proof. □

The **decomposition matrix** of the group G with respect to the prime p is the matrix representation of the restriction map from $\mathrm{span}(\mathrm{Irr}(G))$ to $\mathrm{span}(\mathrm{IBr}(G))$ (with respect to the

$S_5, p = 3$	$\{\!\{5\}\!\}$	$\{\!\{4,1\}\!\}$	$\{\!\{3,2\}\!\}$	$\{\!\{3,1^2\}\!\}$	$\{\!\{2^2,1\}\!\}$
$[\![5]\!]$	1	0	0	0	0
$[\![4,1]\!]$	0	1	0	0	0
$[\![3,2]\!]$	0	1	1	0	0
$[\![3,1^2]\!]$	0	0	0	1	0
$[\![2^2,1]\!]$	1	0	0	0	1
$[\![2,1^3]\!]$	0	0	0	0	1
$[\![1^5]\!]$	0	0	1	0	0

Table 3.2: The decomposition matrix for S_5 relative to the prime 3.

bases $\mathrm{Irr}(G)$ and $\mathrm{IBr}(G)$). A rather large collection of decomposition matrices is contained in the computer algebra system GAP [10].

Each row of the decomposition matrix tells us how to express the restriction of an ordinary character as a sum of Brauer characters. More precisely, if $D = (d_{\chi\phi})$ is the decompositiom matrix, then for each $\chi \in \mathrm{Irr}(G)$ we have $\chi' = \sum_{\phi \in \mathrm{IBr}(G)} d_{\chi\phi}\phi$, where χ' is the restriction of χ to p-regular elements of G [2, p. 267].

We call the partitions of n that do not contain p copies of the same integer p-**regular**. In an approach similar to [3] we will use the decomposition matrix to assign p-regular partitions as labels to the elements of $\mathrm{IBr}(S_n)$. Recall that there is a one-to-one correspondence between the set of partitions α of n and the irreducible characters $[\![\alpha]\!] \in \mathrm{Irr}(S_n)$. The row of the decomposition matrix corresponding to the irreducible character $[\![\alpha]\!]$ is called row α.

Theorem 3.21 ([3, p. 282])**.** *Arrange the rows of the decomposition matrix in lexicographic order, and let α be a p-regular partition of n. Then there is a column of the decomposition matrix whose first nonzero entry is on row α, and this entry is a 1. Additionally, given a p-regular partition β, the entry on row β of this column is nonzero only if α majorizes β.*

Theorem 3.21 gives us a way to assign a Brauer character to each p-regular partition α of n. For every such partition we call the column corresponding to row α in Theorem 3.21 column α and denote the associated Brauer character by $\{\!\{\alpha\}\!\}$. Since the number of p-regular partitions of n is equal to $|\mathrm{IBr}(S_n)|$ [3, p. 285], this assigns a label to every irreducible Brauer character of S_n.

Observe that the decomposition matrix in Table 3.2 is a row permutation of a matrix of the form $\left[\frac{I}{X}\right]$, where I is the identity matrix. This means that each irreducible Brauer character is the image of an ordinary irreducible character under the restriction map.

Theorem 3.22. *Let $p = 3$. If $n \leq 5$ and ϕ is an irreducible Brauer character of S_n, then*

$$\phi \geq_{\mathcal{P}} \varepsilon.$$

$S_6, p=3$	$\{\!\{6\}\!\}$	$\{\!\{5,1\}\!\}$	$\{\!\{4,2\}\!\}$	$\{\!\{3^2\}\!\}$	$\{\!\{4,1^2\}\!\}$	$\{\!\{3,2,1\}\!\}$	$\{\!\{2^2,1^2\}\!\}$
$[\![6]\!]$	1	0	0	0	0	0	0
$[\![5,1]\!]$	1	1	0	0	0	0	0
$[\![4,2]\!]$	0	0	1	0	0	0	0
$[\![3^2]\!]$	0	1	0	1	0	0	0
$[\![4,1^2]\!]$	0	1	0	0	1	0	0
$[\![3,2,1]\!]$	1	1	0	1	1	1	0
$[\![2^2,1^2]\!]$	0	0	0	0	0	0	1
$[\![2^3]\!]$	1	0	0	0	0	1	0
$[\![3,1^3]\!]$	0	0	0	0	1	1	0
$[\![2,1^4]\!]$	0	0	0	1	0	1	0
$[\![1^6]\!]$	0	0	0	1	0	0	0

Table 3.3: The decomposition matrix for S_6 relative to the prime 3.

Proof. By inspecting the decomposition matrices for S_n and the prime 3, we can see that all of the Brauer characters are just restrictions of ordinary characters of S_n. For $\chi \in \mathrm{Irr}(S_n)$ and $A \in \mathcal{P}$, we have

$$\bar{d}_{\chi'}(A) = \bar{d}_{\chi}(A) \geq \bar{d}_{\varepsilon}(A),$$

where the inequality is from Theorem 1.3. □

For S_6 and the prime 3 there exist irreducible Brauer characters that are not restrictions of ordinary irreducibles. By inspecting the decomposition matrix (Table 3.3) we may see that they are $\{\!\{5,1\}\!\}$, $\{\!\{4,1^2\}\!\}$, and $\{\!\{3,2,1\}\!\}$.

However, $\{\!\{3,2,1\}\!\}$ is a difference of restricted ordinary characters:

$$\{\!\{3,2,1\}\!\} = [\![2,1^4]\!]' - [\![1^6]\!]'.$$

We will be able to use this fact to "push" inequalities of ordinary irreducible characters through the decomposition map to generate an inequality involving $\{\!\{3,2,1\}\!\}$.

Theorem 3.23 (Pate's Theorem [6, p225]). *Suppose* $\pi = [\pi_1, \pi_2, \ldots, \pi_t]$ *and*

$$\rho = [\pi_1, \pi_2, \ldots, \pi_{s-1}, \pi_s - 1, \pi_{s+1}, \ldots, \pi_t, 1]$$

are partitions of n. *Then* $\pi \geq \rho$.

Corollary 3.24.

$$\{\!\{3, 2, 1\}\!\} \geq_{\mathcal{P}_6(3)} \varepsilon.$$

Proof. From Pate's theorem, we know $[\![2, 1^4]\!] \geq [\![1^6]\!]$, so

$$\frac{[\![2, 1^4]\!]}{[\![2, 1^4]\!](\mathrm{id})} \succeq \frac{[\![1^6]\!]}{[\![1^6]\!](\mathrm{id})}.$$

Using the hook formula (see Theorem 4.5 and explanation), we compute the degrees of the characters to be $[\![2, 1^4]\!](\mathrm{id}) = 5$ and $[\![1^6]\!](\mathrm{id}) = 1$. We have

$$[\![2, 1^4]\!] - 5[\![1^6]\!] \succeq 0,$$

$$[\![2, 1^4]\!]' - 5[\![1^6]\!]' \succeq_{\mathcal{P}} 0,$$

$$\{\!\{3, 2, 1\}\!\} = [\![2, 1^4]\!]' - [\![1^6]\!]' \succeq_{\mathcal{P}} [\![2, 1^4]\!]' - 5[\![1^6]\!]' \succeq_{\mathcal{P}} 0.$$

Since $\{\!\{3, 2, 1\}\!\}$ is nonnegative on p-regular matrices, the conclusion follows from Theorem 3.18. □

There are two other Brauer characters of S_6 with respect to the prime 3 that are not restrictions of ordinary irreducible characters. We can prove a similar inequality for them using the same strategy. The application to $\{\!\{4, 1^2\}\!\}$ is pretty straightforward, but $\{\!\{5, 1\}\!\}$ will be more involved.

Corollary 3.25.

$$\{\!\{4, 1^2\}\!\} \geq_{\mathcal{P}} \varepsilon.$$

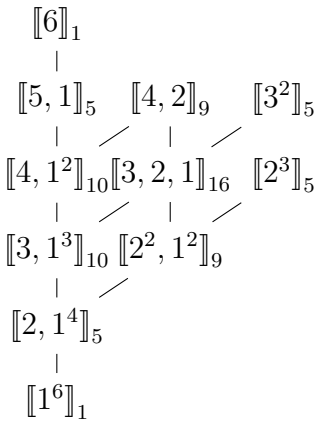


Figure 3.1: Pate's theorem on S_6. The subscripts are the character degrees.

Proof. Using Table 3.3, we can see that $\{\!\{4,1^2\}\!\} = [\![3,1^3]\!]' + [\![1^6]\!]' - [\![2,1^4]\!]'$. Pate's theorem gives us $[\![3,1^3]\!] \geq [\![2,1^4]\!]$, which implies $[\![3,1^3]\!] \succeq 2[\![2,1^4]\!]$, and hence $[\![3,1^3]\!]' - 2[\![2,1^4]\!]' \succeq_{\mathcal{P}} 0$. Using this, we can split $\{\!\{4,1^2\}\!\}$ into a sum of three parts, each of which is nonnegative on matrices in $\mathcal{P}$.

$$\{\!\{4,1^2\}\!\} = \frac{1}{2}[\![3,1^3]\!]' + [\![1^6]\!]' + \frac{1}{2}\left([\![3,1^3]\!]' - 2[\![2,1^4]\!]'\right).$$

Again, by Theorem 3.18, $\{\!\{4,1^2\}\!\} \geq_{\mathcal{P}} \varepsilon$. □

This approach so far can be summarized as follows:

1. Assemble a pool of inequalities of ordinary characters.
2. Find the corresponding inequalities of Brauer characters.
3. Find a way to compose those inequalities to prove the desired Brauer character ϕ satisfies $\phi \succeq_{\mathcal{P}} 0$.
4. Conclude from Theorem 3.18 that $\phi \geq_{\mathcal{P}} \varepsilon$.

The group S_6 has 11 ordinary irreducible characters, and Figure 3.1 shows the partial order revealed by Pate's theorem. For each of the 12 edges we have a $\geq$-inequality of irreducible characters and a corresponding $\succeq$-inequality. This, via the decomposition map, yields an inequality of Brauer characters.

$\geq$	$\succeq$		$\succeq_{\mathcal{P}}$
$[\![6]\!] \geq [\![5,1]\!]$	$[\![6]\!] \succeq \frac{1}{5}[\![5,1]\!]$		$\{\!\{6\}\!\} \succeq_{\mathcal{P}} \frac{1}{5}[\{\!\{6\}\!\} + \{\!\{5,1\}\!\}]$
$[\![5,1]\!] \geq [\![4,1^2]\!]$	$\frac{1}{5}[\![5,1]\!] \succeq \frac{1}{10}[\![4,1^2]\!]$		$\frac{1}{5}[\{\!\{6\}\!\} + \{\!\{5,1\}\!\}] \succeq_{\mathcal{P}} \frac{1}{10}[\{\!\{5,1\}\!\} + \{\!\{4,1^2\}\!\}]$
$[\![4,2]\!] \geq [\![3,2,1]\!]$	$\frac{1}{9}[\![4,2]\!] \succeq \frac{1}{16}[\![3,2,1]\!]$		$\frac{1}{9}\{\!\{4,2\}\!\} \succeq_{\mathcal{P}} \frac{1}{16}[\{\!\{6\}\!\} + \{\!\{5,1\}\!\} + \{\!\{3^2\}\!\} + \{\!\{4,1^2\}\!\} + \{\!\{3,2,1\}\!\}]$
$[\![4,2]\!] \geq [\![4,1^2]\!]$	$\frac{1}{9}[\![4,2]\!] \succeq \frac{1}{10}[\![4,1^2]\!]$		$\frac{1}{9}\{\!\{4,2\}\!\} \succeq_{\mathcal{P}} \frac{1}{10}[\{\!\{5,1\}\!\} + \{\!\{4,1^2\}\!\}]$
$[\![3^2]\!] \geq [\![3,2,1]\!]$	$\frac{1}{5}[\![3^2]\!] \succeq \frac{1}{16}[\![3,2,1]\!]$	(1)	$\frac{1}{5}[\{\!\{5,1\}\!\} + \{\!\{3^2\}\!\}] \succeq_{\mathcal{P}} \frac{1}{16}[\{\!\{6\}\!\} + \{\!\{5,1\}\!\} + \{\!\{3^2\}\!\} + \{\!\{4,1^2\}\!\} + \{\!\{3,2,1\}\!\}]$
$[\![4,1^2]\!] \geq [\![3,1^3]\!]$	$\frac{1}{10}[\![4,1^2]\!] \succeq \frac{1}{10}[\![3,1^3]\!]$		$\frac{1}{10}[\{\!\{5,1\}\!\} + \{\!\{4,1^2\}\!\}] \succeq_{\mathcal{P}} \frac{1}{10}[\{\!\{4,1^2\}\!\} + \{\!\{3,2,1\}\!\}]$
$[\![3,2,1]\!] \geq [\![2^2,1^2]\!]$	$\frac{1}{16}[\![3,2,1]\!] \succeq \frac{1}{9}[\![2^2,1^2]\!]$	(2)	$\frac{1}{16}[\{\!\{6\}\!\} + \{\!\{5,1\}\!\} + \{\!\{3^2\}\!\} + \{\!\{4,1^2\}\!\} + \{\!\{3,2,1\}\!\}] \succeq_{\mathcal{P}} \frac{1}{9}\{\!\{2^2,1^2\}\!\}$
$[\![3,2,1]\!] \geq [\![3,1^3]\!]$	$\frac{1}{16}[\![3,2,1]\!] \succeq \frac{1}{10}[\![3,1^3]\!]$		$\frac{1}{16}[\{\!\{6\}\!\} + \{\!\{5,1\}\!\} + \{\!\{3^2\}\!\} + \{\!\{4,1^2\}\!\} + \{\!\{3,2,1\}\!\}] \succeq_{\mathcal{P}} \frac{1}{10}[\{\!\{4,1^2\}\!\} + \{\!\{3,2,1\}\!\}]$
$[\![2^2,1^2]\!] \geq [\![2,1^4]\!]$	$\frac{1}{9}[\![2^2,1^2]\!] \succeq \frac{1}{5}[\![2,1^4]\!]$	(3)	$\frac{1}{9}\{\!\{2^2,1^2\}\!\} \succeq_{\mathcal{P}} \frac{1}{5}[\{\!\{3^2\}\!\} + \{\!\{3,2,1\}\!\}]$
$[\![2^3]\!] \geq [\![2^2,1^2]\!]$	$\frac{1}{5}[\![2^3]\!] \succeq \frac{1}{9}[\![2^2,1^2]\!]$		$\frac{1}{5}[\{\!\{6\}\!\} + \{\!\{3,2,1\}\!\}] \succeq_{\mathcal{P}} \frac{1}{9}\{\!\{2^2,1^2\}\!\}$
$[\![3,1^3]\!] \geq [\![2,1^4]\!]$	$\frac{1}{10}[\![3,1^3]\!] \succeq \frac{1}{5}[\![2,1^4]\!]$		$\frac{1}{10}[\{\!\{4,1^2\}\!\} + \{\!\{3,2,1\}\!\}] \succeq_{\mathcal{P}} \frac{1}{5}[\{\!\{3^2\}\!\} + \{\!\{3,2,1\}\!\}]$
$[\![2,1^4]\!] \geq [\![1^6]\!]$	$\frac{1}{5}[\![2,1^4]\!] \succeq [\![1^6]\!]$	(4)	$\frac{1}{5}[\{\!\{3^2\}\!\} + \{\!\{3,2,1\}\!\}] \succeq_{\mathcal{P}} \{\!\{3^2\}\!\}$

Table 3.4: Pate's theorem and the decomposition map.

In addition to the inequalities in Table 3.4, we have a set of 11 similar inequalities generated by Schur's theorem. For example, consider $[\![3,2,1]\!] \in \mathrm{Irr}(S_6)$. We have

$$[\![3,2,1]\!] \succeq 0,$$
$$\{\!\{6\}\!\} + \{\!\{5,1\}\!\} + \{\!\{3^2\}\!\} + \{\!\{4,1^2\}\!\} + \{\!\{3,2,1\}\!\} \succeq_{\mathcal{P}} 0. \tag{5}$$

After some exploration we eventually discover that a linear combination of the inequalities (1), (2), (3), (4), and (5) (with coefficients 5, 1, 1, 1, and $\frac{1}{4}$ respectively) yields the inequality $\{\!\{5,1\}\!\} \succeq_{\mathcal{P}} 0$ (and thus $\{\!\{5,1\}\!\} \geq_{\mathcal{P}} \varepsilon$). Since the corresponding inequalities of ordinary characters are shorter, we may use our knowledge of these coefficients to construct a less cluttered proof by postponing the application of the decomposition map.

Corollary 3.26.

$$\{\!\{5,1\}\!\} \geq_{\mathcal{P}} \varepsilon$$

Proof. For each of (1), (2), (3), and (4) we multiply the corresponding inequalities in the middle column of Table 3.4 by 5, 1, 1, and 1, respectively. We have

$$\begin{aligned}
\llbracket 3^2 \rrbracket - \frac{5}{16}\llbracket 3,2,1 \rrbracket &\succeq 0,\\
\frac{1}{16}\llbracket 3,2,1 \rrbracket - \frac{1}{9}\llbracket 2^2,1^2 \rrbracket &\succeq 0,\\
\frac{1}{9}\llbracket 2^2,1^2 \rrbracket - \frac{1}{5}\llbracket 2,1^4 \rrbracket &\succeq 0,\\
\frac{1}{5}\llbracket 2,1^4 \rrbracket - \llbracket 1^6 \rrbracket &\succeq 0,
\end{aligned}$$

and adding these inequalities up yields

$$\llbracket 3^2 \rrbracket - \llbracket 1^6 \rrbracket - \frac{1}{4}\llbracket 3,2,1 \rrbracket \succeq 0.$$

Thus,

$$\{\!\{5,1\}\!\} = \llbracket 3^2 \rrbracket' - \llbracket 1^6 \rrbracket' \succeq_{\mathcal{P}} \frac{1}{4}\llbracket 3,2,1 \rrbracket' \succeq_{\mathcal{P}} 0,$$

where the equality is from Table 3.3. We then have $\{\!\{5,1\}\!\} \geq_{\mathcal{P}} \varepsilon$ by Theorem 3.18. □

However, this proof has a noticable shortcoming. In order to produce it we had to somehow discover the "right coefficients" with which to combine the available inequalities and reach the desired conclusion, but no process for obtaining them has been provided. Unsatisfied with our dependence on sudden flashes of inspiration, we will proceed to discuss a reproducible technique by which a set of "right coefficients" can be determined.

Chapter 4

Linear Programming

The proof of Corollary 3.26 was perhaps unsatisfying in that we relied on providence to give us a suitable collection of coefficients. We would prefer to have a reproducible approach for finding them. The problem of discovering the right coefficients can be phrased as the following geometry problem. Given a collection of vectors $Q = \{q_i \mid i = 1, \ldots, k\}$ from a real vector space V, the **convex cone** of Q is the set of linear combinations of vectors in Q using non-negative coefficients. We want to be able to determine whether some vector $v \in V$ is in the convex cone of Q.

4	−1	0	0	0	0	0
2	1	0	0	−1	0	0
−9	−9	16	−9	−9	−9	0
0	−9	10	0	−9	0	0
−5	11	0	11	−5	−5	0
0	1	0	0	0	−1	0
9	9	0	9	9	9	−16
5	5	0	5	−3	−3	0
0	0	0	−9	0	−9	5
9	0	0	0	0	9	−5
0	0	0	−2	1	−1	0
0	0	0	−4	0	1	0

Table 4.1

If we fix an ordered basis B of V, determining cone membership is equivalent to finding out if the equation

$$\Big[[q_1]_B\Big|[q_2]_B\Big|\ldots\Big|[q_k]_B\Big]x = [v]_B$$

has a solution x with $x_i \geq 0$ for $i \in \{1, \ldots, k\}$.

Let us set up the problem of finding the coefficients used in Corollary 3.26 in this way. Let V be the real span of $\mathrm{IBr}(S_6)$. We fix $B = (\{\!\{6\}\!\}, \{\!\{5,1\}\!\}, \{\!\{4,2\}\!\}, \{\!\{3^2\}\!\}, \{\!\{4,1^2\}\!\}, \{\!\{3,2,1\}\!\}, \{\!\{2^2,1^2\}\!\})$ as an ordered basis for V. In the first row of Table 3.4 we have the inequality $\{\!\{6\}\!\} \succeq_{\mathcal{P}} \frac{1}{5}[\{\!\{6\}\!\} + \{\!\{5,1\}\!\}]$, which is equivalent to $4\{\!\{6\}\!\} - \{\!\{5,1\}\!\} \succeq_{\mathcal{P}} 0$. We write the left hand side as $\langle 4, -1, 0, 0, 0, 0, 0\rangle$. We generate such vectors for every row to produce Table 4.1. Also for $\chi \in \mathrm{Irr}(S_6)$, the coefficients of $\chi' \in V$ in the basis B are given in the corresponding row of the decomposition matrix. So the cone we are interested in is the one generated by the rows of Table 4.1 and the rows of the decomposition matrix. To determine

that $\{\!\{5,1\}\!\}$ is inside the cone we need to find a nonnegative vector x satisfying the following.

$$\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 4 & 2 & -9 & 0 & -5 & 0 & 9 & 5 & 0 & 9 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -9 & -9 & 11 & 1 & 9 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -9 & 0 & 11 & 0 & 9 & 5 & -9 & 0 & -2 & -4 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -9 & -9 & -5 & 0 & 9 & -3 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & -9 & 0 & -5 & -1 & 9 & -3 & -9 & 9 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -16 & 0 & 5 & -5 & 0 & 0
\end{bmatrix} x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A problem of this form is called a **linear program**, and there is a wealth of computer software capable of solving these. Given a candidate solution x we can verify it by matrix multiplication, allowing us to trust results from software. Even if the generation technique is defective, we are in the clear as long as we verify the results properly. This does raise another issue, though. Even if our software is unable to find a solution to the linear program, this does not guarantee that none exists.

Remark 4.1. *At this point one might wonder why we defined V to be the real span of $\mathrm{IBr}(S_6)$ rather than the complex span. Recall from Remark 1.2 that if $f \in \mathbb{C}S_n$, $A \in S \subseteq \mathbb{C}^{n\times n}$, and $d_f(A)$ is complex, then f is not $\preceq_S$-related to any other element of $\mathbb{C}S_n$.*

Theorem 4.2 (Farkas Lemma[7])**.** *Let A be an $m \times n$ matrix and let $b \in \mathbb{R}^m$. Then the equation $Ax = b$ has no nonnegative solution if and only if there exists a vector $y \in \mathbb{R}^m$ such that $y^T A$ has all nonnegative entries and $y^T b < 0$.*

Farkas' Lemma is a classical result of linear programming that guarantees that for problems of a particular form, there exists either a solution or a "Farkas certificate". This is an exclusive or, where the existence of one coincides with the nonexistence of the other. Thus with a candidate vector y in hand, performing the multiplication constitutes a proof that the linear program has no solution. Attacking this programmatically is now really tempting. We can both verify the solvability of solvable linear programs and verify the unsolvability of unsolvable linear programs by performing matrix multiplication, so it is not necessary for us to assert the soundness of the particular software which generated the vector we used.

However, care is still required in setting up the correct linear program and properly checking the results.

4.1 Setting up the linear program

We present an implementation of this technique. Given a positive integer n, a prime p, and a Brauer character $\phi \in \mathrm{IBr}(S_n)$, we construct a linear program L such that if L has a solution then $\phi \succeq_{\mathcal{P}_n(p)} 0$. This section is a literate Haskell program that can be built using the command `ghc ConstructLP.lhs`.

```
module ConstructLP where
import Appendix
```

We will be indexing our space of functions using the corresponding partitions of n. We begin by defining an element of the data type `Partition n p` to be the word "Partition" followed by a list of `Int`s.

```
data Partition n p = Partition [Int] deriving (Eq, Show, Read)
list :: Partition n p -> [Int]
list (Partition xs) = xs
```

Remark 4.3. *The type* `Partition` *is called a* **phantom type** *because it has type variables that do not appear in the right hand side of the definition. A value of type* `Partition A B` *is in a sense identical to a value of type* `Partition C D`*. Each of them is the word Partition followed by a list of integers. However we have specified that these are different types, so an attempt to use one in place of the other is a type error and will not compile.*

We will use the type `n` on the left hand side to specify which natural number this is a partition of. Additionally, we need to order our basis so that we may write elements of our vector space as tuples. We will be using a different ordering depending on the prime chosen, so the type `p` on the left hand side will specify the prime in use. We may extract the n and p associated with a partition through the following functions.

```
getN :: a n p -> n
getN = undefined
getP :: a n p -> p
getP = undefined
```

Remark 4.4. *Since we are representing n and p using types rather than values, it is not necessary for our implementations of* `getN` *and* `getP` *to produce a value of the associated type. Suppose we have a value* `t` *of type* `Partition N6 N3`*. We may infer that the type of* `getN t` *is* `N6`*. The function* `value :: N6 -> Integer` *ignores the value of its argument and returns* `6`*, so* `value (getN x)` *returns* `6`*. The implementation of this is in the* `Appendix` *module.*

We want to choose an ordering so that the decomposition matrix, once we have it, is as similar as possible to the tables of decomposition matrices in [3]. To implement our ordering we make `Partition n p` an instance of the `Ord` type class by defining `compare :: Partition n p -> Partition n p -> Ordering`. The function `compare` is expected to return one of `LT`, `GT`, or `EQ`, depending on whether the first argument is less than, greater than, or equal to the second one. Our ordering will make p-regular partitions least.

```
instance (Natural n, Natural p) => Ord (Partition n p) where
  compare t t' = case (isRegular t, isRegular t') of
    (True, False) -> LT
    (False, True) -> GT
```

Next, shorter partitions will be less than longer ones.

```
    _ -> case compare ((length . list) t) ((length . list) t') of
      LT -> LT
      GT -> GT
```

And lastly, if everything else is equal and α majorizes β, then $\alpha \leq \beta$.

```
      EQ -> case compare (list t) (list t') of
        GT -> LT -- If t ≥ t' in the lexicographic order on lists
```

```
    LT -> GT -- then t majorizes t'.
    EQ -> EQ
```

Given a type `n` representing a natural number n, we will enumerate the partitions of n recursively, using partitions of 1 as the base case.

```
partitions :: (Natural n) => n -> [Partition n p]
partitions n = (fmap Partition . partitions' . value) n
--The function value produces the integer associated with the type-level
--natural number n.
  where
  partitions' :: Int -> [[Int]]
  partitions' 1 = [[1]]
```

Let P_k denote the set of partitions of k. For $k \geq 2$, define $f_k : P_k \to P_{k-1}$ to be the function that decrements the last integer in the partition by 1. Note that f_k is not injective. In particular, if $\alpha = (a_1, \ldots, a_{r-1}, a_r)$ and $\beta = (a_1, \ldots, a_{r-1}, a_r - 1, 1)$ are partitions of k, then $f_k(\alpha) = f_k(\beta)$. These are the only collisions, and we may generate P_k by computing the preimages of P_{k-1} under f.

```
  partitions' n = concatMap preimages (partitions' (n-1))
```

One of the preimages of $(a_1, \ldots, a_r)$ under f_k will always be of the form $(a_1, \ldots, a_r, 1)$.

```
  preimages part = [part ++ [1]] ++ bump part
```

There is only a second preimage if incrementing the last integer yields a partition. This is true in two cases. Either we have a single integer partition or the last integer is less than the one immediately before it.

```
  bump part = case (reverse part) of
    y:[]       -> [[y+1]]    -- a one integer partition
    y1:y2:rest -> if y1 < y2 -- the last one is less
      then [reverse ((y1+1):y2:rest)]
      else []
```

We will denote the real span of $\mathrm{Irr}(S_n)$ by CS_n. Sorting the partitions of n using information in their type yields our chosen ordered basis for this set.

```
irrBasis :: (Natural n, Natural p) => n -> p -> [Partition n p]
irrBasis n p = sort (partitions n)
```

Similarly, CS'_n will denote the real span of $\mathrm{IBr}(S_n)$. Recall that $\mathrm{IBr}(S_n)$ is in one-to-one correspondence with the set of p-regular partitions of n.

```
ibrBasis :: (Natural n, Natural p) => n -> p -> [Partition n p]
ibrBasis n p = filter isRegular (irrBasis n p)
```

The p-regular partitions are the partitions that do not have the same integer appearing p or more times.

```
isRegular :: (Natural p) => Partition n p -> Bool
isRegular t = (all (<p) . map length . group . list) t
  where p = (value . getP) t
```

To represent vectors in CS_n or CS'_n we use a list of rational numbers. The type `s` in `Vec s` is used to distinguish which vector space a particular vector belongs to. Vectors in CS_n will have type `VecCSn n p`, a synonym for `Vec (CSn n p)`. Similarly vectors in CS'_n will have type `VecCSn' n p`.

```
type Scalar = Ratio Integer
newtype Vec s = Vec [Scalar] deriving (Show, Eq)
getSpace :: vec s -> s
getSpace = undefined
data CSn  n p = CSn
data CSn' n p = CSn'
type VecCSn  n p = Vec (CSn  n p)
type VecCSn' n p = Vec (CSn' n p)
```

We produce the vectors associated with basis elements by replacing every element of the basis with one or zero, as needed.

```
embed :: (Natural n, Natural p) => Partition n p -> VecCSn n p
```

```
embed t = Vec (fmap (\x -> if x == t then 1 else 0) basis)
  where
  basis = irrBasis (getN t) (getP t)
embed' :: (Natural n, Natural p) => Partition n p -> VecCSn' n p
embed' t = if isRegular t
  then Vec (fmap (\x -> if x == t then 1 else 0) basis)
  else undefined
  where
  basis = ibrBasis (getN t) (getP t)
```

We are able to determine the dimension of a vector by using the type. This is done by computing the associated basis and counting the elements.

```
class HasDimension a where --A type class allows us to define a function
  dimension :: a -> Int    --with a different body depending on the type.
instance (Natural n, Natural p) => HasDimension (CSn n p) where
  dimension x = length (irrBasis (getN x) (getP x))
instance (Natural n, Natural p) => HasDimension (CSn' n p) where
  dimension x = length (ibrBasis (getN x) (getP x))
instance HasDimension s => HasDimension (Vec s) where
  dimension = dimension . getSpace
```

In our implementation of vector arithmetic we define the operations only on vectors in the same space. This constraint is encoded in the type `Vec s -> Vec s -> Vec s`. It is not possible to add vectors from different spaces without first performing an explicit conversion.

```
(^+^) :: Vec s -> Vec s -> Vec s --Addition.
(Vec xs) ^+^ (Vec ys) = Vec (zipWith (+) xs ys)

(^-^) :: Vec s -> Vec s -> Vec s -- Subtraction.
v ^-^ u = v ^+^ (negateV u)

negateV :: Vec s -> Vec s -- Negation.
negateV (Vec xs) = Vec (fmap negate xs)
```

```
(*^) :: Scalar -> Vec s -> Vec s --Scalar multiplication.
c *^ (Vec xs) = Vec (fmap (*c) xs)

(^*^) :: Vec s -> Vec s -> Scalar --Scalar product.
(Vec xs) ^*^ (Vec ys) = sum (zipWith (*) xs ys)
```

We are ready to use Schur's theorem and Pate's theorem to generate our known inequalities and associated vectors of CS_n. By Schur's theorem (thm 1.3), for every irreducible character χ we have $\chi \succeq 0$. The set $\mathrm{Irr}(S_n)$ form our basis, so to list the associated vectors we just embed the basis into CS_n.

```
thmSchur :: (Natural n, Natural p) => n -> p -> [VecCSn n p]
thmSchur n p = fmap embed basis
  where
  basis = irrBasis n p
```

Now for Pate's theorem (thm 3.23). If we have a partition of n of the form $\alpha = (a_1, \ldots, a_i, \ldots, a_k)$ and $\beta = (a_1, \ldots, a_i - 1, \ldots, a_k, 1)$ is also a partition of n then $\alpha \geq \beta$. Thus we have $\beta(\mathrm{id})\alpha - \alpha(\mathrm{id})\beta \succeq 0$. To generate these vectors we first determine the character degrees using the hook length formula, then for each partition of n we enumerate all the ways we can generate another partition by "moving a piece to the end."

Theorem 4.5 (Hook length formula [3, p. 56]). *The degree of the irreducible character corresponding to the partition p of n is given by*

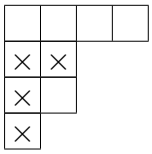
$$\frac{n!}{\prod_{(i,j)\in\lambda_p} h_{ij}}$$

where λ_p is the set of cell coordinates for the Young diagram of p and h_{ij} is the "hook length" of the cell.

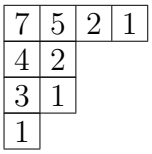
The hook length of a cell (i, j) is the size of the set

$$\{(x, y) | (x = i \wedge y \geq j) \vee (x \geq i \wedge y = j)\}.$$

For example, consider the partition $[4, 2^2, 1]$. The cell $(2, 1)$ has hook length 4 in the corresponding Young diagram.



Computing the hook length of $(2, 1)$.



The hook length of every cell.

Using the forumula, we have that the degree of $[\![4, 2^2, 1]\!]$ is $\frac{9!}{7\times5\times2\times1\times4\times2\times3\times1\times1} = 216$. We may translate the hook formula directly.

```
xs @@ n = xs !! (n-1) --The list indexing operator !! uses indices that
                      --start with 0. Our operator @@ will start at 1.

degree :: (Natural n, Natural p) => Partition n p -> Scalar
degree t = fromIntegral (fac n `div` product hooklens)
  where
  n = (value . getN) t
  fac x = (product . map toInteger) [1..x]
```

Next we produce a list of all of the cell coordinates in the tableau. The height of our Young tableau is the length of the corresponding partition and the width is the greatest (first) element of this partition. To produce a list of all cells in the Young tableau we simply enumerate $\{(i, j)|1 \leq i \leq h, 1 \leq j \leq w\}$ and filter out all the coordinates that refer to positions outside our tableau.

```
  t' = list t
  h = length t'
  w = head t'
  coords = filter legal [(i,j) | i <- [1..h], j <- [1..w]]
```

Denote the current partition of interest by $t = (t_1, \ldots, t_h)$. To determine whether a pair of coordinates (i, j) refers to a cell in the corresponding Young tableau, we check that the column index j is less than t_i.

```
legal :: (Int,Int) -> Bool
legal (i,j) = j <= (t' @@ i)
```

We compute h_{ij} numerically as follows. The number of cells to the right of (i, j) is given by $t_i - j$ and the number of cells below is $|\{t_x | x > i, t_x \geq j\}|$.

```
hooklens = fmap hooklen coords
hooklen :: (Int, Int) -> Integer
hooklen pos = toInteger (1 + rightOf pos + below pos)
rightOf (i,j) = (t' @@ i) - j
below (i,j) = length [t' @@ x | x <- [(i+1)..h], (t' @@ x) >= j]
```

We enumerate the inequalities produced by Pate's theorem by, for each partition t of n, producing the inequalities where t appears as the greater term.

```
thmPate :: (Natural n, Natural p) => n -> p -> [VecCSn n p]
thmPate n p = concatMap thmPate1 basis
  where
  basis = irrBasis n p
```

To list the partitions less than $t = (t_1, \ldots, t_r)$, we loop through every $i \in \{1, \ldots, r\}$ and consider whether $s_i = (t_1, \ldots, t_i - 1, \ldots, t_r, 1)$ is also a partition of n. If so, Pate's theorem applies. We then know $\frac{[\![s]\!]}{[\![s]\!](\mathrm{id})} \geq \frac{[\![t_i]\!]}{[\![t_i]\!](\mathrm{id})}$ so we generate the vector $[\![s_i]\!](\mathrm{id})[\![t]\!] - [\![t]\!](\mathrm{id})[\![s]\!]$.

```
thmPate1 :: (Natural n, Natural p) => Partition n p -> [VecCSn n p]
thmPate1 t = fmap mkVector [decrement i | i <- [1..r], verify i]
  where
  mkVector s = (degree s *^ embed t) ^-^ (degree t *^ embed s)
  t' = list t
  r = length t'
```

We need to verify that $s_i = (t_1, \ldots, t_i - 1, \ldots, t_r, 1)$ is a partition. This is true when either t_i is the last term of t with $t_i > 1$ or t_i occurs before the last term with $t_i > t_{i+i}$.

```
verify :: Int -> Bool
verify i | i == r &&
           (t' @@ i) > 1              = True
```

```
  | i < r &&
    (t' @@ i) > (t' @@ (i+1)) = True
  | otherwise                 = False
```

We construct the partition s_i by gluing the following sequences end to end.

```
decrement :: Int -> Partition n p
decrement i = Partition (
  (take (i-1) t')  ++ [(t' @@ i) - 1] ++ (drop i t') ++ [1] )
```

$\{-\quad (t_1, \ldots, t_{i-1}, \qquad t_i - 1, \qquad t_{i+1}, \ldots, t_r, \qquad 1)\ -\}$

Remark 4.6. *This does not pick up every inequality between irreducible characters generated by Pate's theorem. For example consider the partitions of* 3 *:* $a = (3)$*,* $b = (2,1)$*, and* $c = (1^3)$*. Direct application of Pate's theorem gives us* $a \geq b$ *and* $b \geq c$*, but* $a \geq c$ *comes from transitivity. We are not generating the inequalities that come from transitivity, but this is not a problem because the vector* $a - c$ *is in the convex cone of the vectors* $\{b - a, c - b\}$*. Applying transitivity to extend our list of inequalities cannot create a vector outside the convex cone of the list we started with.*

We have encoded all of the inequalities we will use to generate our convex cone as vectors in CS_n, so now we just need to generate their images in CS'_n under the decomposition map. For this we use a type representing a linear map between two vector spaces. The type encodes the domain and codomain, but the internal representation is just a matrix of rational numbers.

```
data LinearMap d r = LinearMap [[Scalar]] deriving Show
domain :: LinearMap d r -> d
domain = undefined
range :: LinearMap d r -> r
range = undefined
matrix :: LinearMap d r -> [[Scalar]]
matrix (LinearMap m) = m
```

The decomposition matrix is then a value of this type.

```
type DecompositionMatrix n p = LinearMap (CSn n p) (CSn' n p)
```

We implement the action of a linear map f through matrix multiplication.

```
apply :: (HasDimension d, HasDimension r) =>  LinearMap d r -> Vec d -> Vec r
apply f (Vec v) = Vec [entry i | i <- [1..h]]
  where
  mat = matrix f
  entry i = sum [((mat @@ i) @@ j) * (v @@ j)| j <- [1..w]]
  --If  A ∈ ℂ^{h×w} ,  v ∈ ℂ^w ,  and  Av = w ∈ ℂ^h ,  then  w_i = Σ_{j=1}^{w} a_ij v_j .
  w = (dimension . domain) f
  h = (dimension . range ) f
```

The full set of vectors used to generate our convex cone is then the decomposition matrix applied to the results of `thmSchur` and `thmPate`.

```
cone :: (Natural n, Natural p) => DecompositionMatrix n p -> [VecCSn' n p]
cone d = fmap (apply d) (thmSchur n p ++ thmPate n p)
  where
  n = (getN . domain) d
  p = (getP . domain) d
```

Our linear program will be of the form $Ax = b$, where the columns of A are the vectors returned by `cone` and b is the vector representing the current Brauer character of interest. There is only one remaining obstacle to setting this up. Where are we going to get the decomposition matrix?

4.2 The Decomposition Matrix

In this section we will produce the decomposition matrix for a given symmetric group and prime by extracting it from the computer algebra system GAP [10]. The GAP system does not compute the decomposition matrices directly. Rather, it provides an interface to the Modular Atlas project [15] which aims to produce Brauer character tables for all groups in the ATLAS of finite groups [5].

As with the previous section, this is a literate Haskell program. It may be built using the command `ghc Decomposition.lhs`.

```
module Decomposition where
import ConstructLP
import Appendix
```

We begin by defining our interface to GAP. It is not necessary for us to capture all of GAP's functionality. It is sufficient to be able to execute a single command and extract the result. The following data type represents the GAP functions and values we will be using.

```
data Gap = T  String            --A literal value with no arguments.
         | F1 String Gap        --A function with a single argument.
         | F2 String (Gap,Gap)  --A two argument function in prefix notation.
         | I2 String Gap Gap    --An infix two-argument function.
         deriving (Show)
```

We then describe how to convert our internal representation of a GAP command into a string that GAP is able to use.

```
render :: Gap -> String
render (T s) = s
render (F1 f x) = concat [f,"(",render x,")"]
render (F2 f (x1,x2)) = concat [f,"(",render x1,",",render x2,")"]
render (I2 f x1 x2) = concat ["(",render x1,")",f,"(",render x2,")"]
```

The relationship between our internal GAP instructions and the ones we export is as follows.

```
> render (T "val")
val
> render (F1 "foo" (T "val"))
foo( val )
> render (F2 "foo" (T "val1") (T "val2"))
foo( val1 , val2 )
> render (I2 "foo" (T "val1") (T "val2"))
```

```
( val1 ) foo ( val2 )
```

Invoking GAP on the command line via `gap -q` runs a GAP session that accepts standard input and prints the results to standard output.

```
run_gap :: Gap -> IO String
run_gap cmd = readProcess "gap" ["-q"] (render cmd ++ ";\nquit;")
```

With this our interface to GAP is complete. The manual for the GAP character table library provides instructions for computing decomposition matrices [12], which we implement here.

```
sym :: (Natural n) => n -> Gap
sym n = T ("\"S" ++ (show . value) n ++ "\"")
—— > render (sym 3)
—— "S3"


(%) :: Gap -> Gap -> Gap
(%) = I2 "mod"
—— > render (T "x" % T "y")
—— x mod y


gapmodtbl :: (Natural n, Natural p) => n -> p -> Gap
gapmodtbl n p = F1 "CharacterTable" (sym n) % (T . show) p
—— > render ( gapmodtbl n6 n3 )
—— CharacterTable( "S6" ) mod 3

gap_decomposition_matrix :: (Natural n, Natural p) =>
  n -> p -> IO (DecompositionMatrix n p)
gap_decomposition_matrix n p = do
  gapout <- run_gap $  F1 "DecompositionMatrix" (gapmodtbl n p)
```

The output produced when asking GAP to display a decomposition matrix is identical to Haskell's list syntax.

```
gap> DecompositionMatrix( CharacterTable( "M11" ) mod 2 );
[ [ 1, 0, 0, 0, 0 ], [ 0, 1, 0, 0, 0 ], [ 0, 1, 0, 0, 0 ],
```

```
[ 0, 1, 0, 0, 0 ], [ 1, 1, 0, 0, 0 ], [ 0, 0, 1, 0, 0 ],
[ 0, 0, 0, 1, 0 ], [ 0, 0, 0, 0, 1 ], [ 1, 0, 0, 0, 1 ],
[ 1, 1, 0, 0, 1 ] ]
```

Due to the similarity we can parse the result into a matrix of integers by using the existing read function.

```
let mat = (fmap.fmap) fromIntegral $ read gapout
```

The rows and columns of this matrix are naturally not in the same order as our basis. To put them in order we need to ascertain which row corresponds to which partition of n, the details of which we defer for now.

```
labels <- gap_decomposition_row_labels n p
```

With this information we may then order the rows by comparing their partitions.

```
let sortRows rows =
      (fmap snd . sortOn fst . (zip labels)) rows
```

However, we cannot order the columns in a similar way. Recall that we assign labels to the columns of the decomposition matrix according to theorem 3.21, which specifies that the rows are in lexicographic order. We could briefly reorder the rows to deduce the column names, but this is not required since the order they are in admits the same approach.

Theorem 4.7. *Arrange the rows of the decomposition matrix so that*

- *rows corresponding to p-regular partitions come first,*
- *and for every pair of p-regular partitions α, β such that α majorizes β, row α precedes row β.*

Then for every p-regular partition π, the first nonzero entry of column π occurs on row π.

Proof. Fix the column π, and consider a partition α such that row α is above row π. Then α is p-regular and π does not majorize α. By theorem 3.21, the entry on row α is zero. From the same theorem, we know that the entry on row π is 1, completing the proof. □

Since our current ordering meets the hypothesis of this theorem, each column shares a label with the row on which its first nonzero entry occurs. The ordering we have imposed on the row labels is the same for the column labels, so once the columns are sorted the decomposition matrix will be lower triangular. There is only one column order with this property. If we skip deducing the column labels and just rearrange the columns to create a lower triangular matrix, the result will have the columns in the correct order. We do this by sorting based on the index of the first nonzero entry.

```
let sortCols = transpose . sortOn (elemIndex 1) . transpose
```

Then all we need to do is wrap the matrix in `LinearMap` to make the types match.

```
let mat' = (transpose . sortCols . sortRows) mat
return (LinearMap mat')
```

The row labels are contained in the character parameters of the ordinary character table. They are partitions represented as descending lists of integers.

```
gap_decomposition_row_labels :: (Natural n, Natural p) =>
  n -> p -> IO [Partition n p]
gap_decomposition_row_labels n p = do
    let tbl = F1 "OrdinaryCharacterTable" ( gapmodtbl n p )
        second g = F2 "List" (g, T "x_->_x[2]")
    gapout <- (run_gap . second . F1 "CharacterParameters") tbl
```

Similarly to before, we exploit the fact that lists are represented the same way in GAP and in Haskell. The `read` command can already parse them without extra work on our part.

```
    let ps = map Partition . read $ gapout
    return ps
```

4.3 Producing a solution

As before, this section is a literate Haskell program. It may be built by `ghc Solver.lhs`. Here we assemble all of the previous work and run it through an external linear program solver to answer the question of cone membership.

```
module Solver where
import Decomposition
import ConstructLP
import Appendix
```

We assume the solver is in the `lp` directory, and that it accepts the description of a linear program in MPS fixed column format. Fixed MPS format[14] was originally used for linear programming on mainframe systems in the 1960s and has since become an industry standard. We further assume that the solver produces a Haskell expression representing a value of type `Either [Scalar] [Scalar]` and sends it to standard output. The `Right` values will be used for solutions and the `Left` values will be used for Farkas certificates.

```
solver = "./lp/solver.sh"
```

The search function combines all of our previous work. Given a positive integers n, k and a prime p, we generate a cone of inequalities of Brauer characters of S_n with respect to p. Then we set up the linear program for determining whether the kth irreducible Brauer character (in the order we used for our basis) is a member of the cone.

```
search :: (Natural n, Natural p) =>
          n -> p -> Int -> IO ( Either [Scalar] [Scalar] )
search n p k = do
  d <- gap_decomposition_matrix n p
  let cols = cone d
  let target = (embed' . (@@k)) (ibrBasis n p)
  let mps = mps_lp cols target --The details of expressing a linear program
                               --in MPS format will come later.
```

Next, we record the linear program (in case we want to inspect it later) and dispatch our linear program solver.

```
let mpsfile = concat ["./lp/",show n,"/",show p,"/",show k,".mps"]
writeFile mpsfile mps
result <- readProcess solver [] mps
```

The result should be either a solution or a Farkas certificate, and in either case we check that it has the correct algebraic property.

```
let resultFile = concat ["./lp/",show n,"/",show p,"/",show k,".out"]
let solution = read result :: Either [Scalar] [Scalar]
let integrity = verify cols target solution
if integrity then writeFile resultFile result
             else error "Something_has_gone_terribly_wrong."
return solution
```

The MPS fixed format requires us to name the rows and columns of our linear system. We will just name them after the order in which they appear.

```
rowname :: Int -> String
rowname i = ("R" ++ show i)
colname :: Int -> String
colname i = ("X" ++ show i)
```

A data file in MPS fixed format represents a deck of computer input cards, with each row corresponding to a card. Each row is divided into 6 fields, determined by the character position. The positions of the first characters in each field are 1, 5, 15, 25, 40, and 50 respectively. The file is divided into sections delimited by rows with the section indicator in field 1. The meaning of the data in other fields is determined by the section the row is in.

```
mps_lp :: (Natural n, Natural p) => [VecCSn' n p] -> VecCSn' n p -> String
mps_lp columns rhs = concat
  [ mps_section_name    columns rhs
  , mps_section_rows    columns rhs
  , mps_section_columns columns rhs
```

```
  , mps_section_rhs     columns rhs
  , "ENDATA\n"  ]
```

In order to correctly fit our data into the MPS fixed format we will be using these utility functions to append whitespace to either the left or right side.

```
padRight :: Int -> String -> String
padRight x = take x . (++spaces)
  where
  spaces = ' ':spaces
padLeft :: Int -> String -> String
padLeft x s = (reverse . take x . (++spaces) . reverse) s
  where
  spaces = ' ':spaces
```

The (optional) name section consists of just the indicator NAME in field 1 and the name of the problem in field 3.

```
mps_section_name :: (Natural n, Natural p) =>
  [VecCSn' n p] -> VecCSn' n p -> String
mps_section_name _ v = concat
  [ padRight 14 "NAME"
  , "S", (show . value . getN . getSpace) v
  , "P", (show . value . getP . getSpace) v
  , "\n"  ]
```

In the rows section field 1 is used to indicate the type of inequality used (we always use E for equality constraints, but G and L may be used to indicate $\geq$ and $\leq$ respectively), and field 2 indicates the name of the corresponding row.

```
mps_section_rows :: (Natural n, Natural p) =>
  [VecCSn' n p] -> VecCSn' n p -> String
mps_section_rows _ (Vec coeffs) = "ROWS\n" ++ concatMap f [1..d]
  where
  f x = "␣E␣␣" ++ rowname x ++ "\n"
  d = length coeffs
```

In the `COLUMNS` section, the fields have the following meanings.

Field 1	Blank.
Field 2	Column name.
Field 3	Row name.
Field 4	The value of the coefficient in the position specified by field 2 and field 3.
Field 5	(Optional) row name.
Field 6	(Optional) value of the coefficient in the position specified by field 2 and field 5.

For example, in the linear system

$$\begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 3 & 0 \end{bmatrix} x = \begin{bmatrix} 5 \\ 0 \\ 2 \end{bmatrix}, \tag{4.1}$$

if we give the rows and columns the names `R1`, `R2`, `R3` and `C1`, `C2`, `C3` respectively the `COLUMNS` section will be as follows.

```
COLUMNS
    C1        R1                  1   R2                  2
    C1        R3                  1
    C2        R2                  1   R3                  3
    C3        R1                  2
```

We omit the entries corresponding to zeroes since every coefficient we do not specify will be assumed to be zero.

Here we create the ordered pairs necessary to fill the columns section, but the work of organizing the data into the correct format is in the implementation of the `column` function.

```
mps_section_columns :: (Natural n, Natural p) =>
  [VecCSn' n p] -> VecCSn' n p -> String
```

```
mps_section_columns cols v =
  "COLUMNS\n" ++ concat (zipWith format [1..] cols)
  where
  format col (Vec coeffs) =
    column (colname col)
    [ (rowname row, c) |  (row,c) <- zip [1..] coeffs, c /= 0]
```

The `column` function expects a column name and a list of ordered pairs of the form (row name, coefficient), and it generates the corresponding rows of the MPS record.

```
column :: String -> [(String,Scalar)] -> String
column col [] = ""
column col [(r,c)] = concat
      [ "␣␣␣␣", padRight 8 col
      ,   "␣␣", padRight 8 r
      ,   "␣␣", (padLeft 12 . show . floor) c
      ,   "\n"]
column col ( (r1,c1):(r2,c2):rest ) = concat
      [ "␣␣␣␣", padRight 8 col
      ,   "␣␣", padRight 8 r1
      ,   "␣␣", (padLeft 12 . show . floor) c1
      , "␣␣␣", padRight 8 r2
      ,   "␣␣", (padLeft 12 . show . floor) c2
      ,   "\n"] ++ column col rest
```

The `RHS` section specifies the coefficients of the right hand side. The format is the same as the `COLUMNS` section. The `RHS` section corresponding to equation (4.1) is the following.

```
RHS
    rhs         R1                    5    R3                    2
```

To populate it we rely on the previously defined `column` function.

```
mps_section_rhs :: [VecCSn' n p] -> VecCSn' n p -> String
mps_section_rhs _ (Vec rhs) = "RHS\n" ++
  column "rhs" [ (rowname row, c) |  (row,c) <- zip [1..] rhs, c /= 0]
```

Once we have retrieved an answer from the linear program solver, we make sure it has the desired algebraic property. We expect a `Right` value to be a nonnegative solution to the provided linear system.

```
verify :: [VecCSn' n p] -> VecCSn' n p -> Either [Scalar] [Scalar] -> Bool
verify cols b (Right x) =
  (all (>= 0) x) && (sumv (zipWith (*^) x cols) == b)
  where
  sumv = foldr (^+^) zero
  zero = Vec (repeat 0)
```

We expect a `Left` value to be a Farkas certificate. Geometrically, a Farkas certificate is the normal vector to a hyperplane separating the vector on the right hand side of the equation defining the linear program from the convex cone of the columns of the matrix. To verify that a given vector y is a Farkas certificate, we check that the scalar product of y with the right hand side b is nonzero and that for every column c we have $\mathrm{sig}(\langle c, y\rangle) \neq \mathrm{sig}(\langle b, y\rangle)$, where

$$\mathrm{sig}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

This ensures that every column vector either lies on the hyperplane orthogonal to y or is on a different side of it from b.

```
verify cols b (Left y') = (s /= 0) &&
  (not . (any sameSide)) cols
  where
  y = Vec y'
  s = b ^*^ y
  sameSide c = signum (c ^*^ y) == s
```

4.4 Two complete examples

In this section we show the application of this technique from start to finish. First we consider the problem of proving Corollary 3.26. We want to show that the function $\{\!\{5,1\}\!\}$ is in the convex cone of a set of vectors $S \subseteq \mathbb{C}S_n$ satisfying $d_v(A) \geq 0$ for all $v \in S$ and all $A \in \mathcal{P}$. We populate S by applying the decomposition map to every vector of $\mathrm{Irr}(S_6)$ and every inequality yielded by Pate's theorem (3.23). The vectors of S then make up the columns of the matrix on the left hand side of Equation 4.2.

$$\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 4 & 2 & -9 & 0 & -5 & 0 & 9 & 5 & 0 & 9 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -9 & -9 & 11 & 1 & 9 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -9 & 0 & 11 & 0 & 9 & 5 & -9 & 0 & -2 & -4 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -9 & -9 & -5 & 0 & 9 & -3 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & -9 & 0 & -5 & -1 & 9 & -3 & -9 & 9 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -16 & 0 & 5 & -5 & 0 & 0
\end{bmatrix} x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.2}$$

The vector $\{\!\{5,1\}\!\}$, when written in terms of our chosen basis is $\langle 0,1,0,0,0,0,0\rangle$. Thus the problem of writing $\{\!\{5,1\}\!\}$ as a nonnegative linear combination of vectors in S is the linear program 4.2. Written out in MPS format, this linear program looks like so.

```
NAME          S6P3
ROWS
 E  R1
 E  R2
 E  R3
 E  R4
 E  R5
 E  R6
 E  R7
COLUMNS
    X1        R1                    1
```

X2	R1	1	R2	1
X3	R3	1		
X4	R2	1	R4	1
X5	R2	1	R5	1
X6	R1	1	R2	1
X6	R4	1	R5	1
X6	R6	1		
X7	R7	1		
X8	R1	1	R6	1
X9	R5	1	R6	1
X10	R4	1	R6	1
X11	R4	1		
X12	R1	4	R2	-1
X13	R1	10	R2	5
X13	R5	-5		
X14	R1	-9	R2	-9
X14	R3	16	R4	-9
X14	R5	-9	R6	-9
X15	R2	-9	R3	10
X15	R5	-9		
X16	R1	-5	R2	11
X16	R4	11	R5	-5
X16	R6	-5		
X17	R2	10	R6	-10
X18	R1	9	R2	9
X18	R4	9	R5	9
X18	R6	9	R7	-16

```
    X19       R1                  10   R2                  10
    X19       R4                  10   R5                  -6
    X19       R6                  -6
    X20       R4                  -9   R6                  -9
    X20       R7                   5
    X21       R1                   9   R6                   9
    X21       R7                  -5
    X22       R4                 -10   R5                   5
    X22       R6                  -5
    X23       R4                  -4   R6                   1
RHS
    rhs       R2                   1
ENDATA
```

Running this through a linear program solver produced the following solution vector.

```
[0,0,0,0,0,1/4,0,0,0,0,0,0,1/135,0,0,7/108,0,0,0,0,0,1/45,5/27]
```

Remark 4.8. *This solution is different from the vector used to prove corollary 3.26. We do not generally expect the solutions to be unique, but the reason for the difference in this case is that we used vectors with integer coefficients to generate our convex cone.*

From the existence of this vector we may conclude that $\{\!\{5,1\}\!\} \succeq_{\mathcal{P}_6(3)} 0$, which in turn implies that $\{\!\{5,1\}\!\} \geq_{\mathcal{P}} \varepsilon$.

Next we consider the group S_8 and the prime $p = 5$. We will apply this technique to the Brauer character $\{\!\{4^2\}\!\}$, which is the fifth element of $\mathrm{IBr}(S_8)$ by the ordering we defined in Section 4.1. The linear program for determining whether $\{\!\{4^2\}\!\}$ is in the convex cone generated by the Schur vectors and the Pate vectors is the following.

$$
\begin{bmatrix}
1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&7&0&0&0&0&0&70&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&21&0&0&70&64&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&64&21&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&70&64&0&0&0&0&0&0&0&0&70&56&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&70&0&0&0&-70&0&0&0&0&-42&0&0&0&64&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-7&-20&-20&0&-28&0&35&90&35&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-20&0&0&-28&0&0&90&35&0&0&70&90&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-28&0&-14&0&0&0&56&90&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&70&90&0&0&0&0&0&0&0&0&28&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&70&56&0&0&0&0&-70&0&-14&0&0&0&20&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-21&0&-64&0&0&0&0&0&0&35&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-64&0&0&-70&0&-56&0&0&0&64&35&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-70&0&0&0&0&-42&0&-90&0&8&0&-70&0&0&20&21&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-56&0&-42&0&0&0&0&0&28&64&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&28&0&0&0&0&0&0&-7\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-35&0&-90&0&0&0&0&21&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-90&0&-56&0&-70&0&-35&20&-43&0&0&7&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-70&0&-14&0&0&0&20&-20&-21&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-64&0&-28&7&0&0
\end{bmatrix}
x =
\begin{bmatrix}
0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0
\end{bmatrix}
$$

After attempting to solve this linear program we produced the following Farkas certificate.

$$
\begin{bmatrix}
3297517523435520000000\\
0\\
2997743203123200000000\\
4196840484372480000000\\
-11990972812492800000000\\
0\\
9592778249994240000000\\
10492101210931200000000\\
1049210121093120000000\\
2098420242186240000000\\
0\\
13489844414054400000000\\
9592778249994240000000\\
10492101210931200000000\\
0\\
0\\
0\\
0\\
0
\end{bmatrix}
$$

From this we know that $\{\!\{4^2\}\!\}$ is not in the convex cone. This does not, however, mean that there exists a p-regular matrix A such that $d_{\{\!\{4^2\}\!\}}(A) < 0$, in other words, that $\{\!\{4^2\}\!\} \not\succeq_{\mathcal{P}} 0$. It just means that the inequalities due to Schur's theorem and Pate's theorem are not sufficient to guarantee the inequality $\{\!\{4^2\}\!\} \succeq_{\mathcal{P}} 0$. (If it should turn out that this inequality does not hold, this would provide a counterexample to Conjecture 3.11, since if $\{\!\{4^2\}\!\} \not\succeq_{\mathcal{P}} 0$ there exists a matrix $A \in \mathcal{P}$ such that $d_{\{\!\{4^2\}\!\}}(A) < 0 \leq \det(A)$.)

4.5 Linear programming appendix

In this section we consolidate the external dependencies of our program and include the definitions that are tangentially related to our previous work.

```
module Appendix
        ( module Appendix
        , module Data.List
        , module Text.Read
        , module Data.Ratio
        , readProcess
        ) where
```

The source code and documentation for our external dependencies are available on Hackage [13].

```
import Data.Ratio (Ratio, numerator, denominator)
import Data.List (sort, group, elemIndex, transpose, sortBy)
import Data.Ord (comparing)
import System.Process (readProcess, waitForProcess, runCommand)
import Text.Read (readMaybe)
```

The `sortOn` function sorts a list such that the images of the elements under the provided function `f` are in ascending order.

```
sortOn :: (Ord b) => (a -> b) -> [a] -> [a]
sortOn f = fmap fst
         . sortBy (comparing snd)
         . fmap (\x -> (x, f x))
```

The types `One` and `S` define our type level natural numbers. For example, the number three is represented as `S (S One)`.

```
data One = One
data S a = S a
```

In order to convert a type level natural to an integer we use the following `Natural` type class. A type class is needed here rather than an ordinary function because the domain is not represented by a single type. `One` and `S One` are different types.

```
class (Show a) => Natural a where
  value :: (Num b) => a -> b
```

We define the `Natural` instances recursively, starting with `One`.

```
instance Show One where
  show _ = "1"
instance Natural One where
  value _ = 1
instance (Natural n) => Show (S n) where
  show = show . value
instance (Natural n) => Natural (S n) where
  value = (+1) . value . (undefined :: S n -> n)
```

The instance of `Natural` for `S n` work as follows. The `value` of a variable x of type `S n` is defined to be 1 plus the `value` of a variable y of type `n`. Since it is possible to compute the `value` of y without knowing anything about it other than its type, the `undefined` does not get evaluated.

Next, we include abbreviations of some low order type level naturals for convenience.

```
type N1 = One
type N2 = S N1
type N3 = S N2
type N4 = S N3
type N5 = S N4
type N6 = S N5
type N7 = S N6
type N8 = S N7
type N9 = S N8
type N10 = S N9
type N11 = S N10
type N12 = S N11
type N13 = S N12
type N14 = S N13
type N15 = S N14
type N16 = S N15
type N17 = S N16
```

```
type N18 = S N17
type N19 = S N18
type N20 = S N19
type N21 = S N20
type N22 = S N21
type N23 = S N22
type N24 = S N23
```

We include the following values to be used as arguments to functions that expect type level naturals. These are used only to disambiguate type information and never get evaluated. They may be converted to integer values purely by inspecting their types, as shown in the definition of `value` above.

```
n1 = undefined :: N1
n2 = undefined :: N2
n3 = undefined :: N3
n4 = undefined :: N4
n5 = undefined :: N5
n6 = undefined :: N6
n7 = undefined :: N7
n8 = undefined :: N8
n9 = undefined :: N9
n10 = undefined :: N10
n11 = undefined :: N11
n12 = undefined :: N12
n13 = undefined :: N13
n14 = undefined :: N14
n15 = undefined :: N15
n16 = undefined :: N16
n17 = undefined :: N17
n18 = undefined :: N18
n19 = undefined :: N19
n20 = undefined :: N20
```

```
n21 = undefined :: N21
n22 = undefined :: N22
n23 = undefined :: N23
n24 = undefined :: N24
```

Chapter 5

Further Work

We have used Pate's theorem (3.23) to generate most of the dominance inequalities we used on $\mathbb{C}S_n$. However, the continued work of T. H. Pate includes many more results of this nature, a few of which we will mention here. Pushing these through the decomposition map would surely yield more relations between Brauer characters.

Theorem 5.1 (Pate [8]). *If α is a partition of n of the form $(p, q^w, 2^s, 1^t)$ where p, q, s, t, and w are non-negative integers and $0 \leq w \leq 2$, then $[\![\alpha]\!] \leq [\![n]\!]$.*

These next results are about the ordering $\ll$ on partitions of n. Given partitions α, β of n we say $\alpha \ll \beta$ if for every partition ω of m such that the sequence concatenations (ω, α) and (ω, β) are partitions of $m + n$ we have $[\![(\omega, \alpha)]\!] \leq [\![(\omega, \beta)]\!]$. We extend the definition to include the case where ω is the empty sequence, so $\alpha \ll \beta$ implies $[\![\alpha]\!] \leq [\![\beta]\!]$.

Theorem 5.2 (Pate [9, thm. 14]). *Suppose $\alpha = (\alpha_1, \ldots, \alpha_s)$ is a partition of n. For $0 \leq i < s$, define $\beta_i = \left(\alpha_1, \alpha_2, \ldots, \alpha_i, \sum_{j=i+1}^{s} \alpha_j\right)$. If β_i is a partition of n for every such i, we have*

$$\alpha = \beta_{s-1} \ll \cdots \ll \beta_1 \ll \beta_0 = (n).$$

Theorem 5.3 (Pate [8, thm. 7]). *If n, p and k are positive integers such that $p \geq n + k$, then $(p, n^{k+1}) \ll (n + p, n^k)$.*

Consider the partition $\alpha = (4, 2^2)$. By Theorem 5.2 we have $(4, 2^2) \ll (4^2) \ll (8)$, thus $[\![4, 2^2]\!] \leq [\![4^2]\!] \leq [\![8]\!]$. These inequalities were not included as basis vectors for the convex cone we generated, so including them may pick up vectors we previously missed.

The approach used in this dissertation is unlikely to scale well to larger groups. Unfortunately the worst case time complexity of the simplex method, typically used to find a solution to a linear program, is not very good. There exist examples ([4]) of linear programs where the time complexity is exponential in the size of the problem. While the linear programs we constructed do not necessarily achieve the worst case performance – in fact it is clear that in many cases they did not – no care was taken to prevent this from happening. We describe a technique by which we may generate convex cones such that testing vectors for membership has better worst case performance. However the trade off is that since these are subsets of the convex cones we previously considered we may miss vectors that we previously might have picked up.

Let V_1 and V_2 be vector spaces and put $V = V_1 \oplus V_2$. We observe that if C is the convex cone of the set $X \subseteq U$ and $X = X_1 \cup X_2$, with $X_1 \subseteq V_1$ and $X_2 \subseteq V_2$, then v is an element of C if and only if there exist vectors $v_1 \in V_1$ and $v_2 \in V_2$ such that $v = v_1 + v_2$ and v_1 and v_2 are in the convex cones of X_1 and X_2 respectively. Thus the computational complexity of determining whether $v \in C$ is bounded by the complexity of determining whether the projections of v onto V_1 and V_2 are in their respective cones.

Since the worst case complexity is exponential in the dimension of V this is a significant improvement. Suppose the dimensions of V_1 and V_2 are both 10, so that the dimension of V is 20. We see that 2^{20} is far greater than $2^{10} + 2^{10}$.

We can use the block structure ([6, p. 244]) of the decomposition matrix to obtain a cone with a decomposition as just described, namely, the cone corresponding to only those inequalities that involve irreducible characters in the same block. Although this cone is not as large as the one obtained by using the full list of inequalities, it provides a natural way of getting results in some situations where the computations are prohibitively time consuming otherwise (cf. Appendix A).

Consider the example of S_7 with the prime 3. The decomposition matrix, when written in block form, is the following.

	$\{\!\{7\}\!\}$	$\{\!\{5,2\}\!\}$	$\{\!\{4,3\}\!\}$	$\{\!\{4,2,1\}\!\}$	$\{\!\{3,2,1^2\}\!\}$	$\{\!\{6,1\}\!\}$	$\{\!\{3,2^2\}\!\}$	$\{\!\{5,1^2\}\!\}$	$\{\!\{3^2,1\}\!\}$
$[\![7]\!]$	1	0	0	0	0				
$[\![5,2]\!]$	1	1	0	0	0				
$[\![4,3]\!]$	0	1	1	0	0				
$[\![4,2,1]\!]$	1	1	1	1	0				
$[\![3,2,1^2]\!]$	1	0	1	1	1				
$[\![4,1^3]\!]$	0	0	0	1	0				
$[\![2^3,1]\!]$	1	0	0	0	1				
$[\![2^2,1^3]\!]$	0	0	1	0	1				
$[\![1^7]\!]$	0	0	1	0	0				
$[\![6,1]\!]$						1	0		
$[\![3,2^2]\!]$						1	1		
$[\![3,1^4]\!]$						0	1		
$[\![5,1^2]\!]$								1	0
$[\![3^2,1]\!]$								1	1
$[\![2,1^5]\!]$								0	1

We organize the irreducible characters into their respective blocks and remove all inter-block relationships. For an example of the resulting partial order see Figure 5.2.

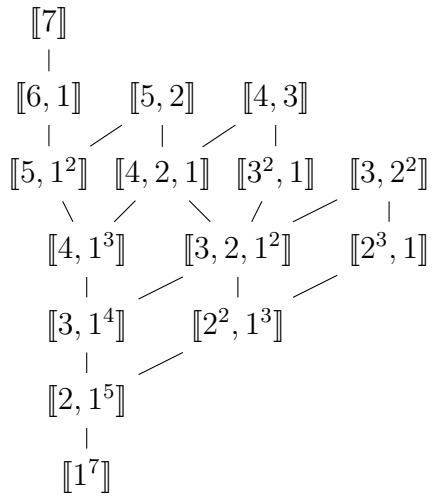


Figure 5.1: Pate's theorem on S_7

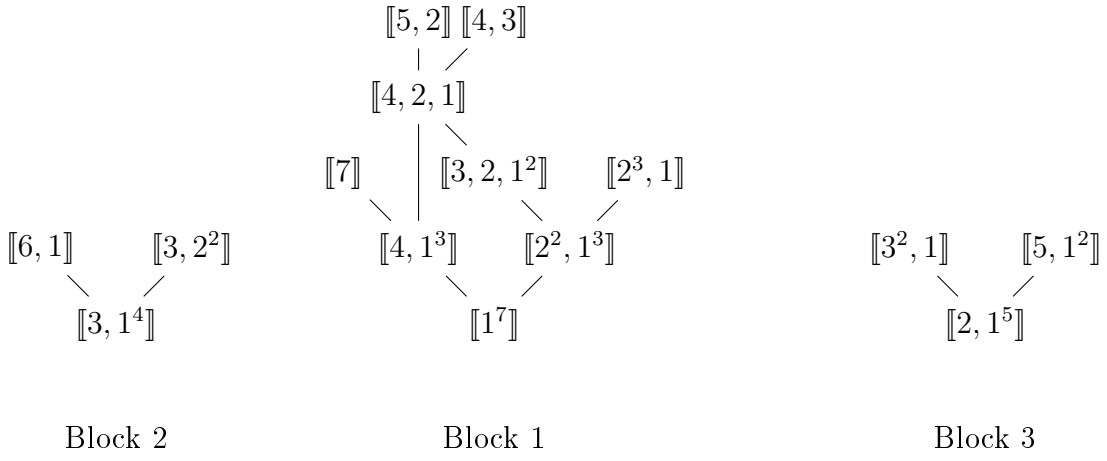


Figure 5.2: Pate's theorem on S_7, split into blocks.

Appendix A

Generated Results

The following are the results we have programmatically checked. For most of the cases given a prime p and a character $\phi \in \mathrm{IBr}(S_n)$, we found that $\phi \succeq_{\mathcal{P}_n(p)} 0$. Because of this we will save space by writing out the characters for which the problem proved infeasible. We point out once again that the fact that a Brauer character produces an infeasible problem does not say that it is a counterexample to Conjecture 3.11. Rather, it just says that our method, using Schur's Theorem and Pate's Theorem alone, cannot be used to check the conjecture for that Brauer character.

Group	Prime	Infeasible
S_6	3	none
S_6	5	none
S_7	3	none
S_7	5	none
S_7	7	$\{\!\{6,1\}\!\}$
S_8	3	none
S_8	5	$\{\!\{4^2\}\!\}$, $\{\!\{5,2,1\}\!\}$
S_8	7	$\{\!\{6,2\}\!\}$
S_9	3	$\{\!\{8,1\}\!\}$
S_9	5	$\{\!\{6,3\}\!\}$, $\{\!\{5,3,1\}\!\}$, $\{\!\{5,2,1^2\}\!\}$
S_9	7	$\{\!\{7,2\}\!\}$, $\{\!\{6,3\}\!\}$
S_{10}	3	$\{\!\{8,2\}\!\}$
S_{10}	5	$\{\!\{9,1\}\!\}$, $\{\!\{6,4\}\!\}$, $\{\!\{5^2\}\!\}$, $\{\!\{8,1^2\}\!\}$, $\{\!\{6,3,1\}\!\}$, $\{\!\{5,4,1\}\!\}$, $\{\!\{5,3,1^2\}\!\}$, $\{\!\{4,3,2,1\}\!\}$
S_{10}	7	$\{\!\{7,3\}\!\}$, $\{\!\{6,4\}\!\}$, $\{\!\{7,2,1\}\!\}$

From S_{11} onward the problem grew too large for us to approach this way. The worst case time complexity for solving a linear program is exponential in both the number of equations and the number of variables. The system we generated for S_{11} and the prime 3 has 174 variables and 27 equations, which appears to be past the limit for the software we used. However, we have some partial results for S_{11} and S_{12} produced by considering a subset of our cone generated by inequalities only between characters that share a block (see Chapter 5).

Group	Prime	Infeasible
S_{11}	3	$\{\!\{9,2\}\!\}$, $\{\!\{8,3\}\!\}$
S_{11}	5	$\{\!\{9,2\}\!\}$, $\{\!\{7,4\}\!\}$, $\{\!\{6,5\}\!\}$, $\{\!\{8,2,1\}\!\}$, $\{\!\{6,4,1\}\!\}$, $\{\!\{6,3,2\}\!\}$, $\{\!\{6,3,1^2\}\!\}$, $\{\!\{5,4,1^2\}\!\}$, $\{\!\{4,3^2,1\}\!\}$, $\{\!\{5,3,1^3\}\!\}$
S_{11}	7	$\{\!\{8,3\}\!\}$, $\{\!\{7,4\}\!\}$, $\{\!\{6,5\}\!\}$, $\{\!\{7,3,1\}\!\}$, $\{\!\{5^2,1\}\!\}$, $\{\!\{7,2,1^2\}\!\}$, $\{\!\{6,2^2,1\}\!\}$
S_{11}	11	$\{\!\{10,1\}\!\}$, $\{\!\{9,1^2\}\!\}$, $\{\!\{8,1^3\}\!\}$
S_{12}	3	No results.
S_{12}	5	$\{\!\{10,2\}\!\}$, $\{\!\{9,3\}\!\}$, $\{\!\{7,5\}\!\}$, $\{\!\{6^2\}\!\}$, $\{\!\{8,3,1\}\!\}$, $\{\!\{8,2^2\}\!\}$, $\{\!\{7,4,1\}\!\}$, $\{\!\{6,5,1\}\!\}$, $\{\!\{6,4,2\}\!\}$, $\{\!\{7,2^2,1\}\!\}$, $\{\!\{6,4,1^2\}\!\}$, $\{\!\{6,3,2,1\}\!\}$, $\{\!\{5^2,1^2\}\!\}$, $\{\!\{5,3^2,1\}\!\}$, $\{\!\{6,3,1^3\}\!\}$, $\{\!\{5,4,1^3\}\!\}$, $\{\!\{4,3^2,1^2\}\!\}$, $\{\!\{5,2^2,1^3\}\!\}$
S_{12}	7	$\{\!\{8,4\}\!\}$, $\{\!\{7,5\}\!\}$, $\{\!\{6^2\}\!\}$, $\{\!\{8,3,1\}\!\}$, $\{\!\{7,4,1\}\!\}$, $\{\!\{6,4,2\}\!\}$, $\{\!\{5^2,2\}\!\}$, $\{\!\{7,3,1^2\}\!\}$, $\{\!\{6,3,2,1\}\!\}$, $\{\!\{5^2,1^2\}\!\}$, $\{\!\{7,2,1^3\}\!\}$, $\{\!\{6,2^2,1^2\}\!\}$, $\{\!\{5,2^3,1\}\!\}$
S_{12}	11	$\{\!\{10,2\}\!\}$, $\{\!\{9,2,1\}\!\}$, $\{\!\{8,2,1^2\}\!\}$

Bibliography

[1] Larry L Dornhoff. *Group Representation Theory: Modular representation theory*. Vol. 2. M. Dekker, 1972.

[2] I Martin Isaacs. "Lifting Brauer characters of p-solvable groups". In: *Pacific Journal of Mathematics* 53.1 (1974), pp. 171–188.

[3] Gordon James and Adalbert Kerber. *The representation theory of the symmetric group, volume 16 of Encyclopedia of Mathematics and its Applications*. Addison-Wesley Publishing Co., Reading, Mass, 1981.

[4] Katta G Murty. "Linear programming". In: (1983).

[5] John Horton Conway. *Atlas of finite groups: maximal subgroups and ordinary characters for simple groups*. Oxford University Press, 1985.

[6] Russell Merris. *Multilinear algebra*. Vol. 8. CRC Press, 1997.

[7] Vilmos Komornik. "A Simple Proof of Farkas' Lemma". English. In: *The American Mathematical Monthly* 105.10 (1998), pp. 949–950. ISSN: 00029890. URL: http://www.jstor.org/stable/2589288.

[8] Thomas H Pate. "Tensor inequalities, ξ-functions and inequalities involving immanants". In: *Linear algebra and its applications* 295.1 (1999), pp. 31–59.

[9] Thomas H Pate. "Tensor inequalities, ξ-functions and inequalities involving immanants". In: *Linear algebra and its applications* 295.1 (1999), pp. 31–59.

[10] *GAP – Groups, Algorithms, and Programming, Version 4.7.7*. The GAP Group. 2015. URL: http://www.gap-system.org.

[11] Randall R. Holmes. *Linear Representations of Finite Groups*. 2015. URL: http://auburn.edu/~holmerr/book.pdf.

[12] *Decomposition Matrices in GAP*. URL: http://www.math.rwth-aachen.de/homes/MOC/htm/ctbldeco.htm.

[13] *"Hackage, the Haskell package archive"*. "http://hackage.haskell.org".

[14] *IBM ILOG CPLEX Optimization Studio reference manual*. URL: http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.4.0/ilog.odms.cplex.help/CPLEX/File_formats_reference/topics/MPS_records.html.

[15] *The Modular Atlas project*. URL: http://www.math.rwth-aachen.de/~MOC/.