



UiT The Arctic University of Norway

DEPARTMENT OF MATHEMATICS AND STATISTICS

## **Murnaghan-Nakayama Rule**

The Explanation and Usage of the Algorithm

Elias Sandal

Master's thesis in Mathematical Sciences – master MAT-3900 May 2023





## Summary

Character values are not the easiest to calculate, so it is important to find good algorithms that can help ease these calculations. In the 20th century, the two mathematicians Murnaghan and Nakayama developed a rule that calculates character values for partitions on some computations. This rule has later been given the name *The Murnaghan-Nakayama rule*, after Murnaghan and Nakayama.

The Murnaghan-Nakayama rule is a combinatorial method for computing character values of irreducible representations of symmetric groups. This makes this rule an important part of representation theory. One of the versions of this rule is stated in Theorem 15;

**Theorem 15.** *Let  $\lambda \vdash n$ , and suppose that  $\alpha = (\alpha_1, \dots, \alpha_m)$  is a composition of  $n$ . Then the character value  $\chi^\lambda$  is given by*

$$\chi_\alpha^\lambda = \chi^\lambda(\alpha) = \sum_{\nu} (-1)^{ht(\nu)} \chi^{\lambda \setminus \nu}(\alpha \setminus \alpha) \quad (33)$$

where we sum over all border strips  $\nu$  of size  $\alpha_1$ .

We can therefore use border strips and diagrams to calculate character values of representations on a given composition. This algorithm is quite fast in these calculations. The Murnaghan-Nakayama rule is therefore very favourable in these calculations.

The Murnaghan-Nakayama rule can also be considered a central algorithm in representation theory over symmetric groups. It is a fascinating and powerful algorithm that has a strong connection to both combinatorics and representation theory.



## Foreword

I am honoured to present this master's thesis, which is a personal highlight of mine. This is because I got the opportunity to conduct more in-depth research in representation theory, as well as the ability to apply theoretical mathematics more visually. When visualising theoretical mathematics, one might reach out to a broader audience, where people who better understand visualised theory have an opportunity to learn more abstract mathematics.

The software used for this thesis is SageMath by The Sage Developers, where I mainly used their online service <https://cocalc.com/>. I found this program to be perfect for programming the algorithm of the Murnaghan-Nakayama rule, due to all of its integrated functions. The fact that its language is heavily influenced by Python was also a plus, as it's a language I am quite familiar with.

I have been working on this Master's Thesis since December 2022. Though this last semester has been quite stressful, it has also been quite eye-opening. Both in the academic sense and also in my capability to work in such a short period.

During this period I have been very fortunate to receive great support and guidance from several great individuals.

First and foremost, I would like to express my gratitude to my supervisor, Cordian Riener. He introduced me to representation theory in one of his courses and suggested the Murnaghan-Nakayama rule as a topic for my thesis. His encouragement, expertise, and enthusiasm have been of great help in forming this thesis.

I would also like to give a heartfelt thanks to my friends. They gave me great encouragement and advice. This helped me navigate all the challenges of graduate school and has made this journey more enjoyable and rewarding. And a special thanks to Isabel, who was a great source of motivation and inspiration when I wrote this master thesis. She was always there when I was struggling, and gave me great advice.

Finally, I would like to acknowledge the support of my family. They gave me great support throughout my academic and personal goals. I would like to give a big thank you to my sibling who, although they are not as involved in mathematics as I am, were excited to hear me present this topic to them.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	P and NP complexity classes . . . . .	2
1.3	History behind the rule . . . . .	4
1.4	Structure of the Master Thesis . . . . .	5
<b>2</b>	<b>Representation Theory on General Groups</b>	<b>6</b>
2.1	Conjugate Classes of Groups . . . . .	6
2.2	Representations of Groups . . . . .	8
2.2.1	Matrix Representation . . . . .	8
2.2.2	Modules . . . . .	9
2.2.3	Decomposition of Representations . . . . .	10
2.3	Characters of Groups . . . . .	12
2.3.1	Character Table . . . . .	16
<b>3</b>	<b>Representation Theory on Symmetric Groups</b>	<b>22</b>
3.1	Conjugacy Classes of Symmetric Groups . . . . .	24
3.2	Representations of Symmetric Groups . . . . .	27
3.3	Diagrams . . . . .	30
3.3.1	Tableaux . . . . .	36
3.4	Specht Modules . . . . .	41
3.5	Branching Rule . . . . .	44
3.6	Symmetric Functions . . . . .	46

3.6.1	Schur Function . . . . .	47
3.6.2	Characteristic Map . . . . .	50
3.6.3	Littlewood-Richard Rule . . . . .	50
<b>4</b>	<b>Murnaghan–Nakayama rule</b>	<b>54</b>
4.1	Alternative Versions . . . . .	64
4.1.1	Quantum Murnaghan-Nakayama rule . . . . .	64
4.1.2	Plethystic Murnaghan-Nakayama rule . . . . .	64
4.2	Usage of the Murnaghan-Nakayama rule . . . . .	66
4.2.1	Finding unknown partitions . . . . .	66
4.2.2	Expansions of Irreducible Symmetric Group-characters . . . . .	70
4.3	Results from Pak and Panova . . . . .	72
<b>5</b>	<b>Computation</b>	<b>74</b>
5.1	Predefined Functions . . . . .	74
5.1.1	Characters of Representations . . . . .	74
5.1.2	Character Tables . . . . .	76
5.2	Murnaghan-Nakayama Recursive Algorithm . . . . .	80
5.3	Comparison Of Functions . . . . .	88
5.3.1	Single Character value . . . . .	89
5.3.2	List of Character Values . . . . .	92
5.3.3	Character table . . . . .	94
5.3.4	Result . . . . .	96
<b>6</b>	<b>Conclusion</b>	<b>99</b>





# 1 Introduction

## 1.1 Motivation

In representation theory, characters have a central role in the study of representations of groups (and algebras). A character is a function that associates each element of a group with a complex number. Characters are considered to be a way of measuring how an element in a group act on a vector space. The character then encodes how the element transforms vectors in a vector space.

Characters are important since they give us the ability to distinguish different representations of a group, where we will consider two representations to be equivalent if they have the same character. Characters play an important role in subjects like algebraic geometry, quantum mechanics, and molecular chemistry, to name a few. For example in molecular chemistry, one uses characters to understand and predict the properties of molecules and their electronic states.

Finding these character values can be quite difficult. Just finding out if a representation has a character value equal to zero has been proved to be hard and complicated by Pak and Panova (2015), see Section 4.3 for further explanation. However, in some special cases, one can compute the character values using combinatorial methods. One of these methods is the Murnaghan-Nakayama rule (Theorem 15), which computes the characters of irreducible representations of symmetric groups.

The Murnaghan-Nakayama rule is a fascinating algorithm that has extensive applications in several different fields. It is an evolving algorithm and there are still studies made to further refine and develop the rule. The rule may then be a crucial area of research in the future, with more potential for discoveries and applications in various fields.

## 1.2 P and NP complexity classes

To understand the complexity around characters and how to find their values in representation theory, we should understand what complexity means.

When talking about complexity, we often refer to the time an algorithm takes to solve a problem. If there is a problem we want to solve, we can either 1) try to find the solutions, or 2) see if a set of solutions can be efficiently verified.

Complexity classes are defined to be a collection of Decision Problems. These are problems where you take an input  $x$ , and want an answer, *yes* or *no*, as an output. Or, these problems could be collections of Search Problems. Search Problems are problems where we have an input  $x$ , and we want a correct answer  $y$  to the problem as an output. When looking at Decision Problems that can be solved in Polynomial Time by a Deterministic Turing Machine (or simply just a computer). We are then looking at a *P complexity class*. The P is short for "Polynomial Time". Polynomial Time implies there exists some polynomial function  $f$ , such that if there is the input  $x$ , with the length  $l$  when written in binary, then the machine should take at most  $f(l)$  time to give out the correct answer (maycontainmaths, 2014).

When looking at Search Problems, the P complexity is more or less the same. The only catch is that the length of the answer  $y$ , when written in binary, must always be less than  $g(l)$  for some polynomial  $g$ . If there were no bounds, then the machine would take too long to solve the problem.

If we have Decision Problems that could be solved in Polynomial time, but not by a Deterministic Turing Machine, we are then looking at the *NP complexity class*. The NP is short for "Non-deterministic Polynomial Time". A Non-deterministic Turing Machine doesn't match up with a computer we know. Meaning NP is not good at looking at Decision Problems. Therefore, NP is defined in terms of Search Problems instead.

Search Problem has *Efficiently Checkable Solutions* when, given a problem instance and an answer  $(x, y)$ , checking if  $y$  is a correct solution to  $x$  is in  $P$ . I.e., it is possible to verify the solutions in polynomial time. Here we still need the length of  $y$  to be less than  $g(l)$  for some polynomial  $g$ , such that the computer can be able to compute this in a reasonable time. NP is the collection of Search Problems with Efficient Checkable Solutions. From this, we can refer to P as a collection of problems that are solvable in polynomial time, while NP is the collection of problems for which the solutions can be verified in polynomial time (Mehlhorn and Sun, 2013).

A problem  $Y$  is NP-hard if the problem  $X \in \text{NP}$  can be reduced in a polynomial time to  $Y$ . Formally speaking; a problem  $X$  is polynomial time reducible to  $Y$  if there exists a polynomial time function  $f$  that sends elements in  $X$  to some elements in  $Y$ , such that  $f(x) = y$  (Mehlhorn and Sun, 2013).

From intuition, one can imagine that a NP-hard problem is at least as hard as a NP problem, though we don't have to look over NP problems. Meaning that the  $Y$  does not have to be a NP problem, even though the  $X$  is. When looking at NP-hard problems where  $Y \in \text{NP}$ , then we are looking at NP-complete problems.

By using the Euler diagram, we can get a better visualisation of how to compare NP-hard and NP-complete problems.

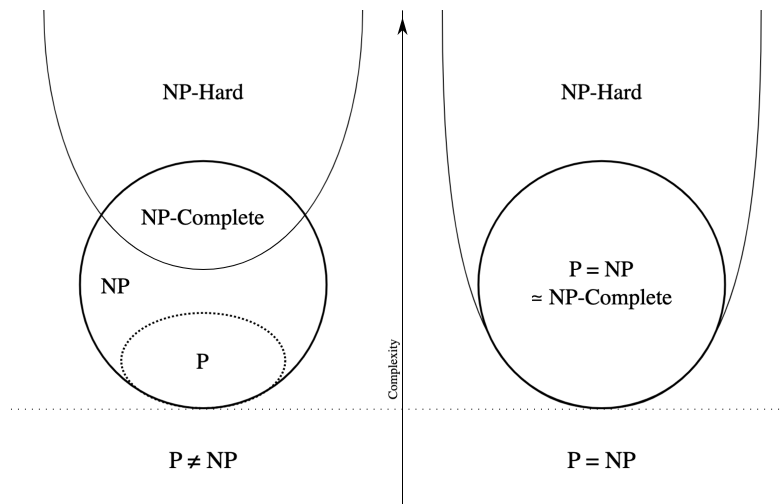


Figure 1: An Euler diagram for P, NP, NP-complete, and NP hard sets of problems Wikipedia contributors (2023a).

In NP-problems we ask whether there exists a solution to a given problem. However, one might also be interested in how many solutions in addition to if the solutions exist. We then let  $\#P$  be the set of counting problems associated with decision problems in the set of NP problems. While NP problems ask "are there any valid solutions", the  $\#P$  problems ask for "how many solutions" Valiant (1979). Since we can associate  $\#P$  with NP-problems, then  $\#P$  are as hard as NP. The same can be said for  $\#P$ -hard problems.

### 1.3 History behind the rule

The representation theory of symmetric groups was first studied by G. Frobenius in the late 19th century. Later authors like I. Schur, A. Young, and H. Weyl made important contributions to this field. Though, it was mostly R. Brauer with his collaborators that influenced the theory of modular representation in the 1930s (Lewis, 2003). This field of mathematics is crucial in nuclear physics, which is why F.D. Murnaghan wanted to simplify the theory of the representations of symmetric groups in a way that was more accessible for physicists in his paper "On the Representations of the Symmetric Group". One of the motivations for this paper was the following quote;

The original papers of Frobenius, and particularly those of Schur, arouse in a persevering reader an emotion akin to that inspired by one of the great symphonies; but they are by no means easy reading and we hope that a somewhat elementary orchestration may acquaint a larger audience with the word of the masters.

(Murnaghan 1937)

As Stanley and Fomin (1999) wrote in their notes on symmetric functions (Stanley and Fomin, 1999, p 401), the Murnaghan-Nakayama rule was originally proved by Littlewood and Richardson (1934, §11). In their proof, they derive it as a corollary of the older Frobenius formula for characters of symmetric groups. (See (Stanley and Fomin, 1999, 7.77) for a modern statement of Frobenius's formula.) Later F.D. Murnaghan (1937, p. 452, (13)) gave an independent derivation of the rule. T. Nakayama continued the study in 1940 and investigated some modular properties of irreducible representations of symmetric groups, where he was the first to introduce the concept of hooks (James, 1978). In the paper "On some modular properties of irreducible representations of a symmetric group" Nakayama (1940) gave a more precise formulation of Murnaghan's formula, by using hooks in Young diagrams that correspond to an irreducible representation (Lewis, 2003). With this, he also gave more concise proof of Murnaghan's formula.

Nakayama's contribution to Murnaghan's formula consequently gave the formula the name "Murnaghan-Nakayama rule", in honour to both mathematicians.

## 1.4 Structure of the Master Thesis

In this Master thesis, we start by introducing the notions and definitions that are needed to understand the Murnaghan Nakayama rule and the proofs around this rule.

First, we look at representation theory on general groups in Section 2. Here we define some of the important notations from representation theory. Where we later specify the definitions from Section 2 to those of symmetric groups in Section 3. In this section, we will look at some special properties of symmetric groups, which are useful for representation theory. Both of these sections have definitions that are based on those in the book by Sagan (2001) and the book by Stanley and Fomin (1999). We refer the reader to these for more details and further explanations.

After that, we will state the Murnaghan Nakayama rule in Section 4. This section also introduces some of the different variants of this rule and looks into some articles that reformulate the Murnaghan-Nakayama rule for different purposes.

Lastly, in Section 5, we will use the algorithm of the Murnaghan-Nakayama rule to solve examples with SageMath. This section will include a code, written in SageMath, which will run the algorithm of the Murnaghan-Nakayama rule. Here we will also compare the algorithm for the Murnaghan-Nakayama rule to some already defined functions in SageMath, to see if the algorithm is efficient or not.

## 2 Representation Theory on General Groups

A group  $G$  is an ordered pair of a set and a binary operation on the set (i.e., addition or multiplication), such that (Grillet, 2007)

- the operation on the set is associative
- there exists an identity element  $\epsilon \in G$
- every element  $x \in G$  has an inverse (or opposite when looking over addition), meaning there is an element  $y \in G$  such that
  - in multiplicative notation;  $xy = yx = \epsilon = 1_G$ , or
  - in additive notation;  $x + y = y + x = \epsilon = 0_G$ .

The set  $G$  is called the *underlying set* of the group. Note that one usually denotes a group and its underlying group by the same letter.

### 2.1 Conjugate Classes of Groups

Let us look at a group  $G$  acting on itself by left multiplication. Then the group has following action

$$g * x = gx \quad \forall g, x \in G.$$

We can also define another action of  $G$  action on itself by defining the action

$$g \bullet x = xg^{-1} \quad \forall g, x \in G.$$

Here  $G$  is action on itself by inverse right multiplication.

Then, for any given  $y \in G$  the following action can be found, by combining both the right inverse multiplication and left multiplication;

$$x \bullet y * x = (x \bullet y) * x = (yx^{-1})x = y.$$

We say that  $G$  acts on itself by conjugation when the group have both left multiplication and inverse right multiplication actions on the group (Loehr, 2011).

A set of all the elements that *conjugate* with some element  $g \in G$ , is called the *conjugacy class of  $g$* . This set is defined as

$$C_g = \{xgx^{-1} : x \in G\}. \tag{1}$$

To understand this concept better, let's look at an example.

► **Example 1.** Let  $G$  be the cyclic group of order 4, meaning the group is generated by one element,  $g \in G$ . Because the group is cyclic, we can write  $G$  as

$$G = \{g, g^2, g^3, g^4 = g^0 = \epsilon\} = \{g, g^2, g^3, \epsilon\} = \{\epsilon, g, g^2, g^3\}.$$

From Equation (1), all the conjugacy classes of the elements in  $G$  are

$$C_\epsilon = \{\epsilon\}, \quad C_g = \{g\}, \quad C_{g^2} = \{g^2\}, \quad C_{g^3} = \{g^3\}.$$

◀

We can also make a set of the elements that are conjugate with some element  $g \in G$ .

**Definition 1.** Let  $G$  be any group. The *centraliser* of  $g \in G$  is defined to be the set of all elements that commute with  $g$ ;

$$Z_g = \{h \in G : hgh^{-1} = g\}.$$

There is a bijection between the cosets of the centraliser  $Z_g$  and the elements of the conjugacy class  $C_g$ , such that

$$|C_g| = \frac{|G|}{|Z_g|}. \tag{2}$$

To easier see what centraliser corresponds to which conjugacy class, we can denote the centraliser as

$$Z_g = Z_{C_g}.$$

► **Example 2.** Let us again look at the same  $G$  as in Example 1, namely the group  $G = \{\epsilon, g, g^2, g^3\}$ . The centralisers of the different elements in  $G$  are then

$$Z_\epsilon = \{\epsilon, g, g^2, g^3\} = Z_g = Z_{g^2} = Z_{g^3} = G$$

From Equation (2), one can easily find the sizes of each  $Z_{C_j}$ , when you know the size of the corresponding conjugacy class  $C_j$ . Since all the conjugacy classes of this group are one dimensional, that is to say, that  $|C_\epsilon| = |C_g| = |C_{g^2}| = |C_{g^3}| = 1$ , then the size of each centraliser of elements in  $G$  must be the same as the size of the whole group  $G$ . Then we know that  $Z_g = G$  for all  $g \in G$ .

◀



## 2.2 Representations of Groups

### 2.2.1 Matrix Representation

Instead of looking at groups with an underlying set, we can think of them as a group with underlying *matrices*. In other words, let us look at a group represented as matrices.

First, let us define the general linear group.

**Definition 2.** Let  $\text{Mat}_d$  be the set of all  $d \times d$  matrices with entries in  $\mathbb{C}$  (also called a *fully complex matrix algebra of degree  $d$* ). The *complex general linear group of degree  $d$* ,  $\text{GL}_d$ , is the group of all  $X = (x_{i,j})_{d \times d} \in \text{Mat}_d$  that are invertible with respect to multiplication.

From this definition, we define a matrix representation of a group as follows;

**Definition 3.** A *matrix representation of a group  $G$*  is a group homomorphism

$$X : G \rightarrow \text{GL}_d.$$

Equivalently, to each  $g \in G$  there is an assigned  $X(g) \in \text{Mat}_d$ , such that

1.  $X(\epsilon) = I$ , the identity matrix for the identity  $\epsilon \in G$ ,
2.  $X(gh) = X(g)X(h)$ , for all  $g, h \in G$ .

The parameter  $d$  is called the *degree*, or *dimension*, of the representation, and is denoted as  $\deg X = d$ .

From the conditions in Definition 3,  $X(g^{-1}) = X(g)^{-1}$ . These matrices are therefore in some group, namely the general linear group  $\text{GL}_d$ . We now have a valid way to represent the groups as matrices.

The simplest representations of a group are those that are of degree 1, where the group is represented as a  $1 \times 1$  matrix. One of these representations is the trivial representation.

**Definition 4.** The *trivial representation* of a group  $G$  is the representation that sends all elements  $g \in G$  to 1.

Every group has a trivial representation. Therefore, we know at least one representation for any group.

► **Example 3.** Let  $G$  be the cyclic group of order 4.

$$G = \{\epsilon, g, g^2, g^3\},$$

where  $g$  is the (cyclic) generator of the group.

A matrix representation of four dimensions could then be

$$\begin{aligned}
 g^0 &\mapsto \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, & g^1 &\mapsto \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \\
 g^2 &\mapsto \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, & g^3 &\mapsto \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.
 \end{aligned}$$

As we can see with these matrices, they express what happens when each element acts on different elements in the group. ◀

### 2.2.2 Modules

Groups can also be looked at with more general representations. Since matrices correspond to linear transformations, we can also think of matrix representations the same way. This is the idea behind  $G$ -modules.

When looking at a (finite) vector space  $V$  over the complex numbers, define the *general linear group of  $V$* ,  $\text{GL}(V)$ . This is the set of all invertible linear transformations of  $V$  to itself. If  $\dim V = d$ , then  $\text{GL}(V)$  and  $\text{GL}_d$  are isomorphic as groups.

**Definition 5.** Let  $V$  be a vector space and  $G$  be a group. We say that  $V$  is a  $G$ -module if there is a group homomorphism

$$\varphi : G \rightarrow \text{GL}(V).$$

Equivalently, if  $V$  is a  $G$ -module then  $V$  has the properties:

1.  $g\mathbf{v} \in V$ ,
2.  $g(c\mathbf{v} + d\mathbf{w}) = c(g\mathbf{v}) + d(g\mathbf{w})$ ,
3.  $(gh)\mathbf{v} = g(h\mathbf{v})$ ,
4.  $\epsilon\mathbf{v} = \mathbf{v}$ , for the identity  $\epsilon \in G$

for all  $g, h \in G$ ,  $\mathbf{v}, \mathbf{w} \in V$  and  $c, d \in \mathbb{C}$ .

When the group  $G$  is given, we refer to the  $G$ -module  $V$  as a module.

### 2.2.3 Decomposition of Representations

When studying large objects, one would like these large structures to be broken into smaller pieces. There must be a method to "decompose" a representation into smaller ones.

**Definition 6.** Let  $V$  be a  $G$ -module. A submodule of  $V$  is a subspace  $W$ , where this subspace is closed under actions of  $G$ , such that

$$gw \in W, \quad w \in W, \forall g \in G.$$

Here  $W$  is said to be a  $G$ -invariant subspace. If  $W$  is a submodule of  $V$ , it is denoted by  $W \leq V$ .

Any module  $V$  has both  $W = V$  and  $W = \{0\}$  as submodules, these two submodules are called *trivial submodules*. Any other submodule is then called *nontrivial submodule*.

**Definition 7.** A  $G$ -module  $V$  is reducible if it contains a nontrivial submodule  $W$ . Otherwise, we say that the module  $V$  is irreducible.

From this definition, it is clear that any representation of degree 1 is irreducible. Looking at every subspace to find out which are submodules is time-consuming, so we would like to avoid this. Instead, one can look at matrices.

When looking at matrices, we could bring the matrices that are reducible  $G$ -modules to the block diagonal form

$$X(g) = \begin{pmatrix} A(g) & 0 \\ 0 & B(g) \end{pmatrix}$$

for all  $g \in G$ . This can be done by the notation of direct sums.

**Definition 8.** If  $X$  is a matrix, then  $X$  is said to be the *direct sum* of matrices  $A$  and  $B$ , denoted as  $X = A \oplus B$ , if  $X$  has the block diagonal form

$$X = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$$

Let's redefine this definition to indicate how direct sums affect  $G$ -modules.

**Definition 9.** Let  $V$  be a vector space with  $U$  and  $W$  as subspaces. We say that  $V$  is the (*internal*) *direct sum* of  $U$  and  $W$ , written  $V = U \oplus W$ , every  $\mathbf{v} \in V$  can be written as a unique sum

$$\mathbf{v} = \mathbf{u} + \mathbf{w}, \quad \mathbf{u} \in U, \mathbf{w} \in W.$$

If  $V$  is a  $G$ -module, where  $U$  and  $W$  are  $G$ -submodules, then we say that  $U$  and  $W$  are *complements* of each other.

Note that, for a vector space with a subspace  $W$ , the *orthogonal complement* of  $W$  can be found using the formula

$$W^\perp = \{\mathbf{v} \in V : \langle \mathbf{v}, \mathbf{w} \rangle = 0, \forall \mathbf{w} \in W\}. \quad (3)$$

From these two subspaces of  $V$ , the whole space can be defined as the direct sum  $V = W \oplus W^\perp$ . When  $W \leq V$  and the inner product is  $G$ -invariant then the following proposition is also valid;

**Proposition 1.** *Let  $V$  be a  $G$ -module,  $W$  be its submodule, and  $\langle \cdot, \cdot \rangle$  be an inner product that is invariant under the actions of  $G$ . Then  $W^\perp$  is also a  $G$ -submodule.*

With this, we obtain a key theorem in group representation theory that holds for any *finite* group. Specifically, Maschke's theorem.

**Theorem 1** (Maschke's Theorem). *Let  $G$  be a finite group, and let  $V$  be a nonzero module. Then  $V$  can be written as*

$$V = W^{(1)} \oplus W^{(2)} \oplus \dots \oplus W^{(k)} = \bigoplus_{i=1}^k W^{(i)}$$

where each  $W^{(i)}$  is an irreducible  $G$ -submodule of  $V$ .

The matrix version of Theorem 1, is given as the corollary;

**Corollary 1.** Let  $G$  be a finite group, and  $X$  be a matrix representation of  $G$  with dimension  $d > 0$ . Then, there is a fixed matrix  $T$ , such that every matrix  $X(g)$ , for  $g \in G$ , is of the form

$$TX(g)T^{-1} = \begin{pmatrix} X^{(1)}(g) & 0 & \dots & 0 \\ 0 & X^{(2)}(g) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & X^{(k)}(g) \end{pmatrix} = \bigoplus_{i=1}^k X^{(i)}(g).$$

Each  $X^{(i)}$  is an irreducible matrix representation of  $G$ .

Theorem 1 is true for vector spaces over any type of field, as long as the field has characteristics equal to either zero or prime to  $|G|$ . The finiteness of a group is central to the theorem, and can therefore not be removed from consideration.

## 2.3 Characters of Groups

Most of the information that is contained in a representation can be found by simply taking the trace of the corresponding matrices.

**Definition 10.** Let  $X(g)$  be a matrix representation of the group  $G$ , where  $g \in G$ . Then the *character* of this representation,  $\chi$ , is the map

$$G \xrightarrow{\text{tr}(X)} \mathbb{C}$$

defined by

$$\chi(g) = \text{tr}(X(g)) \tag{4}$$

If  $V$  is a  $G$ -module, then the *character* of  $V$  is the character of a matrix representation  $X$  corresponding to the module  $V$ .

We need to make sure that this character is well-defined as there are many matrix representations corresponding to a single  $G$ -module  $V$ .

Assume both  $X$  and  $Y$  are representations that correspond to the same module. Then, for some fixed matrix  $T$ ,  $Y$  can be rewritten to be  $Y = TXT^{-1}$ . Then, for all  $g \in G$ , the character value of  $Y$  can be found to be

$$\text{tr}(Y(g)) = \text{tr}(TX(g)T^{-1}) = \text{tr}(T)\text{tr}(X(g))\text{tr}(T^{-1}) = \text{tr}(X(g)) = \chi(g),$$

due to trace being invariant under conjugation. From this calculation, the two representations have the same characters. This implies that characters are indeed well-defined.

The terminology for representations can be adapted to their corresponding characters, without much change. If  $X$  is a representation with character  $\chi$ , then the character  $\chi$  is irreducible when  $X$  is irreducible, etc.

When looking at a one-dimensional representation,  $X(g)$ , the character  $\chi(g)$  is the sole entry of  $X(g)$ , for  $g \in G$ . Characters like this are referred to as *linear characters*. The characters of the trivial representation  $X(g) = (1)$  is just  $\chi(\epsilon) = 1$ , where this character is called the *trivial character*.

► **Example 4.** Going back to Example 3, where the fourth-dimensional representation

of the cyclic group of order 4 was

$$g^0 \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad g^1 \mapsto \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

$$g^2 \mapsto \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \quad g^3 \mapsto \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The characters of this representation are then

$$\begin{aligned} \chi(g^0) &= \text{tr}(X(g^0)) = 4, & \chi(g^1) &= \text{tr}(X(g^1)) = 0, \\ \chi(g^2) &= \text{tr}(X(g^2)) = 0, & \chi(g^3) &= \text{tr}(X(g^3)) = 0 \end{aligned}$$

◀

The characters from the example is a quite special kinds of characters;

**Definition 11.** Let  $G = \{g_1, g_2, \dots, g_n\}$ . The *regular representation* is defined to be  $X_{reg}(\epsilon) = I_n$ , such that the *regular character* is

$$\chi^{reg} = \begin{cases} |G| & \text{if } g = \epsilon, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

One can further use the regular representation to find all the irreducible representations of a finite group  $G$ . From the book by Fulton and Harris (2004), the following corollary was introduced;

**Corollary 2.** Any irreducible representation  $V$  of a group  $G$  appears in the regular representation  $\dim V$  times. The regular character can be written as the following sum;

$$\chi_{reg} = \sum_V \dim(V) \cdot \chi_V \quad (6)$$

The character of some group representation has the following properties.

**Proposition 2.** Let  $X \in \text{Mat}_d$  be a matrix representation of a group  $G$  on a  $\mathbb{K}$ -vector space, with character  $\chi$ . We then have the following properties of  $\chi$ .

1.  $\chi(\epsilon) = d = \dim X$ , where  $\epsilon \in G$  is the identity.

2. If  $C$  is a conjugacy class of  $G$ , then

$$g, h, \in C \implies \chi(g) = \chi(h).$$

3. If  $Y$  is another representation of  $G$  with character  $\psi$ , then

$$X \cong Y \implies \chi(g) = \psi(g)$$

for all  $g \in G$ .

4. If  $X \cong m_1 X^{(1)} \oplus m_2 X^{(2)} \oplus \cdots \oplus m_k X^{(k)}$ , where  $X^{(i)}$  are pairwise inequivalent irreducibles with characters  $\chi^{(i)}$ , then we can write the character  $\chi$  of  $X$  as

$$\chi = m_1 \chi^{(1)} + m_2 \chi^{(2)} + \cdots + m_k \chi^{(k)}.$$

5. If  $Z$  is a representation for the group  $H$  with character  $\phi$ , and  $X \otimes Z$  is a representation of  $G \times H$  with the character  $\chi \otimes \phi$ . Then

$$(\chi \otimes \phi)(g, h) = \chi(g)\phi(h)$$

for all  $(g, h) \in G \times H$ .

*Proof.*

1. The representation on the identity element of the group sends all elements back to itself, such that  $X(\epsilon) = I_d$ . From Definition 10, the character of this representation is then

$$\chi(\epsilon) = \text{tr}(I_d) = \sum_{i=1}^d 1 = d = \dim X.$$

2. From the definition of conjugacy classes  $g = khk^{-1}$  for  $g, h \in C$  and some  $k \in G$ . The character of  $g$  is then

$$\begin{aligned} \chi(g) &= \text{tr}(X(g)) = \text{tr}(X(khk^{-1})) = \text{tr}(X(k)X(h)X(k)^{-1}) \\ &= \text{tr}(X(k)) \text{tr}(X(h)) \text{tr}(X(k)^{-1}) = \text{tr}(X(h)) = \chi(h). \end{aligned}$$

So  $\chi(g) = \chi(h)$  for any  $g, h \in C$ .

3. This assertion just says that equivalent representations have the same character. This has already been proven. But, for the sake of this proof, it can be repeated;

If  $X$  and  $Y$  correspond to the same  $G$ -Module, then  $Y = TXT^{-1}$ , for some fixed  $T$ . Then, for all  $g \in G$ ,

$$\text{tr}(Y(g)) = \text{tr}(TX(g)T^{-1}) = \text{tr}(X(g)) = \chi(g),$$

as seen before.

4. The trace of a direct sum is the sum of traces,  $\text{tr}(A \oplus B) = \text{tr}(A) + \text{tr}(B)$ . From this, the character is

$$\chi = \text{tr}(X) = \text{tr}\left(\bigoplus_{i=1}^k m_i X^{(i)}\right) = \sum_{i=1}^k m_i \text{tr}\left(X^{(i)}\right) = \sum_{i=1}^k m_i \chi^{(i)}.$$

5. Let  $A \in \text{Mat}_d$  and  $B \in \text{Mat}_e$ . The direct product  $A \otimes B$  is the matrix of degree  $(d \times e)$ , such that each element  $(a_{ij})$  in  $A$  is multiplied with the matrix  $B$ . The trace of this direct product is then

$$\text{tr}(A \otimes B) = \text{tr}(a_{i,j}B) = \sum_i a_{i,i} \text{tr}(B) = \text{tr}(A) \text{tr}(B).$$

Now, we must find the tensor product representation.

**Definition 12.** Let  $X$  and  $Y$  be the matrix representations of  $G$  and  $H$ , respectively. Then the *tensor product representation*,  $X \otimes Y$ , assigns the following matrix to each  $(g, h) \in G \times H$

$$(X \otimes Y)(g, h) = X(g) \otimes Y(h).$$

By using the rule of traces on direct products and the definition of tensor product representation, the character  $\chi \otimes \phi$  can be expressed as

$$(\chi \otimes \psi)(g, h) = \text{tr}(X(g) \otimes Y(h)) = \text{tr}X(g) \text{tr}Y(h) = \chi(g)\psi(h).$$

□

Note that the converse of the third property is also true, but only if we are looking over the complex vector space, i.e. if and only if  $\chi : G \rightarrow \mathbb{C}$ . So, if two representations have the same character, then they must be equivalent to each other when looking over complex numbers. This property alone is very important and is used as the main motivation when it comes to group characters. The proof of this statement can be found in the book written by Sagan, p. 38

The second property of Proposition 2, says that characters are constant on conjugacy classes. Functions with this property are given the name "class functions".

**Definition 13.** A *class function* on a group  $G$  is a mapping  $f : G \rightarrow \mathbb{C}$  such that  $f(g) = f(h)$  whenever  $g$  and  $h$  are in the same conjugacy class. The set of all class functions on  $G$  is denoted by  $R(G)$ .



Since  $R(G)$  has a natural basis that consists of the functions that have value 1 on a given conjugacy class and 0 elsewhere, therefore

$$\dim(R(G)) = \#\{C_g : g \in G\}.$$

Because of the property of character on conjugacy classes from Proposition 2,  $\chi_C$  can be defined as the corresponding character value of the conjugacy class  $C$ . The characters can therefore be denoted with

$$\chi_C = \chi(g), \quad \forall g \in C.$$

### 2.3.1 Character Table

To easier recognise what characters correspond to which conjugacy class in a group  $G$ , we can place them in a table. This kind of table is called the "character table" of  $G$ .

**Definition 14.** Let  $G$  be a group, where its *character table* is an array. The rows of the array are indexed by the inequivalent irreducible characters of  $G$ , while the columns are indexed by their conjugacy classes. From this array, we can read that the table entry in row  $\chi$  and column  $C$  is the character value  $\chi_C$

	...	$C$	...
⋮		⋮	
$\chi$	...	$\chi_C$	
⋮			

Usually, the first row in this array is the trivial character, and the first column corresponds to the class of identity,  $C = \{\epsilon\}$ . From the first property in Proposition 2, the first column notes the degree of the representations corresponding to the different characters. However, this is not always the case.

Another way to present the character table is as follows;

**Definition 15.** Let  $C_i$  be the conjugacy class of  $g_i \in G$ . Assuming that there are  $k$  conjugacy classes, then the character table has the following setup:

	$C_1$	...	$C_k$
$\chi_1$	$\chi_1(g_1)$	...	$\chi_1(g_k)$
⋮	⋮	⋱	⋮
$\chi_n$	$\chi_n(g_1)$	...	$\chi_n(g_k)$
#	$ C_1 $	...	$ C_k $

Where, in the last row, we denote the size of each conjugacy class.

When writing the sizes of the conjugacy classes, the inner product of the characters is easier to compute. We will come back to the inner product after the following example.

► **Example 5.** Let us look at the cyclic group  $G = \{g^0 = \epsilon, g, g^2, g^3\}$  of order 4. Here the following conjugacy classes are,

$$C_1 = \{\epsilon\}, \quad C_2 = \{g\}, \quad C_3 = \{g^2\}, \quad C_4 = \{g^3\}.$$

We can see that there are  $\dim(R(G)) = 4$  conjugacy classes, meaning that we need to find four characters.

We already know that the group must include the trivial character  $\chi_1 = (1, 1, 1, 1)$ . From Corollary 2, the last three characters can be found by using the regular representation from Definition 11. The regular character is defined to be

$$\chi_{reg} = \sum_{i=1}^{\dim(R(G))} \chi_i(g_1)\chi_i,$$

where  $g_1 = \epsilon$  in this group.

From this sum, we get the following equations;

$$\begin{aligned} 1 \cdot 1 + \chi_2(\epsilon) \cdot \chi_2(\epsilon) + \chi_3(\epsilon) \cdot \chi_3(\epsilon) + \chi_4(\epsilon) \cdot \chi_4(\epsilon) &= 4 \\ 1 \cdot 1 + \chi_2(\epsilon) \cdot \chi_2(g) + \chi_3(\epsilon) \cdot \chi_3(g) + \chi_4(\epsilon) \cdot \chi_4(g) &= 0 \\ 1 \cdot 1 + \chi_2(\epsilon) \cdot \chi_2(g^2) + \chi_3(\epsilon) \cdot \chi_3(g^2) + \chi_4(\epsilon) \cdot \chi_4(g^2) &= 0 \\ 1 \cdot 1 + \chi_2(\epsilon) \cdot \chi_2(g^3) + \chi_3(\epsilon) \cdot \chi_3(g^3) + \chi_4(\epsilon) \cdot \chi_4(g^3) &= 0 \end{aligned}$$

From the first equation, the only valid solution is  $\chi_2(\epsilon) = \chi_3(\epsilon) = \chi_4(\epsilon) = 1$ . This is because the degree of an irreducible representation must be a positive integer, and not equal to zero.

Since all the representations are of dimension one, we can assume that each representation is  $X_i(g) = (c)$ , for  $c \in \mathbb{C}$  and  $i = 1, 2, 3, 4$ . Then the matrix for every element of  $G$  is determined, since  $X(g^k) = (c^k)$  from the second property of Definition 3. By the definition's first property;

$$(c^4) = X(g^4) = X(\epsilon) = (1)$$

such that  $c$  must be the 4th root of unity. The four fourth roots of unity are  $1, i, -1, -i$ . We let the four corresponding representations be written as  $X_1, X_2, X_3, X_4$ , respectively. Indexing the characters with the same number as the representation that they correspond to. Therefore,  $\chi_1 = (1, 1, 1, 1)$  corresponds to the representation  $X_1$ ,  $\chi_2 = (1, i, -1, -i)$  corresponds to  $X_2$ , and so on.

We now have all the information needed to write out the character table. Using the same notations as in Definition 15, the character table is then

	$C_1$	$C_2$	$C_3$	$C_4$
$\chi_1$	1	1	1	1
$\chi_2$	1	$i$	-1	$-i$
$\chi_3$	1	-1	1	-1
$\chi_4$	1	$-i$	$-i$	$i$
#	1	1	1	1

◀

Every character has a special relation to other characters. When we are looking at the characters  $\chi, \phi$  over the same group,  $G$ , then the inner product between them is defined as

$$\langle \chi, \phi \rangle = \frac{1}{|G|} \sum_{g \in G} \chi(g) \phi(g^{-1}). \quad (7)$$

Since characters are constant on conjugacy classes, the inner product between two characters can be rewritten as

$$\langle \chi, \phi \rangle = \frac{1}{|G|} \sum_{g \in G} |C_g| \chi(g) \phi(g^{-1}), \quad (8)$$

where  $C_g$  is the conjugacy class corresponding to  $g \in G$ .

When looking over the complex vector space, the character  $\phi(g^{-1})$  is equal to the complex conjugate of  $\phi(g)$ ,  $\overline{\phi(g)}$ .

For irreducible characters, the inner product is quite easy to find.

**Theorem 2.** *Let  $\chi$  and  $\phi$  be irreducible characters of a group  $G$ . Then*

$$\langle \chi, \phi \rangle = \delta_{\chi, \phi} = \begin{cases} 1, & \chi = \phi \\ 0, & \chi \neq \phi \end{cases}.$$

From this inner product, unknown characters can be found if we already know some of them. Let's use inner products to find unknown characters in the following example.

► **Example 6.** This example is taken from a question asked by user Ben (2022) on Mathematics Stack Exchange.

Here, the group has the order  $|G| = 168$ . This group has  $\dim(R(G)) = 6$  conjugacy classes of order 1, 21, 42, 56, 24, and 24. We assume that we are looking over the complex vector space.

We are given three characters for this group,  $A = (14, 2, 0, -1, 0, 0)$ ,  $B = (15, -1, -1, 0, 1, 1)$ , and  $C = (16, 0, 0, -2, 2, 2)$ . However, any group always has a trivial representation, such that the trivial character is included in the list of known characters. We can thus list the known characters in a temporary character table:

	$C_1$	$C_2$	$C_3$	$C_4$	$C_5$	$C_6$
$\chi_1$	1	1	1	1	1	1
$A$	14	2	0	-1	0	0
$B$	15	-1	-1	0	1	1
$C$	16	0	0	-2	2	2
$\#$	1	21	42	56	24	24

The conjugacy class of order 1 is the conjugacy class that corresponds to the set including only  $\epsilon$ . We thus know which column we need to look at to see the dimension of the representations, i.e., the first column.

By taking their inner product, we can see how the different characters relate to each other.

$$\begin{aligned}
 \langle A, A \rangle &= 2 & \langle B, B \rangle &= 2 & \langle C, C \rangle &= 4 \\
 \langle \chi_1, A \rangle &= 15 & \langle \chi_1, B \rangle &= 15 & \langle \chi_1, C \rangle &= 18 \\
 \langle A, B \rangle &= 1 & \langle B, C \rangle &= 2 & \langle A, C \rangle &= 2
 \end{aligned}$$

Since  $\langle A, A \rangle \neq \langle B, B \rangle \neq \langle C, C \rangle \neq 1$ , none of them is irreducible, so they must be sums of irreducible characters.

Since  $\langle C, C \rangle = 4$ ,  $C$  must either be a sum of four irreducible characters or include two of the same irreducible characters. If  $C$  was a sum of four characters then  $C - A$  must be a character of degree 1. This means that  $C - A$  would be an irreducible character. It would also have to be a sum of two irreducible characters from the inner product  $\langle A, C \rangle = 2$ . An irreducible character cannot be a sum of two different irreducible characters, such that  $C$  must include two of the same character;

$$C = 2 \cdot \chi_2 = 2 \cdot (8, 0, 0, -1, 1, 1).$$

Since  $\langle B, C \rangle = \langle A, C \rangle = 2$ , we know that  $\chi_2$  must be one of the two irreducible characters

in both  $A$  and  $B$ . Let  $A = \chi_2 + \chi_3$  and  $B = \chi_2 + \chi_4$ , the irreducible characters are the following.

	$C_1$	$C_2$	$C_3$	$C_4$	$C_5$	$C_6$
$\chi_1$	1	1	1	1	1	1
$\chi_2$	8	0	0	-1	1	1
$\chi_3$	6	6	0	0	-1	-1
$\chi_4$	7	-1	-1	1	0	0
#	1	21	42	56	24	24

Keep in mind that there exists the same amount of irreducible characters as conjugacy classes, so we are still missing two irreducible characters,  $\chi_5$  and  $\chi_6$ .

To find the character values of the last two characters, use the regular character of the group. From Corollary 2, the character associated with the regular character is

$$\chi^{reg}(G) = \sum_{i=1}^{\dim(R(G))} \chi_i(1) \cdot \chi_i = (|G|, 0, 0, 0, 0, 0).$$

From this sum, we get the following equation

$$\begin{aligned} \sum_{i=1}^6 \chi_i(1) \cdot \chi_i &= (168, 0, 0, 0, 0, 0) \\ \chi_1(1) \cdot \chi_1 + \chi_2(1) \cdot \chi_2 + \chi_3(1) \cdot \chi_3 \\ + \chi_4(1) \cdot \chi_4 + \chi_5(1) \cdot \chi_5 + \chi_6(1) \cdot \chi_6 &= (168, 0, 0, 0, 0, 0) \\ \chi_5(1) \cdot \chi_5 + \chi_6(1) \cdot \chi_6 &= (18, -6, 6, 0, -3, -3). \end{aligned}$$

However, the only valid dimension for both  $\chi_5$  and  $\chi_6$  is 3, since the dimension must be a positive integer. This means that the sum of the two characters is

$$\chi_5 + \chi_6 = (6, -2, 2, 0, -1, -1).$$

Note that the last two conjugacy classes have centralisers of order  $\frac{|G|}{|C_5|} = \frac{|G|}{|C_6|} = 7$ . Now, all subgroups of prime order are conjugate (Constantine, 1998, Theorem (Sylow)), the elements in these conjugacy classes must be  $x$  and  $x^{-1}$ . Therefore, the last two values are complex conjugates of each other. Since all other classes are real, it implies that  $\chi_5(g) = \chi_6(g)$  for all other  $g$ .

The characters are therefore

$$\begin{aligned}\chi_5 &= (3, -1, 1, 0, a, a^{-1}) = (3, -1, 1, 0, a, \bar{a}), \\ \chi_6 &= (3, -1, 1, 0, b, b^{-1}) = (3, -1, 1, 0, b, \bar{b}).\end{aligned}$$

Since we are looking over  $\mathbb{C}$ , then the character  $\phi(g^{-1}) = \overline{\phi(g)}$ .

From the equation;  $\chi_5 + \chi_6 = (6, -2, 2, 0, -1, -1)$ , we have to solve for

$$\begin{aligned}a + b &= -1 \\ \bar{a} + \bar{b} &= -1.\end{aligned}$$

By solving this linear system, the character value of these conjugacy classes is  $a = -1 - i\sqrt{2}$  and  $b = i\sqrt{2}$ . The final character table looks as follows

	$C_1$	$C_2$	$C_3$	$C_4$	$C_5$	$C_6$
$\chi_1$	1	1	1	1	1	1
$\chi_2$	8	0	0	-1	1	1
$\chi_3$	6	6	0	0	-1	-1
$\chi_4$	7	-1	-1	1	0	0
$\chi_5$	3	-1	1	0	$-1 - i\sqrt{2}$	$1 + i\sqrt{2}$
$\chi_6$	3	-1	1	0	$i\sqrt{2}$	$-i\sqrt{2}$
#	1	21	42	56	24	24



This example mostly used the inner product and the knowledge of regular characters from Corollary 2 to find the unknown characters. This emphasises the importance of these properties in character theory.

### 3 Representation Theory on Symmetric Groups

When studying representations, a particularly nice theory arises when focusing on the symmetric group. Indeed in this case the beautiful connection between combinatorics and group theory makes this situation easier to understand.

Let  $S_n$  be the symmetric group over  $n$  elements. The elements  $\pi \in S_n$  are called *permutations*, and they indicate the bijection of the set  $[n] = \{1, 2, \dots, n\}$ .

**Definition 16.** For a *permutation*  $\pi \in S_n$  three different notations can be used to describe this element;

- the *Two-line notation*; which is the array

$$\pi = \begin{pmatrix} 1 & 2 & \cdots & n \\ \pi(1) & \pi(2) & \cdots & \pi(n) \end{pmatrix}$$

noting where each element of  $[n]$  is mapped to.

- the *One-line notation*; it is similar to the two-line notation, but the top line is excluded

$$\pi = (\pi(1) \ \pi(2) \ \cdots \ \pi(n))$$

- The *cyclic notation*; given  $i \in [n]$ , after using the same permutation  $\pi$  on the same integer  $i$ , we will eventually end right back up at  $i$  after  $p$  permutations,  $\pi^p(i)$ . We denote this as  $(i, \pi(i), \pi^2(i), \dots, \pi^{p-1}(i))$ . Now, when there are multiple cycles of  $[n]$ , we note it as  $(i, \pi(i), \dots)(j, \pi(j), \dots)(k, \pi(k), \dots) \cdots$ .

Usually, the permutation is written in the cyclic notation, since it is the shorter of the notations. In this notation, one can choose to not include those cycles that map the elements back to themselves, i.e., the trivial cycles (also called the fixed points). This makes the cyclic notation shorter than the others. The order of the cycles in the cyclic notation is not very important. Usually, we write them from the largest cycle to the smallest or order them after the smallest numbers in their cycles.

► **Example 7.** Let's look at the map

$$\begin{aligned} \pi : \quad & 1 \mapsto 2 \\ & 2 \mapsto 4 \\ & 3 \mapsto 5 \\ & 4 \mapsto 1 \\ & 5 \mapsto 3 \\ & 6 \mapsto 6. \end{aligned}$$

This permutation can then be written as

$$\begin{aligned}\pi &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 1 & 3 & 6 \end{pmatrix} = (2 \ 4 \ 5 \ 1 \ 3 \ 6) \\ &= (124)(35)(6) = (124)(35)\end{aligned}$$

◀

When multiplying permutations we do so by going from right to left. If we are looking at the multiplication  $\sigma\pi$ , then the bijection is obtained by first applying  $\pi$  and later  $\sigma$ . To make the notation of the multiplication more visually understanding, we can write the multiplication as follows; First, write all integers in the permutations. In the row just below, write where the integers are mapped to by  $\pi$  (the rightmost permutation). Then, in the last row, write their corresponding numbers, after permuting with respect to  $\sigma$ . It will look like this;

$$\sigma\pi = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ \pi(1) & \pi(2) & \pi(3) & \cdots & \pi(n) \\ \sigma(\pi(1)) & \sigma(\pi(2)) & \sigma(\pi(3)) & \cdots & \sigma(\pi(n)) \end{pmatrix}$$

Now, delete the middle row. The remainder is the permutation  $\sigma\pi$  in the two-line notation;

$$\sigma\pi = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ \sigma(\pi(1)) & \sigma(\pi(2)) & \sigma(\pi(3)) & \cdots & \sigma(\pi(n)) \end{pmatrix} \quad (9)$$

From cycles we can define the  $k$ -cycle, or more precisely the *cycle of length  $k$* , which is a cycle that contains  $k$  elements. The *cycle type*, or simply the *type*, of  $\pi$  is an expression

$$\text{type}(\pi) = (1^{m_1}, 2^{m_2}, \dots, n^{m_n}),$$

where  $m_k$  is the number of cycles of length  $k$  in  $\pi$ . Then, the *type* of  $\pi$  expresses the amount of  $k$ -cycles there are in  $\pi$ , where a 1-cycle of  $\pi$  is a fixed point

Another way to express the cyclic types is by writing them as partitions.

**Definition 17.** A *partition* is a sequence of positive integers  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_k)$  where its parts,  $\lambda_i$ , are weakly decreasing

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k.$$

All of its parts sums up to an integer

$$|\lambda| = \sum_{1 \leq i \leq k} \lambda_i = n.$$

We denote the relation between the integer  $n$  and the partition  $\lambda$  as  $\lambda \vdash n$ . The number of parts of  $\lambda$  is denoted as  $\ell(\lambda)$ .



The set of all partitions of  $n$  is denoted as  $Par(n)$ , where the number of partitions of  $n$  is  $|Par(n)| = p(n)$ .

The cycle type

$$\text{type}(\pi) = (1^{m_1}, 2^{m_2}, \dots, n^{m_n})$$

can be noted as the partition  $\lambda = (n^{m_n}, \dots, 2^{m_2}, 1^{m_1})$ . Remember that the parts of the partition need to be in a weakly decreasing order. The partition can also be written without the exponents, where we list the number  $i$   $m_i$  times in the partition.

A *composition* is similar to a partition, except the sequence is not necessarily weakly decreasing. This means that  $(4, 3)$  is both a partition and a composition, while  $(3, 2, 4)$  is a composition and not a partition.

### 3.1 Conjugacy Classes of Symmetric Groups

Something special with the symmetric group  $S_n$  is their conjugacy classes;

► **Example 8.** Let  $G$  be the symmetric group of 3 elements,  $S_3$ . The conjugacy classes for the elements of this group are then

$$\begin{aligned} G * \text{id} &= \{\text{id}\} \\ G * (12) &= G * (13) = G * (23) = \{(12), (13), (23)\} \\ G * (123) &= G * (132) = \{(123), (132)\} \end{aligned}$$

◀

From this example, we get an idea that the conjugacy classes in  $S_3$  may consist of all the permutations that are of the same cycle type. We want to see if this is true for any symmetric group  $S_n$ . Let's first look at the conjugacy of symmetric groups.

**Theorem 3** ((Loehr, 2011)). *For  $\pi, \rho \in S_n$ , the permutation  $\rho\pi\rho^{-1}$  can be found by applying  $\rho$  to each entry in all of the disjoint cycle decomposition of  $\pi$ . If*

$$\pi = (i_1, i_2, \dots)(j_1, j_2, \dots)(k_1, k_2, \dots) \dots,$$

then we get the following permutation

$$\rho\pi\rho^{-1} = (\rho(i_1), \rho(i_2), \dots), (\rho(j_1), \rho(j_2), \dots), (\rho(k_1), \rho(k_2), \dots) \dots.$$

The cycle type is therefore preserved,  $\text{type}(\rho\pi\rho^{-1}) = \text{type}(\pi)$ .

Let's prove this theorem as an example.

► **Example 9.** Let us find  $\rho\pi\rho^{-1}$  for the permutations

$$\rho = (123)(45)(6)(7)(8)$$

$$\pi = (1234)(56)(78)$$

To see if the theorem is valid, let us first look at the permutation  $\pi\rho^{-1}$ .

We use the same notation as earlier for the multiplication of permutations. The multiplication can therefore be written as

$$\pi\rho^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 1 & 2 & 5 & 4 & 6 & 7 & 8 \\ 4 & 2 & 3 & 6 & 1 & 5 & 8 & 7 \end{pmatrix} = (1465)(2)(3)(78) = (1465)(78).$$

Further, the permutation of  $\rho(\pi\rho^{-1})$  is then

$$\rho(\pi\rho^{-1}) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 2 & 3 & 6 & 1 & 5 & 8 & 7 \\ 5 & 3 & 1 & 6 & 2 & 4 & 8 & 7 \end{pmatrix} = (1523)(46)(78).$$

Where the result is  $\rho\pi\rho^{-1} = (1523)(46)(78)$ .

Using Theorem 3, we get

$$\begin{aligned} \rho\pi\rho^{-1} &= (\rho(1), \rho(2), \rho(3), \rho(4)) (\rho(5), \rho(6)) (\rho(7), \rho(8)) \\ &= (2315)(46)(78) = (1523)(46)(78). \end{aligned}$$

Both methods give the same result. However, with the theorem, there are fewer calculations to worry about. ◀

Using Theorem 3, we derive the following theorem for the conjugacy classes of symmetric groups.

**Theorem 4** ((Loehr, 2011)). *The conjugacy class of  $\pi \in S_n$  consists of all  $\sigma \in S_n$  of the same type as  $\pi$ , such that  $\text{type}(\sigma) = \text{type}(\pi)$ . The number of conjugacy classes is equal to the number of integer partitions of  $n$ .*

*Proof.* For a fixed  $\pi \in S_n$ , let  $C_\pi = \{\rho\pi\rho^{-1} : \rho \in S_n\}$  be the conjugacy class of  $\pi$ , and let  $H$  be the set of all  $\sigma \in S_n$  that has the same cycle type as  $\pi$ ,  $H = \{\sigma \in S_n : \text{type}(\sigma) = \text{type}(\pi)\}$ .

Using the theorem of conjugacy in  $S_n$  (Theorem 3), we can observe that  $C_\pi \subseteq H$ . Now we just need to prove the opposite inclusion.

Let  $\sigma \in S_n$  be of the same cycle type as  $\pi$ . We can then use the following algorithm to find a  $\rho \in S_n$  such that  $\sigma = \rho\pi\rho^{-1}$ ; Start by writing down any complete cycle decomposition of  $\pi$  (also including the 1-cycles), and write the longer cycles before the shorter ones. Below this, write down a complete cycle decomposition of  $\sigma$ , in the same way as we did with  $\pi$ . Now, erase all parentheses and regard the resulting array as the permutation of  $\rho$ , written in the two-line form. This constructed  $\rho$  gives the correct relation,  $\rho\pi\rho^{-1} = \sigma$ .

However, this  $\rho$  is not unique. Other  $\rho$ 's could have been obtained, which also satisfy the same relation by starting with a different complete cycle decomposition of either  $\pi$  or  $\sigma$ , or both.

The last statement of the theorem follows from the fact that the possible cycle types of permutations of  $n$  objects are exactly the integer partition of  $n$ .  $\square$

From Theorem 4, two permutations are in the same conjugacy class if and only if they are of the same cycle type, or if their partitions are equal. There is therefore a one-to-one correspondence between conjugacy classes of  $S_n$  and partitions of  $n$ .

To find the size of the conjugacy classes of symmetric groups of  $n$  elements, we have the following theorem;

**Theorem 5** ((Loehr, 2011)). *For each  $\mu \in \text{Par}(n)$ , the number of permutations  $\pi \in S_n$  with  $\text{type}(\pi) = \mu$  is*

$$|C_\pi| = \frac{n!}{z_\mu}, \tag{10}$$

if  $\mu$  consists of  $m_1$  ones,  $m_2$  twos, etc. Then,  $z_\mu$  is defined as

$$z_\mu = \prod_{i=1}^n i^{m_i} \cdot m_i! \tag{11}$$

*Proof.* Fix  $\pi \in S_n$ , where  $\text{type}(\pi) = \lambda$ . One can count the number of conjugates of any group elements and subgroups as follows

$$[S_n : \text{Stab}(\pi)] = [S_n : C_{S_n}(\pi)] = \frac{|S_n|}{|C_{S_n}(\pi)|}$$

where  $\text{Stab}(\pi) := \{\rho \in S_n : \rho * \pi = \pi\}$  is the stabiliser of  $\pi$  in  $S_n$ . From the fact that  $|S_n| = n!$ , it is enough to show that  $|C_{S_n}(\pi)| = z_\lambda$ .

To understand the arguments it is better to look at a specific example;

► **Example 10.** Let  $\lambda = (3, 3, 2, 2, 2, 1, 1, 1, 1) \vdash 16$ , where we could then look at

$$\pi = (1, 2, 3)(4, 5, 6)(7, 8)(9, 10)(11, 12)(13)(14)(15)(16).$$

A permutation  $\rho \in S_n$  lies in  $C_{S_n}(\pi)$  if and only if  $\rho\pi\rho^{-1} = \pi$ . This again is true if and only if, after applying  $\rho$  to each symbol in the cycle decomposition above, produces another cycle decomposition of  $\pi$ .

We are therefore reduced to counting the number of ways we can write down a complete cycle decomposition of  $\pi$  such that longer cycles come before shorter cycles. Note that we can rearrange the order of all the cycles of a given length as much as we want, and we can also cyclically permute the entries in any given cycle of  $\pi$ .

We could, for example, permute the four 1-cycles in any  $4!$  ways. We could also replace  $(4, 5, 6)$  by one of its three cyclic shifts  $(4, 5, 6)$ ,  $(5, 6, 4)$  or  $(6, 4, 5)$ . For this particular  $\pi$ , the product rule gives us that we have  $(2!3^2)(3!2^3)(4!1^4) = z_\lambda$  different possible complete cycle decompositions. ◀

For the general case, the argument is pretty similar;

Looking at  $z_\lambda = \prod_{i=1}^n i^{m_i} \cdot m_i!$  we can see that the term  $m_i!$  accounts for permuting the  $m_i$  cycles of length  $i$ , while the term  $i^{m_i}$  accounts for all of the  $i$  possible cyclic shifts of each  $m_i$  cycles of length  $i$ . Multiplying these contributions we get  $|C_\pi| = z_\lambda$ , as desired. ◻

## 3.2 Representations of Symmetric Groups

As for general groups, we want to find a (matrix) representation for  $S_n$ . We can then use the same definitions as in Section 2.2.

► **Example 11.** Let us consider the defining representation of  $S_3$  with its character

$\chi^{\text{def}}$ . The elements of  $S_3$  have the following matrix representation

$$(1)(2)(3) \mapsto \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1,2)(3) \mapsto \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1,3)(2) \mapsto \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$(1)(2,3) \mapsto \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (1,2,3) \mapsto \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (1,3,2) \mapsto \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

The character values can then be computed by taking the traces of these matrices. Which means that the results are

$$\begin{aligned} \chi^{\text{def}}((1)(2)(3)) &= 3, & \chi^{\text{def}}((1,2)(3)) &= 1, & \chi^{\text{def}}((1,3)(2)) &= 1, \\ \chi^{\text{def}}((1)(2,3)) &= 1, & \chi^{\text{def}}((1,2,3)) &= 0, & \chi^{\text{def}}((1,3,2)) &= 0. \end{aligned}$$

From this its easy to see that if  $\pi \in \mathcal{S}_n$ , then

$$\begin{aligned} \chi^{\text{def}} &= \text{the number of ones in the diagonal of } X(\pi) \\ &= \text{the number of fixed points of } \pi. \end{aligned}$$

◀

For any  $S_n$  where  $n \geq 2$ , two simple representations are always given; the sign representation and the trivial representation.

**Definition 18.** The sign representation takes a permutation and represents this in a matrix with entries  $\pm 1$ , based on the sign of the permutation.

The sign of a permutation  $\pi \in S_n$  is  $+1$  if  $\pi$  is even, and  $-1$  if  $\pi$  his odd.

An even permutation can be composed of an even number of 2-cycles (also called transpositions), while an odd permutation can be obtained by an odd number of transpositions.

Let's consider the permutation  $\pi = (1234)(5)(67)(8)(9) = (1234)(67)$ . To compose this permutation, write each mapping into 2-cycles:  $\pi = (1234)(67) = (12)(23)(34)(41)(67)$ . For this permutation, there are 5 transpositions. Therefore, the permutation is odd.

With the sign representation and the trivial representation, it is convenient to know which partitions they are associated with.

**Proposition 3.** *The trivial representation corresponds to the partition  $(n)$ , while the sign representation corresponds to  $(1^n)$ .*

► **Example 12.** Let us again look at  $G = S_3$ , where the group is of order  $|G| = 3! = 6$ . Here we have the conjugacy classes

$$C_1 = \{\epsilon\}, \quad C_2 = \{(12), (13), (23)\}, \quad C_3 = \{(123), (132)\}.$$

We know that there are three irreducible representations of  $S_3$  because there has to be the same number of representations as there are conjugacy classes.

We have already looked at two different representations for this group; the trivial and sign representation. From these representations, we have the characters;

$$\begin{aligned} \chi_1 &= (1, 1, 1) && \text{trivial character} \\ \chi_2 &= (1, -1, 1) && \text{sign character} \end{aligned}$$

We can use the Corollary 2 to find the last character. The character of the regular representation is defined to be

$$\chi^{reg} = \sum_{i=1}^3 \chi_i(g_1)\chi_i = (6, 0, 0)$$

From this sum, we get the following equations:

$$\begin{aligned} 1 \cdot 1 + 1 \cdot 1 + \chi_3(1) \cdot \chi_3(1) &= 6 \\ 1 \cdot 1 + 1 \cdot (-1) + \chi_3(1) \cdot \chi_3(2) &= 0 \\ 1 \cdot 1 + 1 \cdot 1 + \chi_3(1) \cdot \chi_3(3) &= 0. \end{aligned}$$

Solving these, the last irreducible character is

$$\chi_3 = (2, 0, -1).$$

We can then fill in the known values in the character table

	$C_1$	$C_2$	$C_3$
$\chi_1$	1	1	1
$\chi_2$	1	-1	1
$\chi_3$	2	0	-1
#	1	3	2



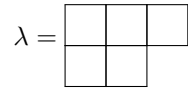
The character table of any symmetric group has entries that are all integers. If we were to look at any other groups, their character tables would have some entries which are not integers. For instance, the character table of abelian groups is determined by the root of unity (Tubbenhauer, 2022).

### 3.3 Diagrams

Partitions can be visualised in the following way:

**Definition 19.** Let  $\lambda = (\lambda_1, \dots, \lambda_m) \vdash n$ . The *Young diagram*, also referred to as the *Ferrers diagram*, of shape  $\lambda$  is an array of  $n$  boxes with  $m$  left-justified rows. In this array, the row  $i$  contains  $\lambda_i$  boxes, for  $1 \leq i \leq m$ . The boxes, often referred to as cells, in row  $i$  and column  $j$  have the coordinates  $(i, j)$ .

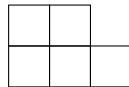
► **Example 13.** The partition corresponding to the cycle  $(1, 2, 4)(3, 5)$  is  $\lambda = (3, 2) \vdash 5$ . This partition has the Young diagram:



As we can see, there are  $n = 5$  cells in this diagram, which corresponds to the  $3 + 2 = 5$  cycle lengths we had in the cycle. ◀

In this master's thesis, the partition and the associated diagram will be denoted the same. This is to clarify which diagram belongs to which partition, and since diagrams are just a visual version of the partitions this notation does not have a great significance. The diagram visualises the length of each cycle of the permutation in its rows. From this, we also can note how many cycles there are of the same cycle type.

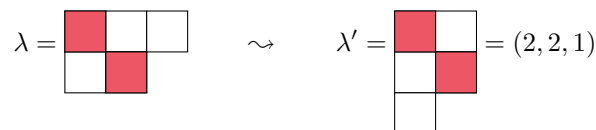
A Young diagram can also be said to be a diagram with the "English" notation, where we build east- and southward from the start position  $(0, 0)$ . If we were to use, for instance, the "French" notation we would draw the diagram as we were looking at the Cartesian coordinate system. That is to say, for  $\lambda = (3, 2)$  it would look like the following diagram:



Here we go east- and upward from the start position  $(0, 0)$ , and we have our largest row at the bottom as a "building block" for the rest of the diagram.

We will be using the English notation in this thesis, if not stated otherwise.

The *conjugate*  $\lambda'$  of  $\lambda$  is defined to be the sequence of column lengths of  $\lambda$ . If we visualise  $\lambda$  by its diagram, we can find its conjugate  $\lambda'$  by flipping the diagram of  $\lambda$  by its (principal) diagonal. The diagram of  $\lambda = (3, 2)$ , has the conjugate



The coloured boxes are those on the (principal) diagonal.

When looking at partitions, we would often like to know if they are equal to each other, or if one "dominates" the other.

**Definition 20.** Let  $\lambda, \mu \vdash n$ . We say that  $\lambda$  *dominates*  $\mu$ , denoted as  $\lambda \supseteq \mu$ , if

$$\sum_{i=1}^k \mu_i \leq \sum_{i=1}^k \lambda_i, \quad \forall k = 1, 2, \dots$$

If  $k > \ell(\lambda)$ , then we let  $\lambda_k$  to be zero, similarly for  $\mu_k$  if  $k > \ell(\mu)$ .

This dominance order is a partial order. For partitions of  $n$ , the sign partition  $(1^n) = (1, \dots, 1)$  is the smallest, while the trivial partition  $(n)$  is the largest amongst these partitions.

► **Example 14.** Looking at the two partitions

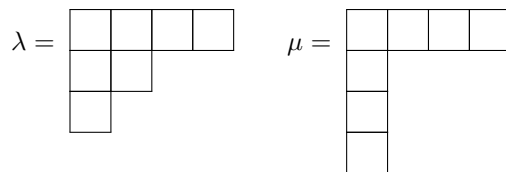
$$\begin{aligned} \lambda &= (4, 2, 1) \\ \mu &= (4, 1, 1, 1) \end{aligned} \quad \vdash 7$$

Using the same notations as in the definition 20, we look at the sums up to  $k = 4$ .

$$\begin{aligned} k = 1 & : \quad \mu_1 = 4 = \lambda_1 \\ k = 2 & : \quad \mu_1 + \mu_2 = 5 < \lambda_1 + \lambda_2 = 6 \\ k = 3 & : \quad \sum_{i=1}^3 \mu_i = 6 < \sum_{i=1}^3 \lambda_i = 7 \\ k = 4 & : \quad \sum_{i=1}^4 \mu_i = 7 = \sum_{i=1}^4 \lambda_i \end{aligned}$$

We can already see from  $k = 2$  that  $\lambda \supseteq \mu$ .

We could also look at the partitions visually;



Observe that  $\lambda$  and  $\mu$  have the same amount of cells in the first row, but  $\lambda$  has more cells in the second row than  $\mu$ . This results in that  $\lambda \supseteq \mu$ . ◀

We could also list all the partitions of an integer  $n$  in the dominance order. By doing this we can easier see how the partitions relate to each other.



► **Example 15.** Let's again look at partitions of  $n = 7$ . The complete dominance order would then be;

$$(7) \supseteq (6, 1) \supseteq (5, 2) \supseteq (5, 1^2) \supseteq (4, 3) \supseteq (4, 2, 1) \supseteq (4, 1^3) \supseteq (3^2, 1) \supseteq (3, 2^2) \\ \supseteq (3, 2, 1^2) \supseteq (3, 1^4) \supseteq (2^3, 1) \supseteq (2^2, 1^3) \supseteq (2, 1^5) \supseteq (1^7)$$

From this list of relations, we see that  $\lambda = (4, 2, 1)$  appears before  $\mu = (4, 1^3)$ . Therefore,  $\lambda$  dominates  $\mu$ . ◀

Instead of looking at a complete diagram, we can remove one diagram from another to create a "skew diagram".

**Definition 21.** Let  $\lambda \vdash n$  and  $\mu \vdash m$  be partitions, where  $\lambda \supseteq \mu$  and  $n \geq m$ . The corresponding *skew diagram*, also referred to as *skew shape*, is the set of boxes

$$\lambda \setminus \mu = \{c : c \in \lambda \cap c \notin \mu\}. \quad (12)$$

The skew diagram is just a normal diagram if  $\mu = \emptyset$ .

In this Master Thesis, we will denote the skew diagram as follows

$$\lambda \setminus \mu = (\mu_1 + (\lambda_1 - \mu_1), \mu_2 + (\lambda_2 - \mu_2), \dots, \mu_k + (\lambda_k - \mu_k),)$$

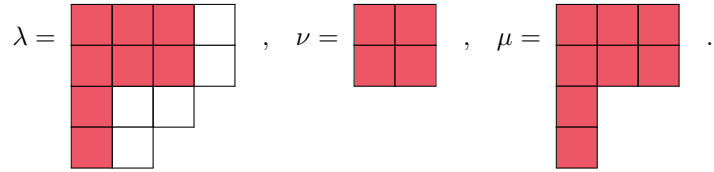
These diagrams do not look like the diagrams we have seen up until now. A skew diagram lacks a core, which the other diagrams we have seen before have. If  $\lambda = (4, 4, 3, 2)$ , and  $\mu = (3, 3, 1)$ . By removing  $\mu$  from  $\lambda$ , we get the following skew diagram;

$$\lambda \setminus \mu = \begin{array}{cccc} & & & \square \\ & & & \square \\ & & \square & \square \\ & \square & \square & \\ \square & \square & & \end{array} = (3 + 1, 3 + 1, 1 + 2, 2)$$

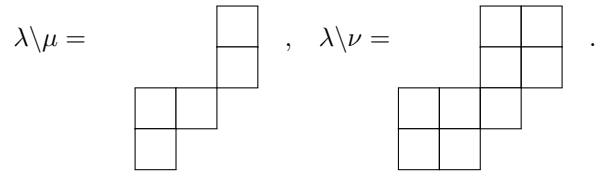
A skew diagram is said to be connected if all the boxes in the diagram are next to or below each other. This means that each box in the connected skew diagram touches another box by one of its sides.

► **Example 16.** Let  $\lambda = (4, 4, 3, 2)$ ,  $\mu = (3, 3, 1, 1)$  and  $\nu = (2, 2)$ . We can visualise them as their diagrams, where we colour in  $\nu$  and  $\mu$  inside  $\lambda$  to easier see what boxes we

are removing when looking at their skew diagrams;



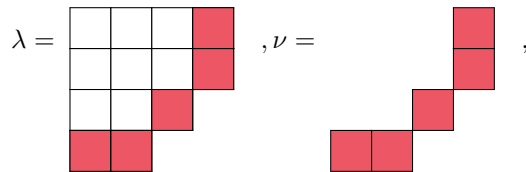
The skew diagrams are then



We see that the skew diagram  $\lambda \setminus \nu$  is connected, whereas the skew diagram  $\lambda \setminus \mu$  is not connected. ◀

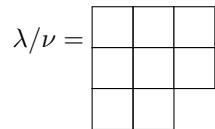
Instead of removing one partition from another partition, we can discard a skew diagram from a partition. This notation is very similar to the skew diagram, but now we write  $\lambda / \nu$  for removing a skew diagram  $\nu$  from  $\lambda$ .

Let  $\lambda = (4, 4, 3, 2)$ . We now want to remove the skew diagram  $\nu = (3 + 1, 3 + 1, 2 + 1, 2)$ . The diagrams then look as follows



where the coloured boxes are those that we want to remove.

The remaining diagram after removing  $\nu$  from  $\lambda$  would then be



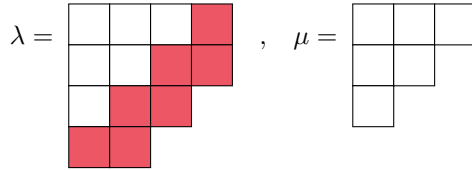
A special kind of skew diagram is the so-called "border strip". This skew diagram consists only of boxes at the border of a diagram.

**Definition 22** ((Chow and Paulhus, 2020)). A skew diagram  $\lambda \setminus \mu$  that is connected, where each row and column consists of a maximum of two boxes, is called a *border strip* of  $\lambda$ . If

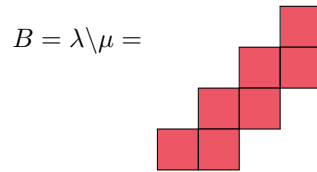
$B$  is a border strip, then its *height*  $\text{ht}(B)$  is the number of rows that  $B$  span, minus 1;

$$\text{ht}(B) = \#\text{rows of } B - 1. \tag{13}$$

► **Example 17.** Let's look at  $\lambda = (4, 4, 3, 2)$ . If we only want to keep the border of the diagram, where the border must be connected, we can remove the diagram  $\mu = (3, 2, 1)$ . To see it clearer, we colour in the boxes that we are interested in keeping;



The border strip  $B$  is then



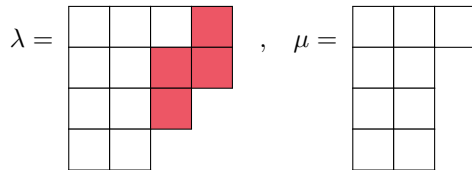
where the height is

$$\text{ht}(B) = 4 - 1 = 3.$$



Though they are called "border strips", these skew diagram does not be the *whole* border of the diagram. Let's look at another valid border strip in the following example;

► **Example 18.** Let  $\lambda = (4, 4, 3, 2)$ , but now we want to remove  $\mu = (3, 2, 2, 2)$ . The visualisation of the diagram  $\lambda$ , where we have coloured the border strip we want to keep, is then;



Such that the border strip  $B$  is



This border strip has the height

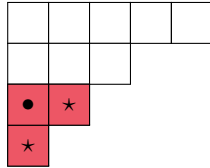
$$\text{ht}(B) = 3 - 1 = 2.$$



Another diagram, which is similar to skew diagrams, is *hooks*. These are diagrams with one arm and one leg, where only one row may consist of more than 1 cell, and only one column may have more than 1 cell. We can assign a corresponding hook for each cell in a diagram.

**Definition 23.** Let  $D$  be a diagram of shape  $\lambda$ . For each cell  $(i, j)$ , one can find the *hook* that corresponds to this cell. To do so we extend an arm to the rightmost cell in the same row, and a leg down to the lowest cell of the same column. The length of a hook in position  $(i, j)$ , also called the *hook length*, is the number of cells in the hook and denoted  $h_{i,j}$ .

If we look at  $\lambda = (5, 3, 2, 1)$ , the hook for the cell  $(3, 1) \in \lambda$  is then



where  $\bullet$  indicates the position  $(3, 1)$ , and  $\star$  is the extended arm and leg of the hook. This hook has the hook length  $h_{3,1} = 3$

The hook length of any cell in the diagram of shape  $\lambda$  can be found by using the formula (Nakayama, 1940)

$$h_{i,j} = \lambda_i + \lambda'_j - i - j + 1. \tag{14}$$

In this formula, we look at the  $i$ th part of  $\lambda$ , and the  $j$ th part of its conjugate  $\lambda'$ . From their sum, we subtract the row- and column-number, and add one (1).

In our  $\lambda$ , the hook length in the position  $(3, 1)$  is

$$h_{3,1} = \lambda_3 + \lambda'_1 - 3 - 1 + 1 = 2 + 4 - 3 - 1 + 1 = 3.$$

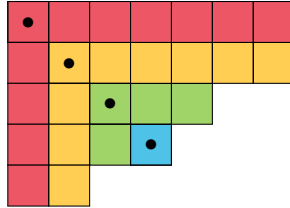
This particular formula can be helpful, especially when looking over large partitions.

The set of hooks in the principal diagonal is the *principal hook*, which is defined as (Chow and Paulhus, 2020)

$$H_i := \{(i, j) \in \lambda : j \geq i\} \cup \{(j, i) \in \lambda : j \geq i\}.$$

The hook lengths of the principal hooks are denoted as  $h_i$ , for the corresponding principal hook  $H_i$ .

► **Example 19.** Let  $\lambda = (7, 7, 5, 4, 2) \vdash 25$ . The principal hooks (visualised with different colours) are the following:



Here the principal hook lengths are

$$\begin{aligned} h_1 &= 11 && \text{(red principal hook)} \\ h_2 &= 9 && \text{(orange principal hook)} \\ h_3 &= 4 && \text{(yellow principal hook)} \\ h_4 &= 1 && \text{(green principal hook)} \end{aligned}$$



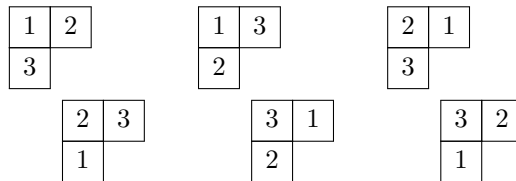
### 3.3.1 Tableaux

Diagrams are quite general, so let us specify them more. Instead of just looking at diagrams with empty boxes, each box can be assigned an entry.

**Definition 24.** A *Young tableau* is similar to a Young diagram, but in a Young tableau, every box is filled with exactly one number of  $[n]$ . There are  $n!$  different Young tableaux for each Young diagram.

This creates a bijective map between each box and elements in  $[n]$ .

► **Example 20.** The diagram corresponding to the partition  $\lambda = (2, 1) \vdash 3$  has  $3! = 6$  different tableaux. These tableaux are





To each cell, we can assign the *content*  $c = i - j$  of the position  $(i, j)$ . From this, the corresponding tableau is

0	1	2	3	4
-1	0	1		
-2				
-3				

Another way to give each cell an entry is to assign them to their associated hook length,  $h_{i,j}$  in position  $(i, j)$ . The corresponding tableau is, therefore,

8	6	4	2	1
5	2	1		
2				
1				



Every tableau can also be noted by its rows.

**Definition 27.** The *row word* of a tableau  $T$  is the permutation

$$\pi_T = R_l R_{l-1} \cdots R_1,$$

where  $R_1, \dots, R_l$  are the rows of  $T$

Let the tableau be given as

$$T = \begin{array}{|c|c|c|c|} \hline 1 & 3 & 4 & 8 \\ \hline 2 & 5 & 6 & \\ \hline 7 & & & \\ \hline \end{array},$$

then the row word for this tableau is

$$\pi_T = (7 \ 2 \ 5 \ 6 \ 1 \ 3 \ 4 \ 8).$$

When looking at tableaux, we want to know if they are equivalent to each other or not. One way of doing so is by looking at their row equivalence.

**Definition 28.** Two  $\lambda$ -tableaux  $T_1$  and  $T_2$  are said to be *row equivalent*,  $T_1 \sim T_2$  if the corresponding rows of the two tableaux contain the same elements. A *tabloid of shape*  $\lambda$ , or  $\lambda$ -tabloid, is defined to be a set

$$\{t\} = \{T_1 : T_1 \sim T\}$$

where  $T$  is of shape  $\lambda$ .

To visualise a tabloid, we think of a normal Young tableau but *deleting* the individual boxes in the rows. The tabloid for

$$t = \begin{array}{|c|c|c|} \hline 1 & 4 & 4 \\ \hline 4 & 3 & \\ \hline \end{array}$$

would then be

$$\{t\} = \left\{ \begin{array}{|c|c|c|} \hline 1 & 4 & 4 \\ \hline 3 & 4 & \\ \hline \end{array}, \begin{array}{|c|c|c|} \hline 4 & 1 & 4 \\ \hline 3 & 4 & \\ \hline \end{array}, \begin{array}{|c|c|c|} \hline 4 & 4 & 1 \\ \hline 3 & 4 & \\ \hline \end{array}, \right. \\ \left. \begin{array}{|c|c|c|} \hline 1 & 4 & 4 \\ \hline 4 & 3 & \\ \hline \end{array}, \begin{array}{|c|c|c|} \hline 4 & 1 & 4 \\ \hline 4 & 3 & \\ \hline \end{array}, \begin{array}{|c|c|c|} \hline 4 & 4 & 1 \\ \hline 4 & 3 & \\ \hline \end{array} \right\} \\ = \frac{1 \ 4 \ 4}{3 \ 4}.$$

If  $\lambda = (\lambda_1, \dots, \lambda_\ell) \vdash n$ , then the number of tableaux of shape  $\lambda$ , corresponding to any given equivalence class, would be

$$\lambda_1! \cdots \lambda_\ell! = \lambda!.$$

The number of  $\lambda$ -tabloids is then  $n!/\lambda!$ .

**Definition 29.** Suppose  $\lambda \vdash n$ . Let

$$M^\lambda = \mathbb{C} \{ \{t_1\}, \dots, \{t_k\} \},$$

where  $\{t_1\}, \dots, \{t_k\}$  is a complete list of  $\lambda$ -tabloids. Then  $M^\lambda$  is called the *permutation module* corresponding to  $\lambda$ .

Since we are considering only row equivalence classes, the rows of  $M^\lambda$  can be listed in any order and produce an isomorphic module. Then,  $M^\alpha$  is defined for any composition (ordered partition)  $\alpha$ .

► **Example 22.** Let  $\lambda = (n)$ . Then

$$M^{(n)} = \mathbb{C} \left\{ \frac{1 \ 2 \ \dots \ n}{\phantom{1 \ 2 \ \dots \ n}} \right\}$$

with the trivial action. ◀

► **Example 23.** If  $\lambda = (1^n)$ , then each equivalence class  $\{t\}$  consists of a single tableau. This tableau can be identified with a permutation in one-line notation. Since the action of  $S_n$  is preserved,

$$M^{(1^n)} \cong \mathbb{C}S_n$$



and the regular representation emerges. ◀

► **Example 24.** Consider  $\lambda = (n - 1, 1)$ , where each  $\lambda$ -tabloid is determined by the element in its second row. This gives the module isomorphism

$$M^{(n-1,1)} \cong \mathbb{C} \{ \mathbf{1}, \mathbf{2}, \dots, \mathbf{n} \},$$

which is the defining representation. ◀

Now, looking back at border strips. A new type of tableaux can be defined from these border strips.

**Definition 30** ((Chow and Paulhus, 2020)). Let  $\lambda$  be a partition of  $n$  and  $\alpha$  a composition of  $n$ . Then a *border strip tableau* of *shape*  $\lambda$  and *content*  $\alpha$ , denoted as  $BST(\lambda, \alpha)$ , is a tiling of the diagram of  $\lambda$  with border strips, such that:

- The area of the  $i$ th border strip is  $\alpha_i$ , and
- Writing the number  $i$  in each box of the  $i$ th border strip, then the numbers weakly increase across every row and down every column. The border strip tableau is therefore a semi-standard tableau.

In this paper, we will refer to the border strip with the number  $i$  written in it as  $B_i$ , i.e., the border strip that only contains 1 is  $B_1$ , the border strip that contains 2 is  $B_2$ , and so on.

► **Example 25.** If we have  $\lambda = (5, 5, 4, 1)$  and  $\alpha = (4, 5, 2, 1, 3)$ , then the border strip tableau of shape  $\lambda$  and type  $\alpha$  could be;

$$BST(\lambda, \alpha) \ni \begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 2 & 2 & 2 \\ \hline 1 & 2 & 3 & 3 & 4 \\ \hline 1 & 5 & 5 & 5 & \\ \hline 1 & & & & \\ \hline \end{array}$$

There are in fact  $|BST(\lambda, \alpha)| = 7$  elements in the set, namely

1	2	2	2	2	1	2	2	2	4	1	1	2	2	2
1	2	3	3	4	1	2	3	5	5	1	2	2	3	3
1	5	5	5		1	2	3	5		1	5	5	5	
1					1					4				
1	1	2	2	2	1	1	1	2	2	1	1	1	1	3
1	2	2	5	5	1	2	2	2	4	2	2	2	2	3
1	3	3	5		3	5	5	5		2	5	5	5	
4					3					4				
1	1	1	1	4										
2	2	2	5	5										
2	3	3	5											
2														

The colours are used to easily visually distinguish between each border strip  $B_i$ .

We can see that the area of the border strips is the same in each  $BST(\lambda, \alpha)$ . The border strip  $B_i$  has always an area of  $\alpha_i$  in the  $BST(\lambda, \alpha)$ , as was defined.

The height of each border strip varies, depending on how it is placed in the tableau; so  $ht(B_i^{(a)})$  in  $T_{(a)} \in BST(\lambda, \alpha)$  might not be the same as  $ht(B_i^{(b)})$  in  $T_{(b)} \in BST(\lambda, \alpha)$  for  $T_{(a)} \neq T_{(b)}$ . This, however, is not always the case.

There are only 7 elements in  $BST(\lambda, \alpha)$  from the condition that the numbers have to be weakly increasing across every row and down every column. ◀

Note that it is still a valid border strip tableau, even after removing the border strip  $B_i$  which corresponds to the largest number  $i$ .

### 3.4 Specht Modules

From defining standard young tableaux, they can be used to make another type of  $G$ -modules. When looking at groups, we are usually concentrating on their elements. Instead, let's fixate on the corresponding standard tableaux.

Before we define the module, we need to look at its basis.

**Definition 31.** Let  $\lambda \vdash n$  and  $t$  be a  $\lambda$ -tableau. Set

$$\kappa_t := \sum_{\sigma \in C_t} \text{sign}(\sigma)\sigma.$$

Then the *polytabloid* associated with  $t$  is

$$e_t := \kappa \{t\}.$$

From these elements, we get the following theorem for a new type of  $G$ -module.

**Theorem 6** ((Chow and Paulhus, 2020)). *Given any  $\lambda \vdash n$ , one can construct an irreducible representation corresponding to  $\lambda$ , namely the Specht module denoted  $S^\lambda$ . The Specht modules are pairwise non-isomorphic, where every irreducible representation of  $S_n$  is isomorphic to some  $S^\lambda$ . This  $S^\lambda$  is spanned by the polytabloids  $e_t$ , where  $t$  is a tableau of shape  $\lambda$ .*

For  $\lambda \vdash n$ , the corresponding Specht module forms a complete list of all the irreducible  $S_n$  modules over the complex field.

**Corollary 3.** The permutation modules,  $M^\mu$ , can be decomposed as

$$M^\mu = \bigoplus_{\lambda \supseteq \mu} m_{\lambda\mu} S^\lambda$$

with the diagonal multiplicity  $m_{\mu\mu} = 1$ .

Specht modules are spanned by the polytabloid  $e_t$ , thus having the following theorem.

**Theorem 7.** *The set*

$$\{e_t : t \in SYT(\lambda)\}$$

*is a basis of Specht modules corresponding to the same partition  $\lambda$ . This means that the set spans  $S^\lambda$ .*

Let's find the dimension of  $S^\lambda$ . All the basis of the Specht module is given, so the dimension depends only on how many elements there are in this set of basis.

**Theorem 8** ((Tubbenhauer, 2022)). *The dimension of the irreducible representations of  $S_n$  over  $\mathbb{C}$  are given by the number of standard Young tableaux;*

$$\dim S^\lambda = f^\lambda, \tag{15}$$

*where  $f^\lambda$  is the number of standard  $\lambda$ -tableaux.*

*Proof.* To prove this, we only need to look at the Specht modules. From Corollary 3, the Specht modules give all the irreducible representations of  $S_n$ . Theorem 7 states that  $S^\lambda = \text{Span}(\{e^\lambda : t \in SYT(\lambda)\})$ . This means that the dimension of Specht modules of shape  $\lambda$  is only depending on the number of standard tableaux of shape  $\lambda$ , where the number of  $\lambda$ -tableaux is defined as  $f^\lambda$ .  $\square$

To find the number of standard young tableau we could use the hook formula.

**Theorem 9** (Hook Formula (Griffiths and Lord, 2011)). *Let  $\lambda \vdash n$ , then the number of standard  $\lambda$ -tableaux are given by*

$$f^\lambda = \frac{n!}{\prod_{(i,j) \in \lambda} h_{i,j}}. \quad (16)$$

This formula is very simple to use because we only need to draw the diagram associated with  $\lambda$  to find  $f^\lambda$ .

From the Kostka numbers  $K_{\lambda\mu} = |SSYT_\lambda(\mu)|$ , the theorem for Specht modules is as follows

**Theorem 10** (Young's Rule). *The multiplicity of the Specht module  $S^\lambda$  in  $M^\mu$  is equal to the number of semistandard tableaux of shape  $\lambda$  and content  $\mu$ . The permutation module can then be decomposed into*

$$M^\mu \cong \bigoplus_{\lambda} K_{\lambda\mu} S^\lambda.$$

We can also restrict this direct sum to  $\lambda \supseteq \mu$ , such that

$$M^\mu \cong \bigoplus_{\lambda \supseteq \mu} K_{\lambda\mu} S^\lambda.$$

► **Example 26.** Let  $\mu = (3, 2, 1)$ . Then the possible  $\lambda \supseteq \mu$  are;

$$\begin{aligned} \lambda^1 = (3, 2, 1) &= \begin{array}{ccc} \bullet & \bullet & \bullet \\ \bullet & \bullet & \\ \bullet & & \end{array}, & \lambda^2 = (3, 3) &= \begin{array}{ccc} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & & \end{array}, \\ \lambda^3 = (4, 1, 1) &= \begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & & & \\ \bullet & & & \end{array}, & \lambda^4 = (4, 2) &= \begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & & \\ \bullet & & & \end{array}, \\ \lambda^5 = (5, 1) &= \begin{array}{ccccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & & & & \end{array}, & \lambda^6 = (6) &= \begin{array}{cccccc} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & & & & & \end{array}. \end{aligned}$$

where we get the associated  $\lambda$ -tableaux of content  $\mu$ :

$$T : \left\{ \begin{array}{l} \left( \begin{array}{ccc} 1 & 1 & 1 \\ 2 & 2 & \\ 3 & & \end{array} , \begin{array}{ccc} 1 & 1 & 1 \\ 2 & 2 & 3 \\ & & \end{array} , \begin{array}{cccc} 1 & 1 & 1 & 2 \\ 2 & & & \\ 3 & & & \end{array} , \right. \\ \left. \begin{array}{ccc} 1 & 1 & 1 & 2 \\ 2 & 3 & & \end{array} , \begin{array}{cccc} 1 & 1 & 1 & 3 \\ 2 & 2 & & \\ & & & \end{array} , \begin{array}{ccccc} 1 & 1 & 1 & 2 & 2 \\ 3 & & & & \end{array} , \right. \\ \left. \begin{array}{ccccc} 1 & 1 & 1 & 2 & 3 \\ 2 & & & & \end{array} , \begin{array}{ccccc} 1 & 1 & 1 & 2 & 2 & 3 \\ & & & & & \end{array} \right\}$$

From this set of tableaux, we get

$$M^{(3,2,1)} \cong S^{(3,2,1)} \oplus S^{(3,3)} \oplus S^{(4,1,1)} \oplus 2S^{(4,2)} \oplus 2S^{(5,1)} \oplus S^{(6)}.$$

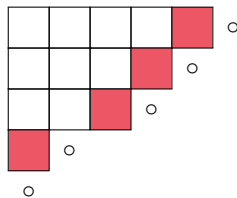


### 3.5 Branching Rule

What would happen if we would restrict or induce an irreducible representation  $S^\lambda$  of  $S_n$  to  $S_{n-1}$  or  $S_{n+1}$ , respectively? Intuitively, these two operations should correspond to either removing or adding a cell to the diagram corresponding with  $\lambda$ .

**Definition 32.** If  $\lambda$  is a diagram, then an *inner corner* of  $\lambda$  is a cell  $(i, j) \in \lambda$  whose removal leaves the Young diagram of a partition. Any partition obtain this way is denoted  $\lambda^-$ . While, an *outer corner* of  $\lambda$  is a cell  $(i, j) \notin \lambda$  whose addition creates the Young diagram of a partition. Every partition obtained like this is denoted  $\lambda^+$ .

The inner corners of  $\lambda$  are the cells at the end of a row and column of  $\lambda$ , whereas the outer corner is outside of the tableau. If  $\lambda = (5, 4, 3, 1)$ , then we can visualise these corners by colouring the inner corners and marking the outer corners with an open circle;





### 3.6 Symmetric Functions

Let us look at a infinite set of variables,  $\mathbf{x} = \{x_1, x_2, \dots\}$ , and consider the formal power series ring  $\mathbb{C}[[\mathbf{x}]]$ . The monomial  $x_{i_1}^{\lambda_1} x_{i_2}^{\lambda_2} \dots x_{i_\ell}^{\lambda_\ell}$  is said to have degree  $n$  if  $\sum_i \lambda_i = n$ . We also say that  $f(\mathbf{x}) \in \mathbb{C}[[\mathbf{x}]]$  is homogeneous of degree  $n$  if every monomial in  $f(\mathbf{x})$  has degree  $n$ .

Now for every  $n$ , there is a natural action of  $\pi \in S_n$  on the function  $f(\mathbf{x}) \in \mathbb{C}[[x]]$ ,

$$\pi f(x_1, x_2, \dots) = f(x_{\pi 1}, x_{\pi 2}, x_{\pi 3}, \dots),$$

where  $\pi i = i$  for  $i > n$ .

**Definition 33.** Let  $\lambda = (\lambda_1, \dots, \lambda_\ell) \vdash n$ . The corresponding *monomial symmetric function* is then

$$m_\lambda = m_\lambda(\mathbf{x}) = \sum x_{i_1}^{\lambda_1} \dots x_{i_\ell}^{\lambda_\ell} \quad (17)$$

where the sum is over all distinct monomials having the parts of  $\lambda$  as exponents.

From these functions, we can create a ring which has the monomial symmetric functions as a basis;

**Definition 34.** The ring of symmetric functions is

$$\Lambda = \Lambda(\mathbf{x}) = \mathbb{C}m_\lambda,$$

i.e., the vector space spanned by all the Monomial Symmetric Function,  $m_\lambda$ .

Since  $\Lambda$  is closed under multiplication, it is a ring. Though, there are certain elements of  $\mathbb{C}[[\mathbf{x}]]$  which are invariant under the  $\pi f(\mathbf{x})$  action, that is not in  $\Lambda$ , since it cannot be written as a finite linear combination of  $m_\lambda$ . The decomposition of the ring of symmetric functions is then

$$\Lambda = \bigoplus_{n \geq 0} \Lambda^n,$$

where  $\Lambda^n$  is the space that is spanned by all  $m_\lambda$  of degree  $n$ .

**Proposition 4.** *The space  $\Lambda^n$  has the basis*

$$\{m_\lambda : \lambda \vdash n\}$$

*and therefore the dimension is equal to the number of partitions of  $n$ ,  $p(n)$ .*

There are also other families of symmetric functions. Some of them are;

**Definition 35.**

- The *n*th Power Sum Symmetric Function:

$$p_n = m_{(n)} = \sum_{i \geq 1} x_i^n. \quad (18)$$

- The *n*th Elementary Symmetric Function:

$$e_n = m_{(1^n)} = \sum_{i_1 < \dots < i_n} x_{i_1} \cdots x_{i_n}. \quad (19)$$

- The *n*th Complete Homogeneous Symmetric Function:

$$h_n = \sum_{\lambda \vdash n} m_\lambda = \sum_{i_1 \leq \dots \leq i_n} x_{i_1} \cdots x_{i_n}. \quad (20)$$

As the name indicates, the *n*th power sum function  $p_n$  is a sum of all variables with power  $n$ . The elementary function  $e_n$  is the sum of all square-free monomials of degree  $n$ . It can therefore be considered a weight-generating function for partitions with  $n$  distinct parts. Similarly,  $h_n$  is the sum of all monomials of degree  $n$ , and it is the weight-generating function for all partitions with  $n$  parts.

We can also extend Definition 35 to  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_\ell)$ , by letting

$$f_\lambda = f_{\lambda_1} f_{\lambda_2} \cdots f_{\lambda_\ell}, \quad (21)$$

where the function  $f$  is one of the symmetric functions defined in Definition 35.

For example, if  $\lambda = (3, 2)$ , then the different symmetric function for  $\lambda = (2, 1)$  are

$$\begin{aligned} p_{(2,1)} &= p_2 p_1 = (x_1^2 + x_2^2 + \cdots)(x_1 + x_2 + \cdots) \\ e_{(2,1)} &= e_2 e_1 = (x_1 x_2 + x_2 x_3 + \cdots)(x_1 + x_2 + \cdots) \\ h_{(2,1)} &= h_2 h_1 = (x_1^2 + x_2^2 + \cdots + x_1 x_2 + x_2 x_3 + \cdots)(x_1 + x_2 + \cdots) \end{aligned}$$

### 3.6.1 Schur Function

One more important basis of  $\Lambda^n$  is the Schur function. These functions are intimately connected with irreducible representations of  $S_n$  and tableaux. The Schur functions are so adaptable that one can define them in many different ways.



Given any composition  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_\ell)$ , there exists a *monomial weight* in  $\mathbb{C}[[x]]$  that corresponds to this composition

$$\mathbf{x}^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_\ell^{\alpha_\ell}.$$

A generalised tableau  $T$ , of shape  $\lambda$  and content  $\alpha$ , also has a weight that is defined similarly to the monomial weight:

$$\mathbf{x}^T := \prod_{(i,j) \in \lambda} x_{T_{i,j}} = \mathbf{x}^\alpha$$

As an example, let us look at the following monomial weight.

► **Example 27.** Let us look at a tableau  $T$  of shape  $\lambda = (4, 2)$  and with the content  $\alpha = (3, 0, 0, 1, 2)$ . This tableau could for instance look like this

$$T = \begin{array}{|c|c|c|c|} \hline 1 & 5 & 1 & 4 \\ \hline 5 & 1 & & \\ \hline \end{array}.$$

Then, the monomial weight of this tableau is

$$\mathbf{x}^T = x_1^3 x_2^0 x_3^0 x_4^1 x_5^2 = x_1^3 x_4 x_5^2.$$



From the weight monomial, the Schur function can be defined. These kinds of functions are similar to the monomial weight, but instead are summing over the semi-symmetric tableaux of form  $\lambda$ .

**Definition 36.** Given a partition  $\lambda$ , the associated Schur function is

$$s_\lambda(\mathbf{x}) = \sum_{T \in SSYT_\lambda} \mathbf{x}^T = \sum_{T \in SSYT_\lambda} \prod_{(i,j) \in \lambda} x_{T_{i,j}}.$$

This function is symmetric.

The Schur function can also be defined to be a sum over compositions  $\mu$  of  $n$  with Kostka numbers:

$$s_\lambda = \sum_{\mu} K_{\lambda\mu} \mathbf{x}^\mu. \tag{22}$$

This is because the Kostka numbers  $K_{\lambda\mu}$  is the number of semi-standard tableaux of shape  $\lambda$  and content  $\mu$ , and  $\mathbf{x}^T = \mathbf{x}^\alpha$  for  $T$  of shape  $\lambda$  and type  $\alpha$ .

► **Example 28.** Let us look at the partition  $\lambda = (2, 1)$ . Some of the valid semi-standard tableaux of shape  $\lambda$  are then;

1	1	1	2	1	1	1	3	...
2		2		3		3		
1	2	1	3	1	2	1	4	...
3		2		4		2		

The Schur function for  $\lambda$  would be

$$\begin{aligned}
 s_{(2,1)} &= \sum_{T \in SSYT_{(2,1)}} x^T = \sum_{\alpha} K_{\lambda\alpha} \mathbf{x}^{\alpha} \\
 &= x_1^2 x_2 + x_1 x_2^2 + x_1^2 x_3 x_1 x_3^2 + \dots + 2x_1 x_2 x_3 + 2x_1 x_2 x_4 + \dots
 \end{aligned}$$



Note that the Schur function for sign representations  $(1^n)$  is

$$s_{(1^n)} = e_n(x), \tag{23}$$

while the Schur function for trivial representations  $(n)$  is

$$s_{(n)} = h_n(x). \tag{24}$$

One could also express the Schur function in terms of power sums;

$$s_{\lambda} = \frac{1}{n!} \sum_{\pi \in S_n} \chi^{\lambda}(\pi) p_{\pi} = \sum_{\mu} z_{\mu}^{-1} \chi_{\mu}^{\lambda} p_{\mu} \tag{25}$$

The last equation comes from the fact that  $\chi_{\mu}^{\lambda}$  is the value of the character  $\chi^{\lambda}$  on  $K_{\mu}$ , and  $z_{\mu} = |Z_g| = \frac{n!}{|K_{\mu}|}$ .

The Schur function can also be expressed associated with a skew partition. This function is a determinant, whose entries are the complete homogeneous symmetric function  $h_i$ , for  $0 \leq i \leq \ell(\lambda)$ . This expansion is called the *Jacobi-Trudi identity* (Stanley and Fomin, 1999, p. 342–344).

**Theorem 12** ((Stanley and Fomin, 1999)). *Let  $\lambda$  and  $\mu$  be two partitions with the same amount of parts,  $\ell(\lambda) = \ell(\mu)$ , where  $\mu \trianglelefteq \lambda$ . Then*

$$s_{\lambda/\mu} = \det \left( h_{\lambda_i - \mu_j - i + j} \right)_{i,j=1}^{\ell(\lambda)},$$

where  $h_0 = 1$  and  $h_m = 0$  for  $m < 0$ .

### 3.6.2 Characteristic Map

Let  $R^n = R(S_n)$  be the space that contains all class functions on  $S_n$ . We want to find a map that sends  $R^n$  to  $\Lambda^n$ .

The first thing to note is that  $\dim R^n = \dim \Lambda^n = p(n)$ , for  $p(n)$  the number of partitions of  $n$ . This means that these two spaces are isomorphic as vector spaces. We then have an inner product on  $R^n$ , where irreducible characters on  $S_n$  form an orthonormal basis (Theorem 2). From this, we can define an inner product on  $\Lambda^n$  by

$$\langle s_\lambda, s_\mu \rangle = \delta_{\lambda\mu}.$$

From the Equation (25), we define a map to preserve the inner product which was just defined.

**Definition 37.** The *characteristic map* is  $ch^n : R^n \mapsto \Lambda^n$ , defined by

$$ch^n(\chi) = \sum_{\mu \vdash n} z_\mu^{-1} \chi_\mu p_\mu,$$

where  $\chi_\mu$  is the character value of the character  $\chi$  on  $\mu$ .

If we apply  $ch^n$  to irreducible characters, then the character map is

$$ch^n(\chi^\lambda) = s_\lambda$$

from Equation (25). Since  $ch^n$  takes one orthonormal basis to another orthonormal basis, the following proposition comes as a consequence;

**Proposition 5.** *The map  $ch^n$  is an isometry between  $R^n$  and  $\Lambda^n$ .*

### 3.6.3 Littlewood-Richard Rule

The Littlewood-Richardson rule is a combinatorial formula of the coefficients when multiplying two Schur functions  $s_\mu s_\nu$ , expanded in the terms of Schur basis.

From Theorem 10,

$$M^\mu \cong \bigoplus_{\lambda} K_{\lambda\mu} S^\lambda. \tag{26}$$

Recall that the Kostka number,  $K_{\lambda\mu}$ , is the number of semistandard tableaux of shape  $\lambda$  and content  $\mu$ . This formula can be looked at from two other perspectives, in terms of characters and symmetric functions.

If  $\mu \vdash n$ , then  $M^\mu$  is a module for the induced character  $1_{S_\mu} \uparrow^{S_n}$ . This can be rewritten, for  $\mu = (\mu_1, \dots, \mu_m)$ , to be

$$1_{S_\mu} = 1_{S_{\mu_1}} \otimes \cdots \otimes 1_{S_{\mu_m}}.$$

The Equation (26) can also be rewritten as

$$1_{S_{\mu_1}} \cdots 1_{S_{\mu_m}} = \sum_{\lambda} K_{\lambda\mu} \chi^\lambda. \quad (27)$$

To apply Theorem 10 for symmetric functions, we apply the characteristic map to the above equation:

$$s_{(\mu_1)} \cdots s_{(\mu_m)} = \sum_{\lambda} K_{\lambda\mu} s^\lambda. \quad (28)$$

Such that, if  $\mu = (3, 2)$ , then the permutation module can be decomposed as

$$M^{(3,2)} = S^{(3,2)} + S^{(4,1)} + S^{(5)}$$

The relevant tableaux are then

$$T : \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 2 & 2 & \\ \hline \end{array}, \quad \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 2 \\ \hline 2 & & & \\ \hline \end{array}, \quad \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 2 & 2 \\ \hline & & & & \\ \hline \end{array}.$$

The permutation module can then be expressed as

$$1_{S_3} \cdot 1_{S_2} = \chi^{(3,1)} + \chi^{(4,1)} + \chi^{(5)},$$

or

$$s_3 s_2 = s_{(3,1)} + s_{(4,1)} + s_{(5)}.$$

We want to find out what would happen when looking at the multiplication between two Schur functions of arbitrary partitions, or generally the expansion

$$s_\lambda s_\mu = \sum_{\nu} c_{\lambda\mu}^{\nu} s_\nu. \quad (29)$$

Equivalently, we want the multiplicities of the irreducibles in

$$\chi^\lambda \cdot \chi^\mu = \sum_{\nu} c_{\lambda\mu}^{\nu} \chi^\nu$$

or

$$(S^\lambda \otimes S^\mu) \uparrow^{S_n} = \bigoplus_{\nu} c'_{\lambda\mu} S^\nu$$

for  $|\lambda| + |\mu| = n$ . The  $c'_{\lambda\mu}$  is called the *Littlewood-Richardson coefficients*. The Littlewood-Richardson rule gives a way to interpret these coefficients combinatorially.

Before defining the Littlewood-Richardson rule, we need to see how these coefficients act in the expansion of skew Schur functions. The definitions in Section 3.6.1 still apply even if  $\lambda$  is replaced with a skew diagram, where the resulting function  $s_{\lambda/\mu}(\mathbf{x})$  is still a symmetric formula.

**Proposition 6.** *Define  $s_\lambda(\mathbf{x}, \mathbf{y}) = s_\lambda(x_1, x_2, \dots, y_1, y_2, \dots)$ . Then*

$$s_\lambda(\mathbf{x}, \mathbf{y}) = \sum_{\mu \subseteq \lambda} s_\mu(\mathbf{x}) s_{\lambda/\mu}(\mathbf{y}). \quad (30)$$

Since  $s_{\lambda/\mu}(\mathbf{x})$  is symmetric, this can be expressed as a linear combination of ordinary Schur functions.

**Theorem 13.** *If the  $c_{\mu\nu}^\lambda$  are Littlewood-Richardson coefficients, and  $|\mu| + |\nu| = |\lambda|$ , then*

$$s_{\lambda/\mu} = \sum_{\nu} c_{\mu\nu}^\lambda s_\nu.$$

Now, we need to explain what the coefficient  $c_{\mu\nu}^\lambda$  counts.

**Definition 38.** A *lattice permutation* is a sequence of positive integers  $\pi = i_1 i_2 \dots i_n$  such that, for any prefix  $\pi_k = i_1 i_2 \dots i_k$  and any positive integer  $l$ , the number of  $l$ 's in the  $\pi_k$  is at least as large as the number of  $(l+1)$ 's in the same prefix. A *reverse lattice permutation* is a sequence  $\pi$  such that its reverse,  $\pi^r$ , is a lattice sequence.

Looking at the three permutations

$$\pi = (1 \ 2 \ 1 \ 1 \ 3 \ 2) \quad \sigma = (2 \ 1 \ 3 \ 2 \ 1 \ 1) \quad \rho = (1 \ 2 \ 2 \ 1 \ 1 \ 3)$$

The permutation  $\pi$  is a lattice permutation,  $\sigma$  is a reverse lattice permutation, while  $\rho$  is neither a lattice permutation nor a reverse lattice permutation. The reason for  $\rho$  not being a lattice permutation is that the prefix  $1 \ 2 \ 2$  has more twos than ones.

Lattice permutations are another way of encoding standard tableaux. Given a standard tableau  $P$  with  $n$  elements, then the sequence  $\pi = (i_1 i_2 \dots i_n)$  can be formed, where  $i_k = j$



## 4 Murnaghan–Nakayama rule

From the article by Murnaghan (1937), we find that the expansion of Schur functions in the power sum basis gives the irreducible characters of the symmetric group as coefficients. Nakayama (1940) gave later an "elegant" restatement of the same expansion. This expression being

$$s_\lambda(x) = \sum_{\alpha} \frac{\chi_{\alpha}^{\lambda}}{z_{\mu}} p_{\alpha}(x) = \sum_{\alpha} \frac{p_{\alpha}}{z_{\alpha}} \sum_{T \in \text{BST}(\lambda, \alpha)} (-1)^{\text{ht}(T)} \quad (31)$$

where  $\chi_{\mu}^{\lambda}$  are the irreducible characters of representations in  $S_n$  of shape  $\lambda$  and with content  $\alpha$ .

One can also phrase the Murnaghan–Nakayama rule as

$$p_r(x) s_{\lambda}(x) = \sum_{\mu} (-1)^{\text{ht}(\mu \setminus \lambda)} s_{\mu}(x), \quad (32)$$

where the sum goes over all partitions  $\mu$  such that  $\mu \setminus \lambda$  forms a border strip with  $r$  cells.

Lately, the Murnaghan–Nakayama rule is defined as follows

**Theorem 15** (Murnaghan–Nakayama rule). *Let  $\lambda \vdash n$ , and suppose that  $\alpha = (\alpha_1, \dots, \alpha_m)$  is a composition of  $n$ . Then*

$$\chi_{\alpha}^{\lambda} = \sum_{\nu} (-1)^{\text{ht}(\nu)} \chi_{\alpha \setminus \alpha}^{\lambda \setminus \nu} \quad (33)$$

where the sum is over all border strips  $\nu$  of size  $\alpha_1$ .

The branching rule, Theorem 11, is a special case of the Murnaghan–Nakayama rule. Take, for instance  $\alpha = (1, \alpha_2, \dots, \alpha_k)$ , and let  $\pi \in S_n$  have the content  $\alpha$ . Since  $\pi$  has a fixed point, the character  $\chi_{\alpha}^{\lambda}$  is then

$$\chi_{\alpha}^{\lambda} = \chi^{\lambda}(\pi) = \chi^{\lambda} \downarrow_{S_{n-1}}(\pi),$$

which corresponds to the left-hand side of the first equation in Theorem 11. For the right hand side of the Murnaghan–Nakayama rule;  $|\nu| = 1$  forces  $\lambda \setminus \nu$  to be of the form  $\lambda^-$  with all signs  $(-1)^0 = +1$ .

*Proof.* Let  $m = \alpha_1$ , and let us consider  $\pi\sigma \in S_{n-m} \times S_m \subseteq S_n$ , where  $\pi$  has the content  $(\alpha_2, \dots, \alpha_k)$  and  $\sigma$  is an  $m$ -cycle. The characters  $\chi^{\mu} \otimes \chi^{\xi}$ , where  $\mu \vdash n-m$  and  $\xi \vdash m$ , form a basis of the class functions on  $S_{n-m} \otimes S_m$ . Then,

$$\chi_{\alpha}^{\lambda} = \chi^{\lambda}(\pi\sigma) = \chi^{\lambda} \downarrow_{S_{n-m} \times S_m}(\pi\sigma) = \sum_{\substack{\mu \vdash n-m \\ \xi \vdash m}} m_{\mu\xi}^{\lambda} \chi^{\mu}(\pi) \chi^{\xi}(\sigma). \quad (34)$$

To find the multiplicities  $m_{\mu\xi}^\lambda$ , we can use Frobenius reciprocity (Sagan, 2001, Theorem 1.12.6) and the characteristic map:

$$\begin{aligned}
m_{\mu\xi}^\lambda &= \langle \chi^\lambda \downarrow_{S_{n-m} \times S_m}, \chi^\mu \otimes \chi^\xi \rangle \\
&= \langle \chi^\lambda, (\chi^\mu \otimes \chi^\xi) \uparrow^{S_n} \rangle \\
&= \langle \chi^\lambda, \chi^\mu \cdot \chi^\xi \rangle \\
&= \langle s_\lambda, s_\mu s_\xi \rangle \\
&= c_{\mu\xi}^\lambda,
\end{aligned}$$

where  $c_{\mu\xi}^\lambda$  is the Littlewood-Richardson coefficient.

Therefore, we can write Equation (34) as

$$\chi^\lambda(\pi\sigma) = \sum_{\mu \vdash n-m} \chi^\mu(\pi) \sum_{\xi \vdash m} c_{\mu\xi}^\lambda \chi^\xi(\sigma). \quad (35)$$

We must now evaluate the character value  $\chi^\xi(\sigma)$ , where  $\sigma$  is an  $m$ -cycle.

**Lemma 1.** *If  $\xi \vdash m$ , then*

$$\chi_{(m)}^\xi = \begin{cases} (-1)^{m-r} & \text{if } \xi \text{ is a hook} \\ 0 & \text{otherwise} \end{cases}.$$

This is a special case of the theorem, and if  $\alpha = (m)$  then  $\chi_\alpha^\xi \neq 0$  if and only if we can remove all the cells of  $\xi$  in one single sweep. This happens when  $\xi$  is a hook diagram, and the Murnaghan-Nakayama sum has only one single term in this case.

*Proof.* By the extension of the Schur function, Equation (31),  $\chi_{(m)}^\xi$  is  $z_{(m)}$  times the coefficient of  $p_m$  in

$$s_\xi = \sum_{\mu} \frac{1}{z_\mu} \chi_\mu^\xi p_\mu.$$

By using the complete homogeneous Jacobi-Trudi determinant (Theorem 12), we get that

$$s_\xi = \det (h_{\xi_i - i + j})_{i,j=1}^l = \sum_{\gamma} \pm h_\gamma,$$

where the sum is over all compositions  $\gamma = (\gamma_1, \dots, \gamma_l)$  that occurs as a term in the determinant. Each  $h_{\gamma_i}$  in  $h_\gamma$  can also be written as a linear combination of power sums. Since the  $p$ 's are a multiplicative basis, the resulting linear combination for  $h_\gamma$  will not contain  $p_m$  unless  $\gamma$  contains exactly one nonzero part. This part must then be  $m$ . Thus,  $\chi_{(m)}^\xi \neq 0$  only when  $h_m$  appears in the preceding determinant.



The largest index to appear in the determinant must be at the end of the first row. The first part of  $\xi$  is  $\xi_1 - 1 + l = h_{(1,1)}$ , for  $h_{(1,1)}$  the hook length of the cell  $(1, 1)$ . Further, we always have that  $m = |\xi| \geq h_{(1,1)}$ . Therefore,  $\chi_{(m)}^\xi$  is nonzero only when  $h_{(1,1)} = m$ , i.e., when  $\xi$  is a hook  $(r, 1^{m-r})$ . We then have

$$s_\xi = \begin{vmatrix} h_r & \cdots & & & & h_m \\ h_0 & h_1 & \cdots & & & \\ 0 & h_0 & h_1 & \cdots & & \\ 0 & 0 & h_0 & h_1 & \cdots & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{vmatrix}$$

$$= (-1)^{m-r} h_m + \text{other terms that does not involve } p_m.$$

Since  $h_m = s_{(m)}$  corresponds to the trivial character, comparing coefficients of  $p_m/m$  in this last set of equalities yields  $\chi_{(m)}^\xi = (-1)^{m-r}$ , as desired.  $\square$

From this lemma and from Equation (35), we need to find  $c_{\mu\xi}^\lambda$  for when  $\xi$  is a hook.

**Lemma 2.** *Let  $\lambda \vdash n$ ,  $\mu \vdash n - m$ , and  $\xi = (r, 1^{m-r})$  be partitions. Then the Littlewood-Richardson coefficient  $c_{\mu\xi}^\lambda = 0$ , unless each edgewise-connected component of  $\lambda/\mu$  is a rim hook. In that case, if there are  $k$  component hooks spanning a total of  $c$  columns, then*

$$c_{\mu\xi}^\lambda = \binom{k-1}{c-r}.$$

*Proof.* By the Littlewood-Richardson rule (Theorem 14),  $c_{\mu\xi}^\lambda$  is the number of semi-standard tableaux  $T$  of shape  $\lambda/\mu$  containing  $r$  ones, and a single copy of each  $2, 3, \dots, m - r + 1$ , such that  $\pi_T$  is a reverse lattice permutation. Thus, the number greater than one in  $\pi_T^r$  must occur in increasing order. This condition, together with the semi-standardness, gives the following constraints on the tableaux  $T$ :

- T1. Any cell of  $T$  having a cell to its right must contain a one
- T2. Any cell of  $T$  having a cell above must contain an element bigger than one.

Now, if  $T$  contains a  $2 \times 2$  block, then there is no way to fill the lower left cell in a way that satisfies both conditions above. Therefore,  $c_{\mu\xi}^\lambda = 0$  if the components of the shape of  $T$  are not rim hooks.

Now, suppose  $\lambda/\mu = \bigsqcup_{i=1}^k \nu^{(i)}$ , where each  $\nu^{(i)}$  is a component skew hook. The above conditions, and the fact that 2 through  $m - r + 1$  increase in  $\pi_T^r$ , show that every rim hook

must be of the form

$$\nu^{(i)} = \begin{array}{cccc} & & 1 & 1 & 1 & b \\ & & d & & & \\ & 1 & 1 & d+1 & & \\ d+2 & & & & & \\ d+3 & & & & & \end{array}$$

where  $d > 1$  is the smallest number that has not yet appeared in  $\pi_T^r$ , and  $b$  is either 1 or  $d-1$ . Thus, all of the entries in  $\nu^{(i)}$  are determined once we choose the value of  $b$ . Whereas, in  $\nu^{(1)}$ , we must have that  $b = 1$ .

By the second condition, T2, there are  $c - (k - 1)$  ones fixed for  $T$ . Hence, there are  $r - c + k - 1$  ones left to be distributed among the  $k - 1$  cells marked with a  $b$ . The number of ways this can be done is

$$c_{\mu\xi}^\lambda = \binom{k-1}{r-c+k-1} = \binom{k-1}{c-r}.$$

□

By putting the values from the two lemmas into Equation (35) we get

$$\chi^\lambda(\pi\sigma) = \sum_{\mu} \chi^\mu(\pi) \sum_{r=1}^m \binom{k-1}{c-r} (-1)^{m-r}. \quad (36)$$

Now since the three quantities  $k \leq c \leq m$  represent the number of skew hooks  $\nu^{(i)}$ , the number of columns in  $\nu^{(i)}$ , and the number of cells in  $\nu^{(i)}$ , respectively. We then get

$$\begin{aligned} \sum_{r=1}^m \binom{k-1}{c-r} (-1)^{m-r} &= (-1)^{m-1} \binom{k-1}{c-1} + (-1)^{m-2} \binom{k-1}{c-2} + \dots \\ &+ (-1)^{m-c} \binom{k-1}{0} + (-1)^{m-c+1} \binom{k-1}{-1} + \dots \\ &+ (-1)^0 \binom{k-1}{c-m} \end{aligned} \quad (37)$$

A binomial coefficient  $\binom{a}{b}$  is rewritten to be

$$\binom{a}{b} = \begin{cases} (-1)^b \binom{-a+b-1}{b} & \text{if } b \geq 0; \\ (-1)^{a-b} \binom{-b-1}{a-b} & \text{if } a \geq b; \\ 0 & \text{otherwise.} \end{cases}$$

The terms in Equation (37) will be equal to 0 if  $r > c$  and if  $c - r > k - 1$ . Meaning that

$$\begin{aligned}
\sum_{r=1}^m \binom{k-1}{c-r} (-1)^{m-r} &= (-1)^{m-1} \binom{k-1}{c-1} + (-1)^{m-2} \binom{k-1}{c-2} + \dots \\
&\quad + (-1)^{m-c} \binom{k-1}{0} + (-1)^{m-c+1} \binom{k-1}{-1} + \dots \\
&\quad + (-1)^0 \binom{k-1}{c-m} \\
&= (-1)^{m-c} \binom{k-1}{c} + (-1)^{m-c-1} \binom{k-1}{1} + \dots \\
&\quad + (-1)^{m-c-k+1} \binom{k-1}{k-1} \\
&= (-1)^{m-c} \sum_{i=1}^{k-1} (-1)^i \binom{k-1}{i} \\
&= \begin{cases} (-1)^{m-c} & \text{if } k-1 = 0; \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

If  $k = 1$ , then  $\lambda/\mu$  is just a single skew hook  $\nu$  with  $m$  cells and  $c$  columns. Hence  $m - c = \text{ht}(\nu)$ , so Equation (36) becomes

$$\chi^\lambda(\pi\sigma) = \sum_{|\nu|=m} (-1)^{\text{ht}(\nu)} \chi^{\lambda \setminus \nu}(\pi),$$

which ends the proof of the Murnaghan-Nakayama rule.  $\square$

As we have already seen in the proof of the Murnaghan-Nakayama rule, there are some special cases that we can calculate for. We have already looked for  $\alpha = (n)$ , where

$$\chi_{(m)}^\lambda = \begin{cases} (-1)^{m-r} & \text{if } \lambda \text{ is a hook} \\ 0 & \text{otherwise} \end{cases}$$

Another special composition is for  $\alpha = (1^n)$ .

**Lemma 3.** *Let  $\lambda \vdash n$ . Then the character associated with  $\lambda$  at  $(1^n)$  is defined to be*

$$\chi_{(1^n)}^\lambda = f^\lambda,$$

where  $f^\lambda$  is the number of standard Young tableaux of shape  $\lambda$ .

Recall the hook formula (Theorem 9), such that the character value can be rewritten as

$$\chi_{(1^n)}^\lambda = \frac{n!}{\prod_{(i,j) \in \lambda} h_{i,j}} \quad (38)$$

This formula only depends on the hook lengths of the different boxes in the diagram associated with  $\lambda$ .

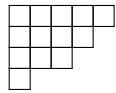
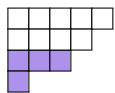
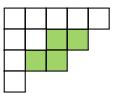
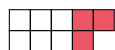

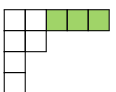





To calculate  $\chi_\alpha^\lambda$  the rule must be used iteratively. First, remove a border strip from  $\lambda$  with  $\alpha_1$  cells in all possible ways such that what is left is a Young diagram. Then strip away border strips with  $\alpha_2$  squares from the resulting diagrams, and so on. At some stage, it will either be impossible to remove a border strip of the right size or all cells will be deleted. If it is impossible to remove a border strip, then the corresponding character value is 0. Whereas, if all cells can be removed, then the character value is

$$\pm\chi_{(0)}^{(0)} = \pm 1. \tag{39}$$

► **Example 29.** Suppose we want to find the character value associated with  $\lambda = (5, 4, 3, 1)$  on  $\alpha = (4, 3, 3, 2, 1)$ . Let us display the Theorem 15 as a table over each step. At each step, we remove a skew hook of length equal to one of the parts of  $\alpha$ . More precisely; at the first step, we look at  $\alpha_1 = 4$ . Then we look at the next part,  $\alpha_2 = 3$ , and so on.

Now let us illustrate the Theorem 15. We colour in the corresponding skew hook  $\nu$  that is to be removed, to easier see the removal. We will also note the sign of the height of the

border strip which is removed under the corresponding border strip.

Step $i$	$\alpha$	$\lambda \setminus \nu$
0		
1	4	  (-)                      (-)
2	3	   (-)                      (+)                      (+)
3	3	   (N/A) (-)                      (+)                      (+)
4	2	(N/A)  (N/A) (+)
4	2	 (+)

The colours that are related are to illustrate which tableau they are related to. So the purple split into red and blue, and the red split into orange and yellow.

In step 3, when looking at the  $(2, 2, 1, 1)$ -diagram, there are no valid ways to remove  $\alpha_4 = 3$  boxes. We are then left with an invalid diagram (noted as N/A in the above array). The character value is then 0. In step 4, we can note that the same happens for the  $(2, 1)$ -diagrams (when we want to remove  $\alpha_4 = 2$  boxes). Meaning that we are only left with one diagram that can be *totally removed*, with character value 1.

We then get the following sequence;

$$\begin{aligned}
\chi_{(4,3,3,2,1)}^{(5,4,3,1)} &= \begin{array}{cccc} & -\chi_{(3,3,2,1)}^{(5,4)} & & -\chi_{(3,3,2,1)}^{(5,2,1,1)} \\ & -\left(-\chi_{(3,2,1)}^{(3,3)}\right) & -\chi_{(3,2,1)}^{(5,1)} & -\chi_{(3,2,1)}^{(2,2,1,1)} \\ & -\chi_{(2,1)}^{(2,1)} + \chi_{(2,1)}^{(3)} & -\chi_{(2,1)}^{(2,1)} & -0 \\ & -0 + \chi_{(1)}^{(1)} & -0 & +0 \\ & 0 + \chi_{()}^{(0)} & +0 & +0 \\ & 0 + 1 & +0 & +0 \\ & = 1 & & \end{array}
\end{aligned}$$

◀

In Theorem 15, we need to do the same algorithm many times before finding the value for the character of the representation. This might be quite time-consuming, especially when looking at bigger tableaux. Instead, we could define the Murnaghan-Nakayama rule to only look at border strips of tableaux of a given shape, and with a given content.

**Theorem 16** ((Chow and Paulhus, 2020)). *Let  $\lambda \vdash n$ , and  $\chi^\lambda$  be the irreducible character of  $S_n$  associated with  $\lambda$ . If  $\pi \in S_n$  and  $(\alpha_i)$  is the sequence of cycle lengths of  $\pi$ , then we get the following:*

$$\chi^\lambda(\pi) = \sum_{T \in \text{BST}(\lambda, \alpha)} \prod_{B \in T} (-1)^{\text{ht}(B)}, \quad (40)$$

where the sum goes over all the BSTs  $T$ , of shape  $\lambda$  and type  $\alpha$ . The product is over all the border strips  $B$  that tile  $T$ .

With this version of the Murnaghan-Nakayama rule, we do not need to use the same rule iteratively. We only need to find all the border strip tableaux, and from there find the character value.

Let's use the rule, as defined in Theorem 16, in some examples.

► **Example 30.** Let us go back the example with  $\lambda = (5, 4, 3, 1)$  and  $\alpha = (4, 3, 3, 2, 1)$ .

We here have the following border strip tableaux;

1	2	3	4	5	1	2	3	4	4	1	2	3	3	3	1	2	3	3	3
1	2	3	4		1	2	3	5		1	2	4	4		1	2	4	5	
1	2	3			1	2	3			1	2	5			1	2	4		
1					1					1					1				
1	2	2	4	4	1	2	2	4	5	1	2	2	3	5	1	2	2	2	5
1	2	3	5		1	2	3	4		1	2	3	3		1	3	3	3	
1	3	3			1	3	3			1	4	4			1	4	4		
1					1					1					1				
1	1	2	2	2	1	1	2	3	3	1	1	1	1	5					
1	3	3	3		1	2	2	3		2	3	3	3						
1	4	4			1	4	4			2	4	4							
5					5					2	4	4							
										2									

Calculating the formula from Theorem 16 we get;

$$\begin{aligned}
 \chi_\alpha^\lambda &= \chi^\lambda(\pi) = \sum_T \prod_{B \in T} (-1)^{ht(B)} \\
 &= \sum_{T \in BST(\lambda, \alpha)} \left( (-1)^{ht(B_1)} + (-1)^{ht(B_2)} + (-1)^{ht(B_3)} + (-1)^{ht(B_4)} + (-1)^{ht(B_5)} \right) \\
 &= \left( (-1)^3 (-1)^2 (-1)^2 (-1)^1 (-1)^0 \right) + \left( (-1)^3 (-1)^2 (-1)^2 (-1)^0 (-1)^0 \right) + \dots \\
 &= 1 - 1 - 1 + 1 - 1 + 1 - 1 - 1 + 1 + 1 + 1 \\
 &= 1
 \end{aligned}$$



There might be many different border strip tableaux for a pair of one partition and one composition. Though we still come to the same answer for the character value. Thus one might argue that this version, though you might have to keep a sharp eye for possible border strip tableaux, is shorter than the other. This, of course, depends on the composition for how easy it is to find all the border strip tableaux.

Now on to a much easier example.

► **Example 31.** Let  $\lambda = (5, 4, 2)$  and  $\alpha = (6, 3, 2)$ . We then have the two  $BST$ s;

1	1	1	1	1
1	2	3	3	
2	2			

1	1	1	1	1
1	2	2	2	
3	3			

So, by using the rule, we find that the character of  $S_n$  indexed with  $\lambda$  is

$$\begin{aligned} \chi_\alpha^\lambda &= \chi^\lambda(\pi) = \sum_{T \in BST(\lambda, \alpha)} \prod_{B \in T} (-1)^{ht(B)} \\ &= \sum_{T \in BST(\lambda, \alpha)} \left( (-1)^{ht(B_1)} + (-1)^{ht(B_2)} + (-1)^{ht(B_3)} \right) \\ &= 1 + (-1) = 0 \end{aligned}$$



From Theorem 16, since we are looking at a composition  $\alpha$ , we might wonder if the ordering is important or not. However, compositions have no criteria for ordering their parts. To see if the rule is affected by the ordering of these compositions, let's look at the following example.

► **Example 32.** Let  $\lambda = (5, 4, 2)$ , and  $\beta = (6, 2, 3)$ . With this composition, we cannot compute any valid  $BST(\lambda, \beta)$ . So with the Murnaghan-Nakayama rule, we have an empty sum. The character value is therefore

$$\chi_\beta^\lambda = 0$$

If we were to rather look at  $\gamma = (2, 3, 6)$ , we would have to conclude with the same result; there exists no  $BST(\lambda, \gamma)$ . Therefore,

$$\chi_\gamma^\lambda = 0$$

If  $\theta = (3, 2, 6)$ , then there exists  $BST(\lambda, \theta)$ . The border strip tableaux are

1	1	1	3	3
3	3	3	3	
2	3			

1	1	2	3	3
1	3	3	3	
1	3			

The corresponding character value is

$$\chi_\theta^\lambda = (-1) + 1 = 0$$





From the example, the character  $\chi^\lambda(\pi)$  is the same regardless of the order of the composition  $\alpha$ . Therefore, we can look at compositions  $\alpha$  that act like a partition, that is to say, it is weakly decreasing.

## 4.1 Alternative Versions

The Murnaghan-Nakayama rule can be found in different function families (Alexandersson, 2022). We already know that the Murnaghan-Nakayama rule is defined by the expansion of the Schur functions. From this expansion, several mathematicians have formulated the Murnaghan-Nakayama for other variants of the Schur functions. These are the  $k$ -Schur functions, the  $K$ - $k$ -Schur functions, and the Cyclic Schur functions.

The Murnaghan-Nakayama rule is also defined in other families. We will look at three families in the remainder of Section 4.1

### 4.1.1 Quantum Murnaghan-Nakayama rule

In the article "Skew Quantum Murnaghan-Nakayama Rule" (2011), Konvalinka gives an expansion of the product of a skew Schur function with a quantum power sum function in terms of skew Schur functions. Here, the author makes several speculations of generalisations of the Murnaghan-Nakayama rules for Hall-Littlewood  $P$ -functions.

The motivation for this article came from an open problem posed by Assaf and McNamara in their talk "A Pieri Rule for Skew Shapes" at FPSAC in 2010. The problem was to find combinatorial proof of the skew Murnaghan-Nakayama rule. Though "Skew Quantum Murnaghan-Nakayama Rule" provides a bijective proof of the skew *quantum* Murnaghan-Nakayama rule, the particular problem remains open.

### 4.1.2 Plethystic Murnaghan-Nakayama rule

The article "A Combinatorial Proof of a Plethystic Murnaghan-Nakayama Rule" by Wildon (2015), gives a combinatorial proof of a plethystic Murnaghan-Nakayama rule. The plethystic Murnaghan-Nakayama rule is a formula for computing the plethysm of a Schur function with a power sum symmetric function. The plethysm is an operation in algebraic combinatorics that describes the composition of symmetric functions, which was introduced by D.E.

Littlewood in 1950 (Wikipedia contributors, 2023b).

This version of the Murnaghan-Nakayama rule can be used to compute the coefficients in the expansion

$$s_\mu \cdot (p_r[h_m]) = s_\mu(p_r \circ h_m) = \sum_{\lambda \vdash rm + |\mu|} (-1)^{\text{ht}_r(\lambda/\mu)} s_\lambda.$$

This expansion was first stated in the article by Désarménien et al., where later Wildon gave a combinatorial proof of the plethystic generalisation of the Murnaghan-Nakayama rule. Wildon states in his that this extension already had been proved using the character theory of the symmetric group, but implies that the expansion implies a combinatorial formula for  $s_\mu(p_{r_1} \cdots p_{r_c} \circ h_m)$ , or more generally for  $s_\mu(p_{r_1} \cdots p_{r_c} \circ h_{m_1} \cdots h_{m_d})$ .

## 4.2 Usage of the Murnaghan-Nakayama rule

There are different ways one could use the Murnaghan-Nakayama rule, other than just to find the character value of a representation with respect to another representation.

The article written by Chow and Paulhus (2020) and the article written by Hamaker and Rhoades are two examples of how we can adjust the Murnaghan-Nakayama rule for different concerns.

### 4.2.1 Finding unknown partitions

One can use the Murnaghan-Nakayama rule explicitly to find an unknown partition. In the article written by Chow and Paulhus (2020), they use the Murnaghan-Nakayama rule to do exactly this. Here they want to answer the following question

Suppose that we are given two distinct irreducible characters  $\chi^\lambda$  and  $\chi^\mu$  of the symmetric group  $S_n$ . How hard is it to find a permutation  $\pi \in S_n$  such that  $\chi^\lambda(\pi) \neq \chi^\mu(\pi)$ ?

(Chow and Paulhus (2020))

To answer this question, they defined an algorithm which exploits the Murnaghan-Nakayama rule (Theorem 16) to find a composition  $\alpha$  that satisfies the in-equation.

The outline of the algorithm is given in two phases;

1. Forward Pass; The principal hook lengths are determined one at a time
2. Backward Pass; The principal hook shapes are determined one at a time in reverse order, starting with the innermost principal hook and working backwards. Here, at each step, we need to determine the amount of which the principal hook *overhangs* the arm and the leg of the principal hook just inside of it.

Starting with finding the hook lengths  $h_i$  in the forward pass, where we are looking at them in the order  $h_1, h_2, \dots$ . Next, we use the backward pass to find the shapes of the principal hooks, where we start at the last hook length and work backwards.

For the forward pass, there is an important observation.

**Lemma 4.** *A border strip tableau of shape  $\lambda$  and content  $\alpha$  cannot exist if  $\alpha_1 > h_1 = h_{(1,1) \in \lambda}$ .*

This is because the first border strip would then be too large to fit in the diagram, and therefore there cannot be any border strip tableau of content  $\alpha$ .

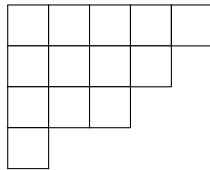
To determine the principal hook length  $h_1$  of  $\lambda$ , we can then consider the compositions

$$\alpha_j^{(i)} = \begin{cases} n - i, & \text{if } j = 1 \\ 1, & \text{if } 2 \leq j \leq i + 1. \end{cases} \quad (41)$$

for  $i \in \{0, 1, \dots, n - 1\}$ . We look at the queries  $\alpha^{(0)}, \alpha^{(1)}, \dots$  successively, and stop as soon as we encounter a nonzero value. We let  $\chi_{\alpha^{(i)}}^\lambda = 0$  if  $\alpha_1^{(i)} > h_1$ . Now, if  $\alpha_1^{(i)} = h_1$ , then there will be a border strip tableau whose first border strip is the first principal hook, while all other cells are covered by singletons.

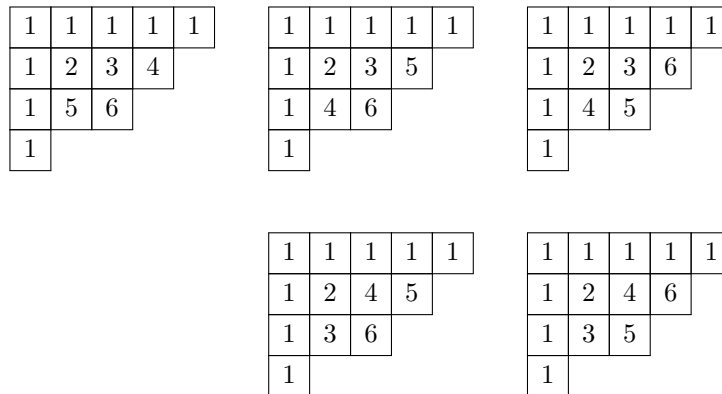
Let us look at this passage as an example;

► **Example 33.** Let us consider  $\lambda = (5, 4, 3, 1) \vdash 13$ .



We see that the principal hook length is  $h_1 = 8$ . Meaning that, if  $\alpha_1 > 8$  then  $\chi_\alpha^\lambda = 0$ . We, therefore, let  $\alpha_1 = 8$ .

We then get that the composition is  $\alpha = (8, 1, 1, 1, 1)$ . From this, we have the following border strip tableaux;



We can then use Theorem 16 to find that the character value corresponding to  $\lambda$  on  $\alpha$  is

$$\chi^{(5,4,3,1)}(8, 1, 1, 1, 1) = (-1)^3 \cdot 5 = -5$$

◀

To recover  $h_2$  we "freeze" the largest part of  $\alpha$  to be  $h_1$ , and then try decreasing values for the next largest part of  $\alpha$ .

Consider the new composition

$$\beta_j^{(i)} = \begin{cases} h_1, & \text{if } j = 1; \\ \min(h_1 - 2, n - h_1) - i, & \text{if } j = 2; \\ 1, & \text{if } 3 \leq j \leq n - \beta_1^{(i)} - \beta_2^{(i)} + 2. \end{cases} \quad (42)$$

Where we, again, look at the queries  $\beta^{(0)}, \beta^{(1)}, \dots$  successively, and stop as soon as we encounter a nonzero value. Since  $\beta_1^{(i)} = h_1$ , then we know that the first border strip of any border strip tableau of shape  $\lambda$  and content  $\beta^{(i)}$  must cover the entire first principal hook. If  $\beta_2^{(i)} > h_2$ , then there cannot be any border strip tableaux of shape  $\lambda$  and content  $\beta^{(i)}$ , because the second border strip is too large to fit inside the second principal hook.

► **Example 34.** Let's continue on Example 33, with  $\lambda = (5, 4, 3, 1) \vdash 13$ . Here  $\beta_1 = 8$ , and we now want to find  $\beta_2$ .

Using the formula from Equation (41), we find that

$$\begin{aligned} \beta_2^{(i)} &= \min(h_1 - 2, n - h_1) - i = \min(8 - 2, 13 - 8) - i = \min(6, 5) - i \\ \beta_2^{(i)} &= 5 - i. \end{aligned}$$

Looking at the diagram for  $\lambda$ . The smallest  $i$ , such that  $\beta_2^{(i)} \leq h_2$ , is  $i = 1$ . This gives us that  $\beta_2 = 4$ . ◀

Now for the backward pass; where we want to recover the shape of the principal hooks one at a time. We can reduce the case where we know the entire shape, except for the first principal hook.

Some useful lemmas that can be used in this passage;

**Lemma 5.** *A border strip of a border strip tableau cannot contain more than one box of the principal diagonal.*

**Lemma 6.** *The first  $k$  border strips in a border strip tableau must be entirely contained in the first  $k$  principal hooks.*

The proof of these can be found in the article.

Assuming that we know all  $k$  of the principal hook lengths  $h_i$ , we want to determine the arm and leg overhand of the first principal hook. To determine this we apply the following sequence of queries until we encounter a nonzero character value of  $\chi^\lambda(\gamma^{(i)})$ ;

$$\gamma_j^{(i)} = \begin{cases} h_1 - i & \text{if } j = 1 \\ h_2 + i & \text{if } j = 2 \\ h_j & \text{if } 3 \leq j \leq k \end{cases}$$

From Lemma 6, for the second border strip to contain more than  $h_2$  boxes, it must contain some of the boxes from the first principal hook. This cannot happen if the first border strip is too large, it would then restrict the second border strip from reaching the required boxes in the first principal hook.

The smallest  $i$  that allows border strips with parts  $\gamma_1^{(i)} = h_1 - i$  and  $\gamma_2^{(i)} = h_2 + i$ , gives us the desired information on the shortest overhand of the first principal hook. And we can therefore deduce the length of the longer overhang.

However, if we did know the overhang of a tableau we do not know where these overhands are supposed to go. We have two possible shapes, depending on which overhang is chosen to be the arm and which is selected to be the leg. The two shapes we are left with are then  $\lambda$  and  $\hat{\lambda}$ , where one denotes  $\hat{\lambda}$  to be the *dopplegänger* of  $\lambda$ . How to distinguish these dopplegänger is written more in-depth in the article.

### 4.2.2 Expansions of Irreducible Symmetric Group-characters

In the article by Hamaker and Rhoades (2022), they alter the Murnaghan-Nakayama rule to instead sum over the *monotonic tiling*  $\tau$  of  $\lambda$ -ribbons of size  $\mu$ . They give an explicit rule for calculating the expansion of a  $k$ -local function on the basis of irreducible  $S_n$  characters when the  $k$ -local function is a class function.

Hamaker and Rhoades (2022) look at power sums and *path power sums* to calculate the expansion using the classical Murnaghan-Nakayama rule (Theorem 15) and their altered Murnaghan-Nakayama rule, respectively. Their altered Murnaghan-Nakayama rule is called the *Path Murnaghan-Nakayama Rule*.

They take the base of the Murnaghan-Nakayama rule in the form of Equation (32). Hamaker and Rhoades start from the Murnaghan-Nakayama rule of power sums;

$$p_\nu = \sum_{\lambda \vdash n} \chi_\nu^\lambda s_\lambda,$$

for

$$\chi_\nu^\lambda = \prod_T (-1)^{\text{ht}(T)}.$$

The product multiplies over any  $\nu$ -ribbon tableaux  $T$  of shape  $\lambda$ .

To understand the *path power sums*, we first need to define Monotonic ribbon tilings.

**Definition 39.** In a *monotonic ribbon tiling* we have the following conditions;

- The tails lie in distinct columns;
- The tail depth decreases weakly from left to right;
- Each initial union of ribbons forms a partition.

Let us look at different tilings of the same diagram  $\lambda = (10, 9, 4, 2, 2)$ . In Figure 2, we draw the start of each tail as a white circle, which also visualises the depth of each tail. The end of the tails is drawn as black circles. We connect the tails with a line, to indicate which start position is related to which end position.

From the Definition 39 there is only one valid Monotonic ribbon tiling, Figure 2a. To better understand why, let us look at why the other two tilings are not valid monotonic ribbon tilings; First, Figure 2b has multiple tails that lie on the same columns, meaning that this tiling does not satisfy the first criteria of Definition 39. It is therefore not a monotonic ribbon tiling.

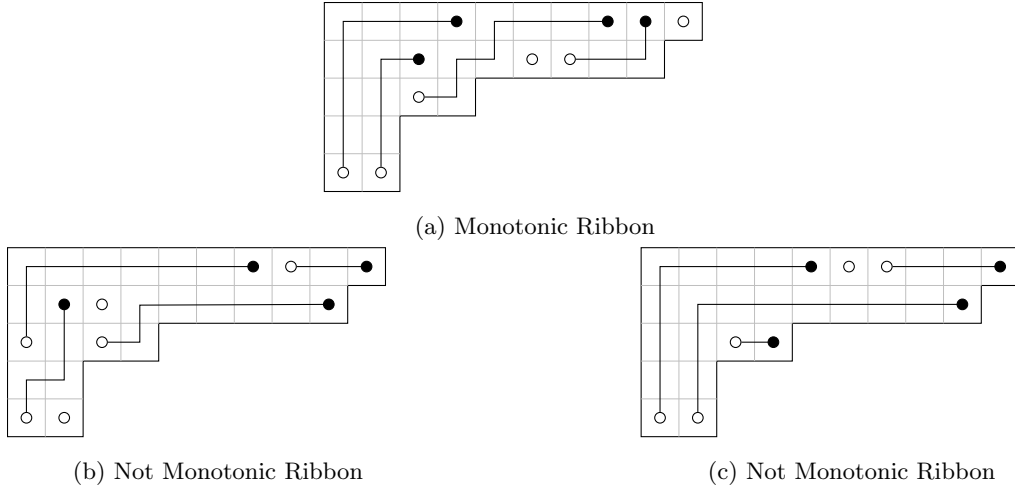


Figure 2: Tilings of Diagram  $(10, 9, 4, 2, 2)$

If we look at Figure 2c, we see that this tiling satisfies the first two criteria from Definition 39. Let us look at a union of two ribbons of this tiling, and see if it satisfies the last condition; If we only consider these two ribbons which start at the bottom of the diagram, we see that the union of these does not form a valid partition. This is because we get fewer cells in the first row than in the second row. Since we do not get a young diagram from the union, the last criterion of Definition 39 is not satisfied. This tiling is then not a valid Monotonic ribbon tiling.

Not that we have these monotonic ribbon tilings, we can define the Path Murnaghan-Nakayama rule;

**Theorem 17.** For  $\mu \vdash n$ , we have

$$\vec{p}_\mu = \sum_{\mu \vdash n} \vec{\chi}_\mu^\lambda s_\lambda.$$

We define  $\vec{\chi}_\mu^\lambda$  to be

$$\vec{\chi}_\mu^\lambda = m(\mu)! \cdot \prod_{\tau} (-1)^{ht(\tau)}$$

where we sum over all the monotonic tilings  $\tau$  of  $\lambda$  ribbons of size  $\lambda$ , and  $m_i(\mu)$  is the multiplicity of  $i \geq 1$  as a part of  $\mu$  and  $m(\mu)! := m_1(\mu)! \cdots m_n(\mu)!$ . The ribbons are to be added in all possible orders.

► **Example 35.** Let us look at the path power sum of  $\mu = (3, 2, 1)$ . Using Theorem 17



we have the formula

$$\vec{p}_\mu = \sum_{\mu \vdash n} \vec{\chi}_\mu^\lambda s_\lambda = \sum_{\mu \vdash n} \left( m(\mu)! \cdot \prod_{\tau} (-1)^{\text{ht}(\tau)} \right) s_\lambda$$

By looking over all possible partitions  $\lambda$  of  $n = 6$ , we try to find all the possible monotonic ribbon tilings.

By trial and failure, we find that there are monotonic ribbon tilings for the partitions  $(6)$ ,  $(5, 1)$ ,  $(4, 1, 1)$ ,  $(3, 3)$ , and  $(3, 2, 1)$ . The path power sum is then

$$\begin{aligned} \vec{p}_\mu &= 6 \cdot (1) s_{(6)} + 4 \cdot (-1) s_{(5,1)} + 2 \cdot (1) s_{(4,1,1)} + 2 \cdot (-1) s_{(3,3)} + 1 \cdot (-1) s_{(3,2,1)} \\ &= 6s_{(6)} - 4s_{(5,1)} + 2s_{(4,1,1)} - 2s_{(3,3)} - s_{(3,2,1)} \end{aligned}$$

◀

### 4.3 Results from Pak and Panova

In the article written by Pak and Panova (2015), they found that it is NP-hard to figure out if a character  $\chi^\lambda(\mu)$  is zero. They called the problem "Is the character  $\chi^\lambda(\mu) = 0$ " for CHARP.

Even when looking at characters of partitions of only two parts on another partition with only even parts, the problem CHARP is still NP-hard. With this observation, it implies that the problem CHARP is at least as hard as the *Knapsack problem*, which is defined to be:

KNAPSACK: Given the input  $(k, a_1, \dots, a_l)$ , determine whether there are  $\epsilon_i \in \{0, 1\}$  for  $i = 1, \dots, l$ , such that

$$k = \sum_{i=1}^l \epsilon_i a_i.$$

For any multiset  $R = \{r_1, \dots, r_q\}$  and integer  $s$ , we denote the number of ways to write  $s$  as a sum of entries from  $R$  as  $P_R$ . Such that

$$P_R(s) = \# \{(i_1, i_2, \dots) : 1 \leq i_1 < i_2 < \dots \leq q, r_{i_1} + r_{i_2} + \dots = s\}.$$

Let  $M$  be the set of all parts of  $\mu$ ,  $M = \{\mu_1, \dots, \mu_l\}$ . Then, to reduce the CHARP problem to KNAPSACK, by first using Jacobi-Trudi identity (Theorem 12), and later the Murnaghan-Nakayama rule to rewrite the character  $\chi^\lambda(\mu)$ , for  $\lambda = (n - 2k, 2k)$  and  $\mu = (2a_1, 2a_2, \dots, 2a_l)$ , to be

$$\chi^\lambda(\mu) = P_M(2k) - P_M(2k - 1).$$

Due to all the elements in  $M$  being even, we have that  $P_M(2k - 1) = 0$ , so we know that the character  $\chi^\lambda(\mu) = 0$  if and only if  $P_M(2k) = 0$ . To determine if  $P_M(2k) = 0$ , is the same as the KNAPSACK problem, since we are looking at a set of  $\epsilon_i$  multiplied with some elements in  $M$ . Because of this, we know that we have a problem that is at least as hard as a NP-complete problem, making it a NP-hard problem. Note that this also implies that to compute characters is itself #P-hard.

## 5 Computation

We can now compute the Murnaghan-Nakayama rule as a function in SageMath. The functions that are defined in this Master Thesis are inspired by predefined functions by The Sage Developers (2022).

### 5.1 Predefined Functions

Coding in a mathematics software system like SageMath comes with a lot of favours. One main factor is that there are already defined functions we can use to find the character table of any group. We can also specific characters of representations from predefined functions in this system.

#### 5.1.1 Characters of Representations

There are different ways of finding the characters of a representation in SageMath. One way is to directly find the characters from the representations.

If we were to look at the partition  $\lambda = (4, 3, 2)$ , the code would then be;

```
1  lm = [4,3,2]
2
3  rho = SymmetricGroupRepresentation(lm)
4  chi = rho.to_character()
5
6  chi.values()
```

Let's start by defining the partition we are interested in, `lm`. After this, generate the representation that is associated with the partition `lm`. Then generate the list of character values of the representation `rho` by using the function `to_character()`. To get the list of character values of this partition we then ask for the `values()` of the characters, `chi`.

The output is then

```
1  [168, 14, 4, 2, 0, -15, -1, 1, -1, 0, 2, -3, -4, -2, 0, -1, 1, 0, 3, -1,
   -1, 0, 1, 2, 0, -1, 0, 0, 0, 0]
```

The function gives a list of all the character values that are associated with the specific partition `lm` with the representation `rho`.

The function `to_character()` is directly based on the definition of characters, Definition 10. The function takes the trace of the matrix representation of the group associated with each element in the group. The function then lists all the traces in a list and returns this.

SageMath also use functions from GAP (2021). The GAP (2021) system provides comprehensive information on group theory, combinatorics, etc.

To find the character value in GAP, we first find all the irreducible representations of a given group  $G$ .

```

1  lm = [4,3,2]
2  n = sum(lm)
3
4  G = SymmetricGroup(n);
5  gap.Irr(G)

```

When running this code, we would get a list of all the irreducible characters of the Symmetric group of  $n$  elements, where  $n = 9$  in this case. To get the list of character values associated with the partition `lm`, the following code is needed;

```

6  S = Partitions(n);
7  N = len(G)
8
9  for i in range(len(S)):
10     if S[-i] == lm:
11         m = N-i
12         Chi = Char[i]
13         Chi

```

This code goes through each partition of  $n$ , finding the corresponding entry in `gap.Irr(G)`. The ordering in `gap.Irr(G)` is the reverse of `Partitions(n)`, meaning that we need to look at the last partition in `Partitions(n)` when we want to find a certain partition in `gap.Irr(G)`.

The output would then be

```

1  Character( CharacterTable( SymmetricGroup( [ 1 .. 9 ] ) ),
2  [ 168, 14, 4, 2, 0, -15, -1, 1, -1, 0, 2, -3, -4, -2, 0, -1, 1, 0, 3, -1,
   -1, 0, 1, 2, 0, -1, 0, 0, 0, 0 ] )

```

This result is the same as the previous one, even when we had to run through more code to get there.

### 5.1.2 Character Tables

SageMath can also compute the character table of a group. Let us look at two ways to generate the character table.

The first method is by using functions which are already defined in SageMath. The code is then:

► **Example 36.**

The code to call for the character table for the symmetric group of  $n = 6$  elements is the following;

```
1 G = SymmetricGroup(6)
2 CT = G.character_table() ; CT
```

Giving the following output;

```
1 [ 1 -1 1 -1 1 -1 1 -1 1 1 -1]
2 [ 5 -3 1 1 2 0 -1 -1 -1 0 1]
3 [ 9 -3 1 -3 0 0 0 1 1 -1 0]
4 [ 5 -1 1 3 -1 -1 2 1 -1 0 0]
5 [10 -2 -2 2 1 1 1 0 0 0 -1]
6 [16 0 0 0 -2 0 -2 0 0 1 0]
7 [ 5 1 1 -3 -1 1 2 -1 -1 0 0]
8 [10 2 -2 -2 1 -1 1 0 0 0 1]
9 [ 9 3 1 3 0 0 0 -1 1 -1 0]
10 [ 5 3 1 -1 2 0 -1 1 -1 0 -1]
11 [ 1 1 1 1 1 1 1 1 1 1 1]
```



This is a simple visualisation of the character table. The rows have the character values of representations as lists with the index that corresponds to conjugacy classes of the group  $\mathfrak{g}$ . We can easily note which row is associated with the sign representation, and which is associated with the trivial representation (it is the first and last rows, respectively). From the notation of the character table, we can then easily print out the list of character values for a specific partition. I.e., the sign representation can be printed out by writing `CT[0]`, where the list corresponding to the sign representation is printed out.

Though, knowing which character value is associated with which conjugacy class can be challenging. There is no clear indication of which column correlate to which conjugacy class of  $\mathfrak{g}$ .

Let's instead look at one of the functions in the GAP system. Particularly the function `gap(group).CharacterTable()`. This function is, as the code indicates, the function that prints out the character table to a group. The function has a lot of information in its table. Therefore, the next example will explain the output of the function `gap(group).CharacterTable()`, and how one can use the information that is provided. More explanation about this particular function can be found in the forum page provided by Zhao (2022).

► **Example 37.** To get the character table of the symmetric group of 6 elements, the code is

```
1 G = SymmetricGroup(6)
2 CT = gap(G).CharacterTable()
3 print(gap.eval("Display(%s)"%CT.name()))
```

We first ask for the symmetric group of 6 elements, and call this group `G`. After this, call for the character table of the group `G` by using the function `gap(G).CharacterTable()`. This character table is given the name `CT`. Lastly, evaluate the character table `CT` and print the display of this character table.

A shorter version, of the code is

```
1 G = SymmetricGroup(6)
2 gap.Display(gap.CharacterTable(G))
```

Here we display the character table of the group `G` directly, without naming the character table. Both `Display` and `CharacterTable()` are functions of GAP, this must be indicated in the code by writing `gap.` in front of them.

Both of these versions give out the same result. The output from the codes is the following set of tables;

```

1      CT1
2
3          2  4  4  4  4  1  1  1  3  3  .  1
4          3  2  1  .  1  2  1  2  .  .  .  1
5          5  1  .  .  .  .  .  .  .  .  1  .
6
7          1a 2a 2b 2c 3a 6a 3b 4a 4b 5a 6b
8      2P 1a 1a 1a 1a 3a 3a 3b 2b 2b 5a 3b
9      3P 1a 2a 2b 2c 1a 2a 1a 4a 4b 5a 2c
10     5P 1a 2a 2b 2c 3a 6a 3b 4a 4b 1a 6b
11
12     X.1      1 -1  1 -1  1 -1  1 -1  1  1 -1
13     X.2      5 -3  1  1  2  . -1 -1 -1  .  1
14     X.3      9 -3  1 -3  .  .  .  1  1 -1  .
15     X.4      5 -1  1  3 -1 -1  2  1 -1  .  .
16     X.5     10 -2 -2  2  1  1  1  .  .  . -1
17     X.6     16  .  .  . -2  . -2  .  .  1  .
18     X.7      5  1  1 -3 -1  1  2 -1 -1  .  .
19     X.8     10  2 -2 -2  1 -1  1  .  .  .  1
20     X.9      9  3  1  3  .  .  . -1  1 -1  .
21     X.10     5  3  1 -1  2  . -1  1 -1  . -1
22     X.11     1  1  1  1  1  1  1  1  1  1  1

```

Let us look further into this output, and try to understand the different tables that are displayed. Note that the table writes "." when the character value is equal to 0, and CT1 is just the name of the character table.

To best explain the output of the function, let us view the result in reverse order and start with the last table.

The last table displayed from the function `gap(G).CharacterTable()`, is the following table;

```

12     X.1      1 -1  1 -1  1 -1  1 -1  1  1 -1
13     X.2      5 -3  1  1  2  . -1 -1 -1  .  1
14     X.3      9 -3  1 -3  .  .  .  1  1 -1  .
15     X.4      5 -1  1  3 -1 -1  2  1 -1  .  .
16     X.5     10 -2 -2  2  1  1  1  .  .  . -1
17     X.6     16  .  .  . -2  . -2  .  .  1  .
18     X.7      5  1  1 -3 -1  1  2 -1 -1  .  .
19     X.8     10  2 -2 -2  1 -1  1  .  .  .  1
20     X.9      9  3  1  3  .  .  . -1  1 -1  .
21     X.10     5  3  1 -1  2  . -1  1 -1  . -1
22     X.11     1  1  1  1  1  1  1  1  1  1  1

```

This table is the character table for the group  $G$ . This table is the same as the character

table obtained from `G.character_table()` in Example 36. Now it also lists the character names on the side of the table.

The table above the character table is;

```

7      1a 2a 2b 2c 3a 6a 3b 4a 4b 5a 6b
8      2P 1a 1a 1a 1a 3a 3a 3b 2b 2b 5a 3b
9      3P 1a 2a 2b 2c 1a 2a 1a 4a 4b 5a 2c
10     5P 1a 2a 2b 2c 3a 6a 3b 4a 4b 1a 6b

```

By looking at the first row of the table we find the conjugacy classes of  $G$ . Meaning that the row;

```

7      1a 2a 2b 2c 3a 6a 3b 4a 4b 5a 6b

```

lists up all the conjugacy classes of  $G$  with their corresponding names. The row gives all the conjugacy classes a name, such that we can refer to them again. We will therefore refer to the conjugacy classes from these notations.

There are 11 conjugacy classes written in this row, which matches the amounts of characters listed. We have all the irreducible characters for this group, and therefore this character table is indeed valid.

The rest of the same table explains the correlations between elements of each conjugacy class. The elements  $c$  in the first column tell us what power we are looking at. Such that, each cell in this table explains where the  $c$ -th power of elements in the conjugacy class is located.

Let's look at the columns corresponding to the conjugacy class  $3a$  and  $4b$ . For  $a \in 3a$  and  $b \in 4b$ , the following information is given;

$c$	$a^c$	$b^c$
2	$a^2 \in 3a$	$b^2 \in 3b$
3	$a^3 \in 1a$	$b^3 \in 2c$
5	$a^5 \in 3a$	$b^5 \in 6b$

The first table in the output from `gap(G).CharacterTable()`, is the following table;

```

3      2  4  4  4  4  1  1  1  3  3  .  1
4      3  2  1  .  1  2  1  2  .  .  .  1
5      5  1  .  .  .  .  .  .  .  .  1  .

```

This table can be used to find the sizes of the different conjugacy classes of  $G$ .



The first column explains the prime divisors of  $|G|$ . The numbers in the other columns are corresponding to the exponents of the prime divisors in the product of the centraliser of the elements of each conjugacy class.

We can then find the order of the centraliser of each conjugacy class by raising the numbers in the first column by the numbers in the row corresponding to that of the conjugacy class and taking the product of them. Meaning that  $Z_C = 2^a \cdot 3^b \cdot 5^c$ , for some conjugacy class  $C$  with column entries  $a, b, c$ .

To find the size of a conjugacy class  $C_i$ , use Equation (2). For the conjugacy class  $3a$ , we can find the order of the centraliser  $Z_{3a}$  of its element

$$|Z_{3a}| = 2^1 \cdot 3^2 \cdot 5^0 = 18.$$

The size of the conjugacy class  $3a$  is then

$$\frac{|G|}{|Z_{3a}|} = \frac{6!}{18} = 40.$$

The order of the conjugacy class  $6a$  is

$$\frac{|G|}{|Z_{6a}|} = \frac{6!}{2^1 \cdot 3^1 \cdot 5^0} = 120$$



## 5.2 Murnaghan-Nakayama Recursive Algorithm

Before we present the algorithm for the Murnaghan-Nakayama rule, we need to define some functions that will be used in the algorithm.

Starting by defining a function for connected skew diagrams.

```

1 def skew_connected(partition,m):
2     # input: partition : the partition we are removing the border from
3     #         m         : the number of cells at the border strip
4     # output: list of connected skew-partitions
5     N = sum(partition)
6     M = N-m
7     if type(partition) == list:
8         partition = Partitions(N)(partition)
9     Par = Partitions(M)
10    Rim = partition.rim()
11    L = []
12    for p in Par:
13        if partition.contains(p):
14            test = 0
15            P = SkewPartition([partition,p])
16            if all(item in Rim for item in P.cells()):
17                place = [Rim.index(c) for c in P.cells() if c in Rim]
18                place.sort()
19                for i in range(len(place)-1):
20                    if place[i+1]-place[i] == 1:
21                        test = test + 1
22                else:
23                    test = 0
24                if test == len(place)-1:
25                    L.append(P)
26    return(L)

```

The function `skew_connected(partition,m)` creates a list of all the connected skew diagrams, where the input is the following: The `partition` we want to create a skew diagram from, and the number of boxes `m` the desired skew diagram contains.

First, we find the total number of boxes `N` which the `partition` is associated with. We then check if the `partition` is of `type(partition) = list`. If we have written the `partition` as a `list`, then we redefine this `partition` to be a partition type in SageMath. This is done by redefining the `partition` to be `Partitions(N)(partition)`.

After this, define a list of all the partitions `Par` of `M = N-m` boxes (the number of boxes that we need to remove to create the desired skew diagram). The `Skew` is defined as the list of coordinates of the border strip of the `partition`. Lastly, we define an empty list where all the connected skew diagrams will be listed at the end of this function.

Then, for all the partitions `p` in `Par`, we check if the `partition` contains each `p`, i.e. the partition `p` can be completely covered by `partition` if we were to place the diagram of `partition` on top of the diagram of `p`. If the `partition` does contain `p`, then we do the following test: First, create a skew partition, `P`, by removing `p` from `partition`. Then, check if all the

coordinates of the boxes in  $\mathfrak{p}$  is in  $\mathbf{skew}$ , the list of coordinates on the border of the diagram. If this is true, then we look at where the boxes are located in the border strip of the  $\mathfrak{partition}$ . If the boxes are next to each other, then we know that the skew diagram  $\mathfrak{p}$  is connected, and we add this skew diagram to the list  $L$ .

We keep doing the same test till we have looked over all possible partitions  $\mathfrak{p}$  of  $m$  boxes. We then have a complete list of all the connected skew diagrams of  $m$  boxes, that do not contain a  $2 \times 2$  box.

The next function needed to define is a function which finds the height of a skew diagram.

```
1 def skew_height(skew_partition):
2     # input : skew_partition : a skew diagram of the form "partition \ core"
3     #                                     or "[partition, core]"
4     # output: the height of the skew_partition as an integer
5     diff = []
6     for i in range(len(skew_partition[0])):
7         if i > len(skew_partition[1])-1:
8             if skew_partition[0][i] != 0:
9                 diff.append(skew_partition[0][i])
10            else:
11                if skew_partition[0][i]-skew_partition[1][i] != 0:
12                    diff.append(skew_partition[0][i]-skew_partition[1][i])
13    height = len(diff)-1
14    if height<0:
15        print("There are no skew diagram")
16    else:
17        return(height)
```

This function looks at a skew diagram `skew_partition`, and calculates the height of this diagram.

We start by creating an empty list, `diff`. This list will keep track of the number of boxes in each row there are in the skew diagram. If the `core` has a row that is of the same size as a row in the `partition`, then we know that the skew diagram does not have any boxes in this particular row. In this case, we do not include the row in the list `diff`.

After looking over all rows of the `partition`, we use the Equation (13) to calculate the height of the skew diagram. If there are no skew diagrams, and the `height` comes out as negative, then we return `"There are no skew diagram"`. Now, if the height is non-negative, then we give out the height as an output.

From these two functions, the algorithm of the Murnaghan-Nakayama rule can be derived. Or, more precisely, for the algorithm in Theorem 15.

Since we are looking at a recursive algorithm, we need to run the same function until the algorithm can no longer execute its calculations. The algorithm stops as soon as either; 1) there is no valid way to remove  $\alpha_i$  boxes from  $\lambda^{(i)}$ , or 2) we are left with an empty partition and an empty composition,  $\lambda^{(k)} = (0)$  and  $\alpha^{(k)} = (0)$ . For the first case, the associated character value is equal to zero. While, for the second case, the associated character value is  $\pm 1$ , depending on the sign.

The Murnaghan-Nakayama rule then has the following code in SageMath;

```

1  def MN_Rule(partition, composition, SUM = None):
2      # input : partition : a partition we wish to find the
3      #           character value of
4      #           composition: the composition we are finding the
5      #           character on (i.e. it indicates how many
6      #           cells we remove each step)
7      #           SUM = None : the sum of all the character values;
8      #           if left empty it means we start from
9      #           the beginning of the algorithm
10     # output: The sum of all the character values
11     N = sum(partition)
12     M = sum(composition)
13     if M != N:
14         raise IndexError ("The total number of boxes {} in the
15         composition {} does not correspond to the number of boxes {} in the
16         partition {}".format(M, composition, N, partition))
17     if type(partition) == list:
18         partition = Partitions(N)(partition)
19     if SUM is None:
20         SUM = 0
21     if len(composition) >= 1:
22         if skew_connected(partition, composition[0]) == []:
23             Character = 0
24             SUM = SUM + Character
25         else:
26             for Skew in skew_connected(partition, composition[0]):
27                 if Skew[1] == []:
28                     Character = (-1)**(skew_height(Skew))
29                 else:
30                     Character = (-1)**(skew_height(Skew)) * MN_Rule(Skew
31                     [1], composition[1:])
32             SUM = SUM + Character
33     return SUM

```

This function starts by looking at the sizes of both the partition and the composition.

The sizes need to be the same, or else the algorithm cannot be completed. The function will therefore print out an error code `IndexError` if these sizes are not the same. Now, if they are the same, the function will continue.

Let's look back at earlier examples and run the code over some of them, to see if the program gives out the same result as we calculated.

► **Example 38.** Let's look at the same example as in Example 29. Here we had  $\lambda = (5, 4, 3, 1)$ , and  $\alpha = (4, 3, 3, 2, 1)$ . We then have to run the following code;

```

1  lm = [5,4,3,1]
2  a = [4,3,3,2,1]
3
4  MN_Rule(lm,a)

```

By running the partition `lm` and composition `a` through `MN_Rule(lm,a)`, the corresponding character value is then

```

1  1

```

Going back to the result of Example 29, the character value was

$$\chi_{(4,3,3,2,1)}^{(5,4,3,1)} = 1,$$

which agrees with the code `MN_Rule(lm,a)`



Let us look at the other example in Section 4;

► **Example 39.** Going back to the same values as in Example 31, where  $\lambda = (5, 4, 2)$  and  $\alpha = (6, 3, 2)$ . Here we write the code;

```

1  lm = [5,4,2]
2  a = [6,3,2]
3
4  MN_Rule(lm,a)

```

Now by running these lines of code, we get that the character value is

```

1  0

```

which again agrees with what we got as a result in the example,  $\chi_{(6,3,2)}^{(5,4,2)} = 0$ .



The code, `MN_rule()`, can also be used to calculate the character value when looking over the special cases.

► **Example 40.** Let's look at the special cases on the partition  $\lambda = (5, 4, 3, 1)$ .

Starting by looking at the case when we are looking at the trivial representation,  $\alpha = (n)$ . Define `a` to be `[sum(lm)]`, i.e. equal to the list with one entry equal to the integer that the partition `lm` is associated to.

```

1  lm = [5,4,3,1]
2  a = [sum(lm)]
3
4  MN_Rule(lm,a)

```

By running this code, the associated character value is then;

```

5  0

```

Recall from Lemma 1, the character value of  $\chi_{(n)}^\lambda$  is  $(-1)^{n-\lambda_1}$  if  $\lambda$  is a *hook*, and 0 otherwise. Since `lm` is not a hook, then  $\chi_{(n)}^\lambda = 0$ .

The next case is when we are looking at the sign representation,  $\alpha = (1^n)$ . Here `a` is generated to be of length `sum(lm)`, where the entries are all 1. We then have both the partition `lm` and composition `a` ready for the algorithm of the recursive Murnaghan-Nakayama rule;

```

1  lm = [5,4,3,1]
2  a = []
3  # adding sum(lm) 1's as entries for the composition a
4  while len(a) < sum(lm):
5      a.append(1)
6
7  f_lm = (factorial(13))/(8*6*6*5*4*4*3*3*2)
8  print("There are",f_lm,"SYT of shape",lm)
9
10 MN = MN_Rule(lm,a);MN
11 f_lm == MN

```

From the special case in Lemma 3, the character value  $\chi_{(1^n)}^\lambda = f^\lambda$ . We then generate `f_lm`, which is the number of standard young tableaux of shape `lm`, i.e.  $f^\lambda$ . We then run the algorithm and check if the number of standard Young tableaux is equal to the character value, `f_lm == MN`.

The result is the following;

```
1   There are 15015 SYT of shape [5, 4, 3, 1]
2   15015
3   True
```

The number of standard young tableaux of shape  $1_m$ , is the same as the character value associated to  $1_m$  at  $\mathbf{a}$ .





### 5.3 Comparison Of Functions

Now that we have different ways to calculate the character value, we should question ourselves on which method is the most convenient. The function should be quick to compute, and easy to understand.

In this section we will look at the functions `MN_Rule()`, `character_table()`, and `gap.Irr()`, where we compare the time it takes each function to calculate the character value. Due to the function `to_character()` using a lot of the memory of SageMath when running over big symmetry groups, we choose to not include this function in the comparison.

To figure out the time each function took to compute the results, the following code can be used:

```
import time

start_time = time.time()

[...]

end_time = time.time()
elapsed_time = end_time - start_time
print("Elapsed time: ", elapsed_time)
```

Here, the normal calculation on the line is done in the line noted with `[...]`. The time is noted before and after the function has worked through the necessary calculations. With the function `time.time()`, we can figure out the exact time that the program started and at what time it ended. From these times, the total time it took to do all the necessary calculations for the different functions can also be found.

Before we start comparing the functions, we need to note that the two predefined functions, `character_table()` and `gap.Irr()`, gives us an output with all the irreducible characters of a given group  $G$ . To make it fairer, let us first try to find how long it takes to find just a single value. After this, we can figure out how long it would take to find all the character values of a representation. Lastly, we figure out how long each function takes to calculate all of the character values for a whole group.

### 5.3.1 Single Character value

To find the character value of a representation on a partition, two partitions need to be computed. So, before any of the calculations, we run the following code.

```
1 import time
2 import random
3
4 n = 12
5 lm = random.choice(Partitions(n));lm
6 a = random.choice(Partitions(n));a
```

Here, the program chooses two random partitions of  $n = 12$ , such that the comparison is not in favour of any particular function.

We will here look at the following partitions

```
[5, 3, 1, 1, 1, 1]
[7, 4, 1]
```

Such that the same partition  $lm = [5,3,1,1,1,1]$  and composition  $a = [7, 4, 1]$  will be used for all the different functions. Note that we call  $a$  a composition even though we got it from the set of partitions of  $n$ . We do this so that we can easier compare the functions, and since the order of a composition does not matter.

Let us start by looking at the algorithm for the Murnaghan-Nakayama rule, which was defined in Section 5.2:

```
9 start_time = time.time()
10
11 chi_MN = MN_Rule(lm,a)
12
13 end_time = time.time()
14 elapsed_time_MN = end_time - start_time
15
16 print("Character from MN: ",chi_MN)
17 print("Elapsed time: ", elapsed_time_MN)
```

```
"Character from MN: " 1
"Elapsed time: " 0.004169940948486328
```

From this algorithm, the character value of  $lm = [5,3,1,1,1,1]$  on the composition  $a = [7, 4, 1]$  is then

$$\chi_{(7,4,1)}^{(5,3,1,1,1,1)} = 1$$

Next, let us find the character value from the character table of the group.

```
9     start_time = time.time()
10
11     G = SymmetricGroup(n)
12     CT = G.character_table()
13     S = Partitions(n)
14
15     for i in range(len(S)):
16         if S[-i] == lm:
17             Chi = CT[i-1]
18
19     for i in range(len(S)):
20         if S[-i] == a:
21             chi_CT = Chi[i-1]
22
23     end_time = time.time()
24     elapsed_time_CT = end_time - start_time
25
26     print("Character from CT: ", chi_CT)
27     print("Elapsed time: ", elapsed_time_CT)
```

```
"Character from CT: " 1
"Elapsed time: " 0.9328291416168213
```

Note that we here need to first find the correct row that is equal to  $1m$  in the character table  $CT$ , and then find the correct entry in this row which is associated with  $a$ . Thus, we have to go through more steps to find the correct value.

Lastly, to find the character value from the function defined in GAP (2021), we use the following code:

```

9     start_time = time.time()
10
11     G = SymmetricGroup(n);
12     S = Partitions(n);
13     Char = gap.Irr(G);
14
15     for i in range(len(S)):
16         if S[-i] == lm:
17             if i == 0:
18                 Chi = Char[i+1]
19             else:
20                 Chi = Char[i]
21
22     for i in range(len(S)):
23         if S[-i] == a:
24             if i == 0:
25                 chi_gap = Chi[i+1]
26             else:
27                 chi_gap = Chi[i]
28
29     end_time = time.time()
30     elapsed_time_gap = end_time - start_time
31
32     print("Character from GAP: ", chi_gap)
33     print("Elapsed time: ", elapsed_time_gap)

```

```

"Character from GAP: " 1
"Elapsed time: " 1.7654774188995361

```

Note again that we need to find the correct row and entry that are associated with `lm` and `a`, respectively. Which again results in a longer process of finding the correct character value.

From these results, we can see that the Murnaghan-Nakayama algorithm `MN_Rule()` is faster than the two other functions when it comes to finding the character value of a partition on a specific composition. Though, the other functions are not taking too much time to compute the character values themselves.

### 5.3.2 List of Character Values

Now let us instead try to find all the character values of a partition of a specific number  $n$ .

We start by running the following code as before, but here we only need one partition.

```
1 import time
2 import random
3
4 n = 12
5 lm = random.choice(Partitions(n)); lm
```

```
[5, 4, 3, 1]
```

We have quite a lot of conjugacy classes for the symmetric group of  $n = 12$  elements, such that the list of character values is quite long. We will therefore shorten the outputs that we get, for convenience's sake.

For the algorithm of the Murnaghan-Nakayama rule, we need to make a list of all the results where we run over all the possible partitions of  $n$ :

```
7 start_time = time.time()
8
9 Chi_MN = []
10 for a in Partitions(n):
11     Chi_MN.append(MN_Rule(lm,a))
12 Chi_MN.reverse()
13
14 end_time = time.time()
15 elapsed_time_MN = end_time - start_time
16
17 print("Character from MN: ",Chi_MN)
18 print("Elapsed time: ", elapsed_time_MN)
```

We then get the following output;

```
"Character from MN: " [15015, 1155, 147, 7, -17, -5, -5, -735, ... , 0,
0, 0, 0, 0]
"Elapsed time: " 17.21722388267517
```

Note that this algorithm takes quite a long time to find all the character values since it has to run through  $n! = 12!$  elements to find all the characters. We can already here hypothesise that this algorithm is not going to be the best function for finding the list of character values.

To confirm this we need to check how much time the other two function takes.

The function `character_table()` with the following syntax

```
7   start_time = time.time()
8
9   G = SymmetricGroup(n)
10  CT = G.character_table()
11
12  G = SymmetricGroup(n)
13  S = Partitions(n)
14  for i in range(len(S)):
15      if S[-i] == lm:
16          Chi_CT = CT[i-1]
17
18  end_time = time.time()
19  elapsed_time_CT = end_time - start_time
20
21  print("Character from CT: ",Chi_CT)
22  print("Elapsed time: ", elapsed_time_CT)
```

This gives the output

```
"Character from CT: " (15015, 1155, 147, 7, -17, -5, -5, -735, ... , 0,
0, 0, 0, 0)
"Elapsed time: " 1.9321322441101074
```

Lastly, the function `gap.Irr()`

```
7   start_time = time.time()
8
9   G = SymmetricGroup(n);
10  S = Partitions(n);
11  Char = gap.Irr(G);
12
13  for i in range(len(S)):
14      if S[-i] == lm:
15          Chi_gap = Char[i+1]
16
17  end_time = time.time()
18  elapsed_time_gap = end_time - start_time
19
20  print("Character from GAP: ",Chi_gap)
21  print("Elapsed time: ", elapsed_time_gap)
```

```
"Character from GAP: " Character( CharacterTable( SymmetricGroup( [ 1 ..
13 ] ) ),
[15015, 1155, 147, 7, -17, -5, -5, -735, ... , 0, 0, 0, 0, 0] )
"Elapsed time: " 1.9053983688354492
```

The preferred function to use in this certain problem would be the function on GAP (2021), `gap.Irr()`, with the character table close after. While the function `MN_Rule()`, which took 17 seconds to compute the list of character values, is not very efficient in this case.

### 5.3.3 Character table

The last thing to consider is the computation of character tables. Since the function `character_table()` is specifically made to compute this, we need to consider also this scenario.

When we look at the character tables, let us consider a smaller number of elements. We do this so that we don't use up all the memory in the online software of SageMath.

The code is then

```

1  import time
2  import random
3
4  n = 8

```

The function `MN_Rule()`, needs to run over all the partitions of  $n = 8$ , two times. Both to find the partition that we want to find the character value of and to find the composition that we want to look over. The code is therefore:

```

5  start_time = time.time()
6
7  CT_MN = []
8  for a in Partitions(n):
9      Chi_MN = []
10     for b in Partitions(n):
11         Chi_MN.append(MN_Rule(a,b))
12     CT_MN.append(Chi_MN)
13
14  end_time = time.time()
15  elapsed_time_MN = end_time - start_time
16
17  #print("character from MN: ",CT_MN)
18  print("Elapsed time: ", elapsed_time_MN)

```

Note that we will not print out the character table, and instead only ask for the time it takes to compute the function. The possibility of printing out the character table is in the code but is commented out, for the sake of saving space in the Master's thesis.

We get the following output

```
"Elapsed time: " 2.0910394191741943
```

The time it takes to generate the character table when using the function `character_table()`, is the following:

```
5 start_time = time.time()
6
7 G = SymmetricGroup(n)
8 CT = G.character_table()
9
10 end_time = time.time()
11 elapsed_time_CT = end_time - start_time
12
13 #print("character from CT: ",CT)
14 print("Elapsed time: ", elapsed_time_CT)
```

```
"Elapsed time: " 0.021608829498291016
```

This function is fast in finding the character table.

Finally, let's look at the time it takes for the function `gap.Irr()` to generate the character table:

```
5 start_time = time.time()
6
7 G = SymmetricGroup(n);
8 Char = gap.Irr(G);
9
10 CT_gap = []
11 for c in Char:
12     Chi_gap = []
13     for a in c:
14         Chi_gap.append(a)
15     CT_gap.append(Chi_gap)
16
17 end_time = time.time()
18 elapsed_time_gap = end_time - start_time
```

```
"Elapsed time: " 0.8707418441772461
```

From these results, we see that `character_table()` is the quickest, followed by `gap.Irr()`.



### 5.3.4 Result

To easier see which function is the fastest, let us run the above comparisons for different numbers of elements in the symmetric group. We can then insert the time it took for each function to compute the character values in a table, and highlight which function worked the best.

We then get the following table, where the last row states which function was the fastest overall. The columns are different ways we compared functions; finding the character value of a partition associated with another partition, finding the list of characters of a specific

partition, and finding the character table.

	Character Value	List of Characters	Character Table
n=1	<b>MN 0.0019</b> <i>CT</i> 0.6152 <i>GAP</i> 2.1340	<b>MN 0.0007</b> <i>CT</i> 0.0067 <i>GAP</i> 0.0039	<b>MN 0.0009</b> <i>CT</i> 0.0052 <i>GAP</i> 0.0068
n=2	<b>MN 0.0014</b> <i>CT</i> 0.0094 <i>GAP</i> 0.0049	<b>MN 0.0011</b> <i>CT</i> 0.0062 <i>GAP</i> 0.0037	<b>MN 0.0018</b> <i>CT</i> 0.0041 <i>GAP</i> 0.0102
n=3	<b>MN 0.0011</b> <i>CT</i> 0.0080 <i>GAP</i> 0.0071	<b>MN 0.0014</b> <i>CT</i> 0.0059 <i>GAP</i> 0.0041	<b>MN 0.0038</b> <i>CT</i> 0.0073 <i>GAP</i> 0.0199
n=4	<b>MN 0.0009</b> <i>CT</i> 0.0068 <i>GAP</i> 0.0049	<i>MN</i> 0.0058 <i>CT</i> 0.8716 <b>GAP 0.0040</b>	<b>MN 0.0091</b> <i>CT</i> 0.0057 <i>GAP</i> 0.0202
n=5	<b>MN 0.0013</b> <i>CT</i> 0.0458 <i>GAP</i> 0.0049	<b>MN 0.0058</b> <i>CT</i> 0.8716 <i>GAP</i> 2.6637	<i>MN</i> 0.0248 <b>CT 0.0051</b> <i>GAP</i> 0.0672
n=6	<b>MN 0.0011</b> <i>CT</i> 0.0107 <i>GAP</i> 0.0045	<i>MN</i> 0.0083 <i>CT</i> 0.0180 <b>GAP 0.0079</b>	<i>MN</i> 0.1199 <b>CT 0.0073</b> <i>GAP</i> 0.2561
n=7	<i>MN</i> 0.0163 <b>CT 0.0153</b> <i>GAP</i> 0.0198	<i>MN</i> 0.0416 <i>CT</i> 0.0295 <b>GAP 0.0106</b>	<i>MN</i> 0.4472 <b>CT 0.0142</b> <i>GAP</i> 0.3694
n=8	<b>MN 0.0008</b> <i>CT</i> 0.0292 <i>GAP</i> 0.0059	<i>MN</i> 0.0580 <i>CT</i> 0.0747 <b>GAP 0.0061</b>	<i>MN</i> 1.6456 <b>CT 0.0829</b> <i>GAP</i> 0.6801
n=9	<b>MN 0.0009</b> <i>CT</i> 0.0761 <i>GAP</i> 0.0064	<b>MN 0.1276</b> <i>CT</i> 0.2784 <i>GAP</i> 0.8568	<i>MN</i> 5.3617 <b>CT 0.0865</b> <i>GAP</i> 1.0204
n=10	<b>MN 0.0023</b> <i>CT</i> 0.1310 <i>GAP</i> 0.0079	<i>MN</i> 0.3902 <i>CT</i> 0.1884 <b>GAP 0.0077</b>	<i>MN</i> 19.3558 <b>CT 0.1175</b> <i>GAP</i> 2.3749
Oft quick	<i>MN</i>	<i>GAP</i> and <i>MN</i>	<i>CT</i>
Least average	<i>MN</i>	<i>MN</i>	<i>CT</i>

(43)

The last two rows give a summary of which function was the quickest, most often and on average.

Each of the functions seems to work in its own defined areas, with some variability. The

algorithm for the Murnaghan-Nakayama rule, `MN_Rule()`, is often the fastest when it comes to finding character values of a partition associated with another partition. The function from the system GAP (2021), `gap.Irr()`, is fast to find the list of character values of a partition as often as `MN_Rule()`. While the function `character_table()` is the most often fastest to find the character table of a group.

On average, the `MN_Rule()` seems to be the fastest in finding single character values and a list of character values. While `ct` is the favourite when it comes to finding character tables. One of the reasons for this might be that we calculated `MN_Rule()` first every time, meaning that we had less memory available when calculating the other functions.

By looking at the array in Equation (43), we can observe that the time in which `MN_Rule()` needs to calculate the character value increases drastically when the value of  $n$  grows. We can therefore hypothesise that `MN_Rule()` will aggravate with big  $n$ .

## 6 Conclusion

The Murnaghan-Nakayama rule is a combinatorial algorithm to calculate the character value of a partition of a specific composition. The recursive version of the Murnaghan-Nakayama rule is;

**Theorem 15.** *Let  $\lambda \vdash n$ , and suppose that  $\alpha = (\alpha_1, \dots, \alpha_m)$  is a composition of  $n$ . Then*

$$\chi_\alpha^\lambda = \chi^\lambda(\alpha) = \sum_{\nu} (-1)^{ht(\nu)} \chi^{\lambda \setminus \nu}(\alpha \setminus \alpha) \quad (33)$$

where the sum goes over all border strips  $\nu$  of size  $\alpha_1$ .

With this algorithm, the character values of the partition can be found by reducing the diagrams associated with the partition. This is done by looking at each part of the composition and removing this amount from the original partition. See Example 29 for a more detailed explanation.

This algorithmic rule has been further processed for different function families and scenarios. The Murnaghan-Nakayama rule is therefore an adaptable rule, which further amplifies the importance of the rule.

The code for the algorithm of the Murnaghan-Nakayama rule runs its calculations quickly when comes to finding character values of partitions associated with a composition. When  $n \leq 5$ , it can also find the list of character values of a partition, and character table of  $S_n$  quickly. When  $n$  is greater in size, the time it takes to find all the character values of the character table increases, and the algorithm is no longer very efficient.

## References

- Alexandersson, P. (2022). The Murnaghan-Nakayama rule. <https://www.symmetricfunctions.com/murnaghanNakayama.htm>. Accessed: 2 November 2022.
- Ben (2022). Constructing character table given group size, conjugacy class size and 3 characters. <https://math.stackexchange.com/q/4499785>. Accessed on 25 April 2023.
- Chow, T. Y. and Paulhus, J. (2020). Algorithmically distinguishing irreducible characters of the symmetric group. <https://arxiv.org/abs/2006.00035>.
- Constantine, G. (1998). Sylow's theorems. <https://sites.pitt.edu/~gmc/ch1/node6.html>. Accessed: 11 April 2023.
- Désarménien, J., Leclerc, B., and Thibon, J. (2000). Hall-littlewood functions and kostka-foulkes polynomials in representation theory. *Lotharingien Combin*, 32.
- Fulton, W. and Harris, J. (2004). *Representation Theory*. Graduate Texts in Mathematics. Springer New York, NY. DOI 10.1007/978-1-4612-0979-9.
- GAP (2021). *GAP – Groups, Algorithms, and Programming, Version 4.11.1*. The GAP Group.
- Griffiths, M. and Lord, N. (2011). The hook-length formula and generalised Catalan numbers. <http://www.jstor.org/stable/23248614>. Accessed: 18 April 2023.
- Grillet, P. A. (2007). *Abstract Algebra*. Graduate Texts in Mathematics. Springer New York, NY, 2 edition. DOI 10.1007/978-0-387-71568-1.
- Hamaker, Z. and Rhoades, B. (2022). Characters of local and regular permutation statistics. <https://arxiv.org/abs/2206.06567>.
- Jackson, D. M. and Sloss, C. A. (2011). Near-central Permutation Factorization and Strahov's Generalized Murnaghan-Nakayama Rule. <https://arxiv.org/abs/1108.4047>.
- James, G. (1978). *Representation Theory of the Symmetric Groups*. Lecture Notes in Mathematics. Springer Berlin, Heidelberg. DOI 10.1007/BFb0067708.
- Konvalinka, M. (2011). Skew Quantum Murnaghan-Nakayama Rule. <https://arxiv.org/abs/1101.5250v1>.
- Lewis, D. (2003). 'To the Glory of God, Honour of Ireland and Fame of America' : A Biographical Sketch of Francis D. Murnaghan. *Mathematical Proceedings of The Royal Irish Academy*, 103:101–112. DOI 10.3318/PRIA.2003.103.1.101.

- Littlewood, D. E. and Richardson, A. R. (1934). Group Characters and Algebra. <http://www.jstor.org/stable/91293>. Accessed: 12 April 2023.
- Loehr, N. A. (2011). *Bijective Combinatorics*. Discrete Mathematics and its Applications. Taylor & Francis Inc.
- maycontainmaths (2014). P vs NP for dummies: Part 1. <https://maycontainmaths.wordpress.com/2014/12/21/p-vs-np-for-dummies-part-1/>. Accessed: 03 April 2023.
- Mehlhorn, K. and Sun, H. (2013). P versus NP, and More. <https://resources.mpi-inf.mpg.de/departments/d1/teaching/ss14/gitcs/>. Accessed: 04 April 2023.
- Murnaghan, F. (1937). On the Representations of the Symmetric Group. *American Journal of Mathematics*, 59(3):437–488.
- Nakayama, T. (1940). On some modular properties of irreducible representations of a symmetric group. I. *Jap. J. Math.*, 18. DOI 10.4099/jjm1924.17.0\_165.
- Pak, I. and Panova, G. (2015). On the complexity of computing Kronecker coefficients. <https://arxiv.org/abs/1404.0653>.
- Sagan, B. E. (2001). *The Symmetric group: Representations, Combinatorial Algorithms, and Symmetric Functions*. Graduate Texts in Mathematics. Springer, 2 edition.
- Stanley, R. P. and Fomin, S. (1999). *Symmetric Functions*, volume 2 of *Cambridge Studies in Advanced Mathematics*, page 286–560. Cambridge University Press. DOI 10.1017/CBO9780511609589.006.
- The Sage Developers (2022). *SageMath, the Sage Mathematics Software System*. DOI 10.5281/zenodo.6259615.
- Tubbenhauer, D. (2022). What is...representation theory? <https://youtube.com/playlist?list=PLuFcVFHMIfhJzL5A5gXRotRiXaY5vuDc->. Accessed: 27 March 2023.
- Valiant, L. G. (1979). The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421.
- Wikipedia contributors (2023a). Np-completeness — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=NP-completeness&oldid=1151510793>. Accessed on 11 May 2023.
- Wikipedia contributors (2023b). Plethysm — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=Plethysm&oldid=1146939586>. Accessed 14 May 2023.

Wildon, M. (2015). A Combinatorial Proof of a Plethystic Murnaghan-Nakayama Rule. <https://arxiv.org/abs/1408.3554v2>.

Zhao, H. (2022). [GAP Forum] Print the irreducible characters of the normal ordinary character table format. <https://www.gap-system.org/ForumArchive2/2022/006387.html>. Accessed; 28 March 2023.





