MPhys Project Report: Computational Group Theory

Lawrence Arscott

Supervisor: Anthony Kennedy

March 28, 2022

Abstract

The focus of this papers is on the scalars that arise during group theoretical calculations, the Wigner 3n-j symbols. In this paper, we work towards efficient numerical computation of what we call '3-j's and '6-j's. Closely related to Wigner 3-j and 6-j symbols, 3-js and 6-js can be used to compute any Wigner 3n-j symbol. As such, they contain all the necessary information to deduce the physical consequences of a symmetry. The reduction of high order terms involving many 3js and 6js, it is worth investigating how we may compute these terms efficiently.

Contents

1 Personal statement

The project had a steep initial learning curve. I had to get to grips with new notation in the form of birdtracks while consolidating and furthering my understanding of representation theory, particularly with regards to the symmetric group. I spent much time studying chapters 7 (The Symmetric Group) and 10 (Linear groups in n-dimensional space, Irreducible tensors) of ref.[4]. This was a great help in understanding the background to ref.[3] which expected a high degree of mathematical maturity on my part. Ref.[8] was also very useful in understanding the representation theory of $S(n)$, particularly with regards to the Garnir algorithm, although I had to be wary of differing conventions.

Having now rewritten my code several times, I feel the biggest improvements I made have been in my coding skills, I notice this especially when looking back at my first attempts. I learnt to program much more patiently and built habits such as regularly implementing thorough checks which made debugging much easier. I also had to make sure I had a genuine understanding of the background theory so as to be sure incorrect results were due to coding errors.

In the end I was able to reach the target of successfully calculating 3js and 6js by computer algorithm using representations of the symmetric group. I have thoroughly enjoyed the mix of careful theoretical study with the opportunity to implement results in computer programs.

Many thanks go to my supervisor Tony, who is passionate about his area of expertise, making it a privilege to learn from him. I would like to thank him for taking the time to format complete LaTeXed answers to my questions during the week, even replying late on the night before my deadline. I would also like to thank Hugh for joining me in overthinking upper, lower, left and right indices instead of learning the material.

2 Introduction

This paper is a first attempt to compute Hermitian 3j and 6j symbols for the unitary group $U(n)$ using representations of the symmetric group algebra. The text assumes some background in the basics of representation theory. Much of the motivation for, and background to, this paper is contained within ref.[3].

In this paper, we will go over the necessary background for grasping the motivation to this paper. In particular, to those unfamiliar, we will briefly introduce birdtrack notation. Birdtracks convey tensor expressions in diagrammatic form. This greatly simplifies the communication of expressions involving many index contractions. Several proofs in this text that may be painstaking to follow in conventional tensor notation seem almost trivial in birdtrack notation.

The background is introduced so that the reader may understand the recoupling relation, and how this allows us to express any Wigner 3n-j symbol in terms of 3js and 6js. The Wigner-Eckart theorem is given as a powerful example of the benefit of calculating 3js and 6js.

Following on, we will introduce further background necessary to understanding what we will call "Elvang's sandwich theorem". This involves the introduction of symmetrisers, antisymmetrisers and Young projection operators. From these objects, we construct the representations of the symmetric group algebra. The limited number of ways Young projectors can connect, a property we will call uniqueness, is the property behind the sandwich theorem. This result, along with the Garnir algorithm, to which we give a proof in birdtrack notation, forms the backbone to our approach to calculating $U(n)$ 3js and 6js.

Accompanying code may be found by following the link:
`https://github.com/IAmAnEvilBunny/Computational-Group-Theory`

3 Background

3.1 Birdtrack notation

Birdtrack notation is introduced to simplify tensor expressions. The use of birdtracks greatly simplifies expressions that would otherwise be chock full of indices, and makes the proof of several properties used throughout the text immediate.

I will demonstrate how birdtrack notation works when working on any vector spaces of the form:

$$\boxed{V^{\otimes n} \otimes \bar{V}^{\otimes m}}$$

where $\bar{V}$ is the dual space to V.

I will denote components of $v \in V$ with lower indices v_i so that $v = v_i e^i$ and those of vectors in the dual space $u \in \bar{V}$ with upper indices u^j. In birdtrack notation, an upper index is represented as an arrow pointing away from the vector. A lower index is represented by an arrow pointing towards:

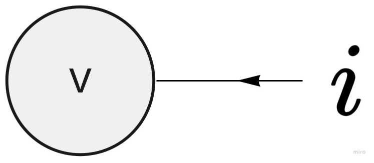


Figure 1: v_i in birdtrack notation

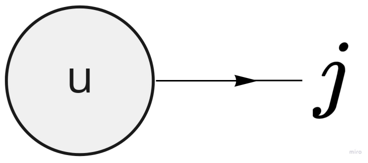


Figure 2: u^j in birdtrack notation

We may contract upper and lower indices $u(v) = u_i v^i$, which in birdtrack notation we display as:

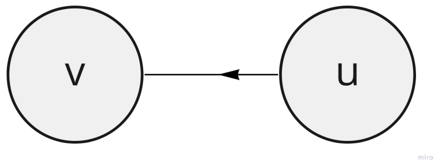


Figure 3: Index contraction in birdtrack notation

As mentioned, the arrow pointing from u to v tells us that the index of u being contracted is in the dual space, while the index of v is in the base space. This extends to tensors with several indices:

$$T \in V^{\otimes n} \otimes \bar{V}^{\otimes m}, \quad U \in \overline{V^{\otimes n} \otimes \bar{V}^{\otimes m}} \cong V^{\otimes m} \otimes \bar{V}^{\otimes n}$$

$$T^{b_1,\dots,b_m}_{a_1,\dots,a_n} U^{a_1,\dots,a_n}_{b_1,\dots,b_m} \qquad \text{is given by:}$$

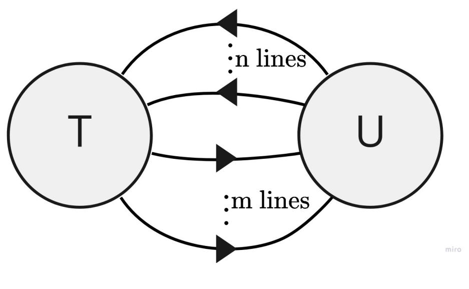


Figure 4: Contraction of tensors T and U

However we will often write a single line to represent all upper indices, and a single line for all lower indices.

Matrices are elements of $(V^{\otimes n} \otimes \bar{V}^{\otimes m}) \otimes (V^{\otimes m} \otimes \bar{V}^{\otimes n})$, so that matrix multiplication is defined. In birdtrack notation:

$$(M_1)_{a}{}^{b}{}_{,d}{}^{c}\,(M_2)_{c}{}^{d}{}_{,f}{}^{e} =$$

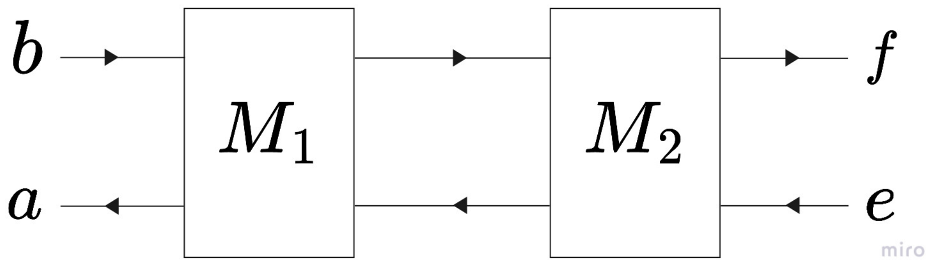


Figure 5: Matrix multiplication

The last object I will introduce is the Kronecker delta. This is simply represented by isolated lines. For example, the expression $M^2 = \mathbb{1}$ is given component-wise by:

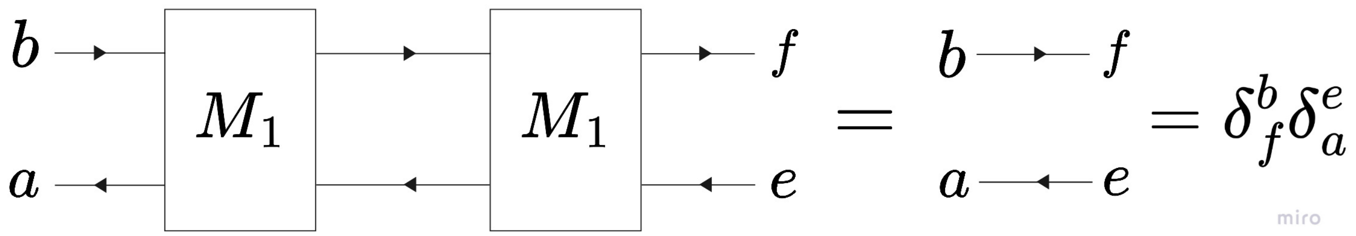


Figure 6: Identity matrix

Finally, given a set of matrices $\{M_i\}$, we may want to express an orthonormality relation $M_i \circ M_j = \delta_{ij}\mathbb{1}$. We do this by adding the i, j indices above the Kronecker delta. In the following, we suppress indices as we will often do throughout the text.

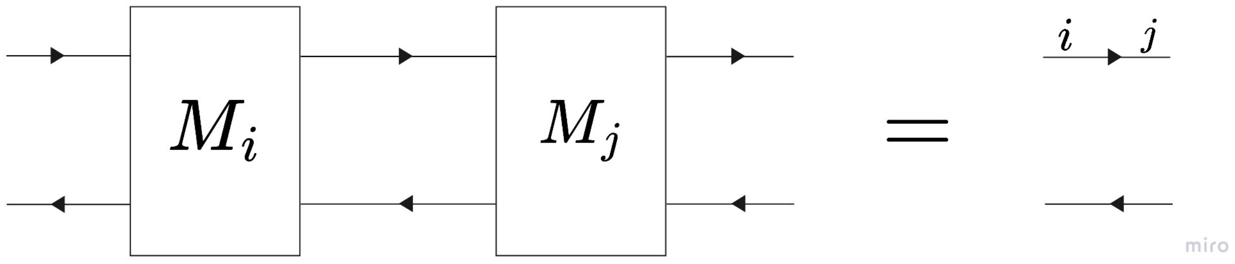


Figure 7: Orthonormality relation

For a more complete guide on birdtrack notation, see for example ref.[6], although be wary of differing conventions.

3.2 The defining representation

Let let the group G act on the vector space V. We will call this representation the **defining representation** $\rho(G)$. This automatically defines a **dual representation** $\bar{\rho}(G)$ on $\bar{V}$ (for which the representation matrices are the transposes of the inverses), and so the defining representation defines a group action on the vector space $V^{\otimes n} \otimes \bar{V}^{\otimes m}$.

The defining action on V decomposes V into irreducible subspaces, with each irreducible subspace furnishing an irreducible representation of G:

$$\rho_V(G) = \rho_1(G) \oplus ... \oplus \rho_n(G)$$

A similar decomposition is induced on $\bar{V}$:

$$\bar{\rho}_V(G) = \bar{\rho}_1(G) \oplus ... \oplus \bar{\rho}_n(G)$$

Projection from V onto the irreducible subspaces is given by a complete set of projection operators $P_1, ..., P_n$ which satisfy:

- Orthonormality
$$P_i P_j = \delta_{ij} P_i^2 \tag{1}$$
- Idempotency
$$P_i^2 = P_i \tag{2}$$
- The completeness relation
$$\sum_i P_i = \mathbb{1} \tag{3}$$

In fact, as we will see, it is helpful to think of projection operators as equivalent to idempotents (see sec.(A) and also ref.[4]). Orthonormality can be seen as proof the projection operators project onto different subspaces, and the completeness relation as a way to express the fact a complete set fully decomposes the vector space.

Remark 3.1 (U(n)).
The work in this text will be carried out in U(n), with $V = \mathbb{C}$, and so we may raise and lower indices by taking the complex conjugate:

$$u^i = (u_i)^*$$

I briefly mentioned earlier that for a representing matrix $\rho(g), g \in G$ on V, the representing matrix in the dual representation $\bar{\rho}(g)$ (which acts on $\bar{V}$) is the inverse transpose. In our case, the representation is unitary, and so

$$\rho(g) = (\bar{\rho}(g))^*$$

3.3 Clebsches, 3js, 6js

3.3.1 Clebsches

Given representations μ, ν, ρ of G, the Clebsch-Gordan coefficients (I will sometimes refer to these as Clebsches) are the coefficients in the expansion of a basis vector in the tensor product space $\mu \otimes \nu$ into the basis vectors in the representing space of ρ:

$$u_i \otimes v_i = \sum_j \left\langle \mu, u^i, \nu, v^i; \rho, p_j \right\rangle p_j \qquad \longrightarrow \quad \text{the } \left\langle \mu, u^i, \nu, v^i; \rho, p_j \right\rangle \text{ are the Clebsch-Gordan coefficients}$$

Clebsch-Gordan coefficients encode how basis vectors of a tensor product of two representation spaces translate to those of a third representation space, usually an irrep.

Consider the Clebsch-Gordan coefficient projecting $V^{\otimes n} \otimes \bar{V}^{\otimes m}$ onto the irrep ρ:

$$\langle p^k | e^{i_1, i_2, \ldots, i_n} \otimes f_{j_1, \ldots, j^m} \rangle \tag{4}$$

where $e^{i_1, i_2, \ldots, i_n} = e^{i_1} \otimes ... \otimes e^{i_n}$ and I have omitted the representations.

This is represented in birdtrack notation as the below image on the left:

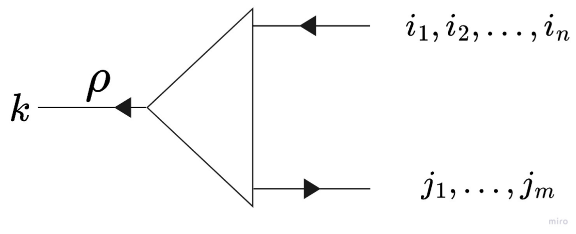


Figure 8: Clebsch of eq.(4) in birdtrack notation

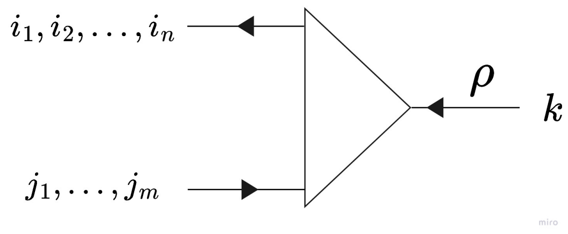


Figure 9: Conjugate Clebsch

The ρ above the line leading to k specifies the k index is in the representation ρ. I have omitted the labels for the defining representation and the dual representation.

The image on the right (9) is the conjugate, obtained by taking a mirror image that does not affect arrows

Clebsches give back projection operators by contraction with their conjugate:

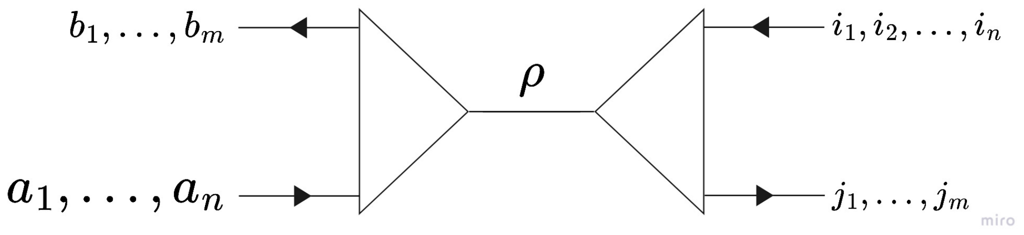


Figure 10: Projection operator P_ρ reconstructed from Clebsch-Gordan coefficients

and so we have a completeness relation (eq. 3):

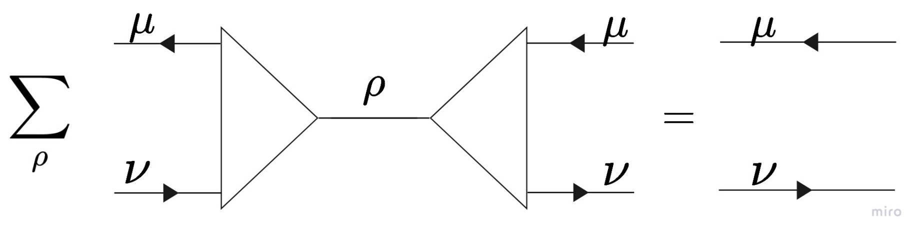


Figure 11: Completeness relation

which has been written for arbitrary representations. Indices have been suppressed. Remember the object on the right is a delta function.

We also get an orthonormality relation (eq. 1):

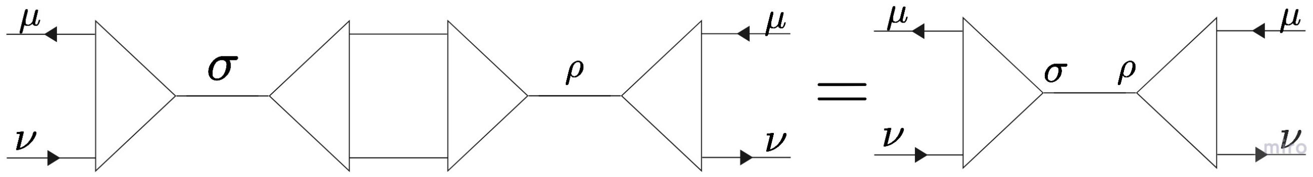


Figure 12: Orthogonality relation in birdtrack notation

which essentially expresses:

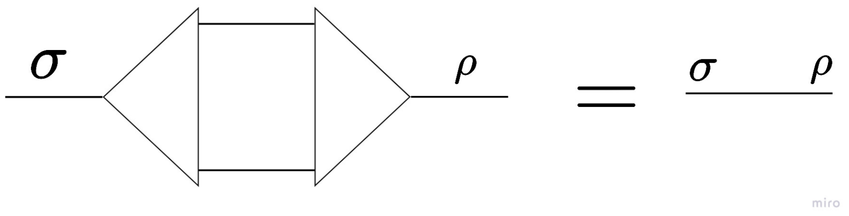
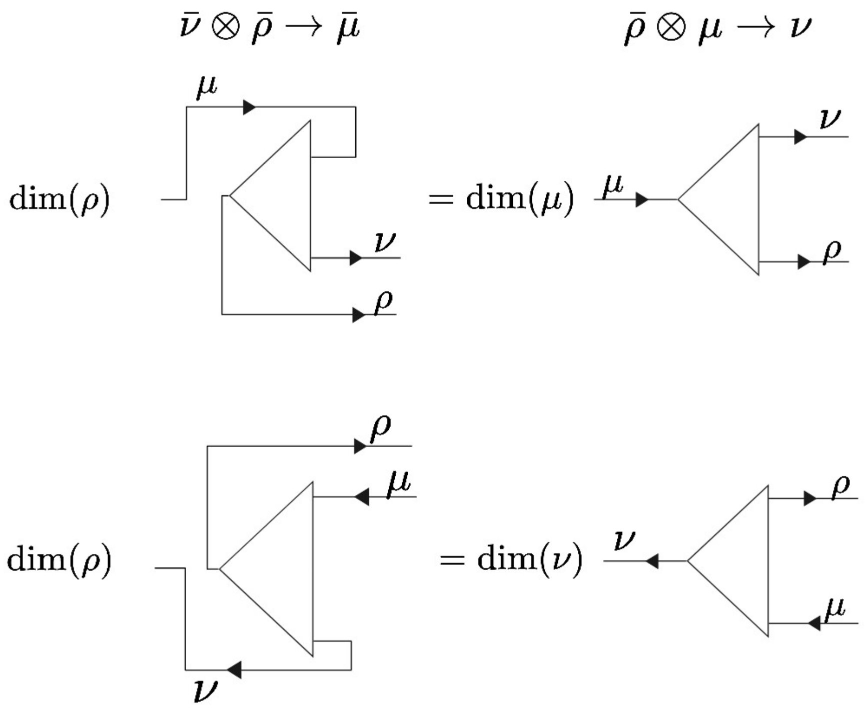


Figure 13: Orthonormality relation for Clebsches

3.3.2 3-vertices and 3-js

When μ, ν, ρ are irreducible representations, any Clebsch is related to two other Clebsches by the fact:



$\bar{\nu} \otimes \bar{\rho} \to \bar{\mu}$
$\bar{\rho} \otimes \mu \to \nu$
$\dim(\rho)$
μ
ν
ρ
$= \dim(\mu)$
$\dim(\rho)$
$= \dim(\nu)$
miro

Figure 14: Caption

A **3-vertex** is defined as a rescaling of a Clebsch so that the projector obtained from contracting a 3-vertex with its conjugate, as in fig.(10), is **idempotent**.

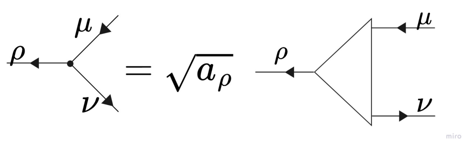
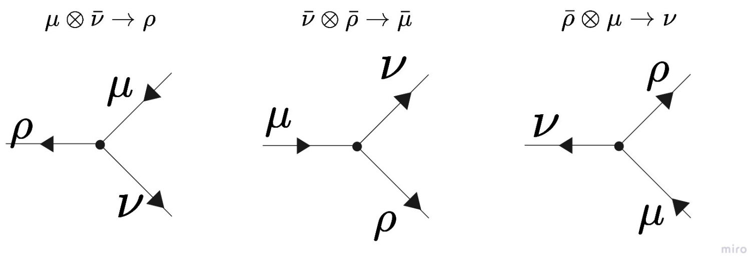


Figure 15: A 3-vertex



$\mu \otimes \bar{\nu} \to \rho$
$\bar{\nu} \otimes \bar{\rho} \to \bar{\mu}$
$\bar{\rho} \otimes \mu \to \nu$
ρ
μ
ν
μ
ν
ρ
ν
ρ
μ
miro

Figure 16: Symmetry of the 3-vertex

Remark 3.2 (Symmetry of the 3-vertex)**.**
The notation is chosen to emphasis the 3-vertex's symmetry. However, one must keep in mind that one still has to keep track of the relative ordering of the representations.

In this notation, the completeness and orthogonality relations are expressed as:

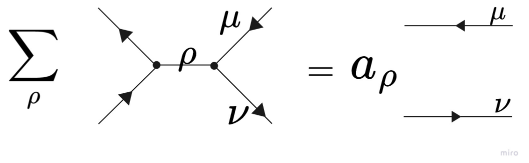


Figure 17: 3-vertex: Completeness relation

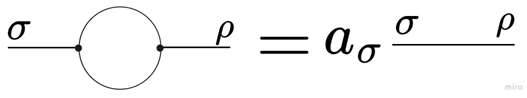


Figure 18: 3-vertex: Orthogonality relation

Given a 3-vertex, we can recover the normalisation factor by contracting both sides of fig.(18) with a delta function:

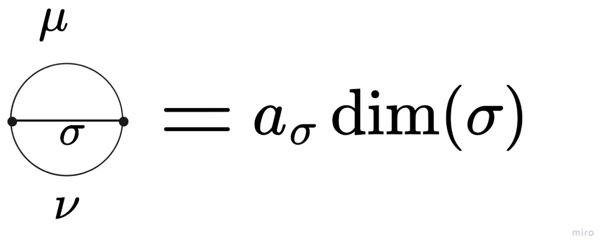


Figure 19: Recovering normalisation

Definition 3.1 (3j).
*The object on the LHS is what we will call a **3j**.*

Remark 3.3 (Symmetry of the 3j).
The reader may notice that, given the symmetry of the 3-vertex:

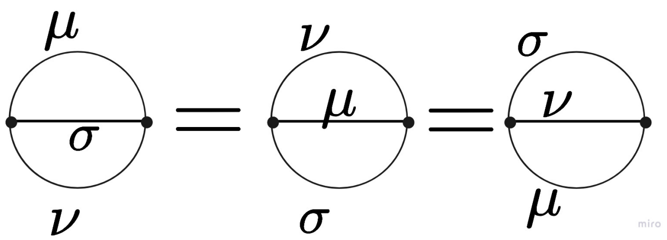


Figure 20: Symmetry of the 3j

This expresses the fact

$$a_\sigma \dim(\sigma) = a_\mu \dim(\mu) = a_\nu \dim(\nu)$$

as one would expect.

3.3.3 The recoupling relation and 6-js

The final piece of the puzzle to understanding the motivation of this paper is the recoupling relation. The equality is obtained by applying fig.(11) to both sides of the diagram on the left. See ref.[3] for the calculation in full.

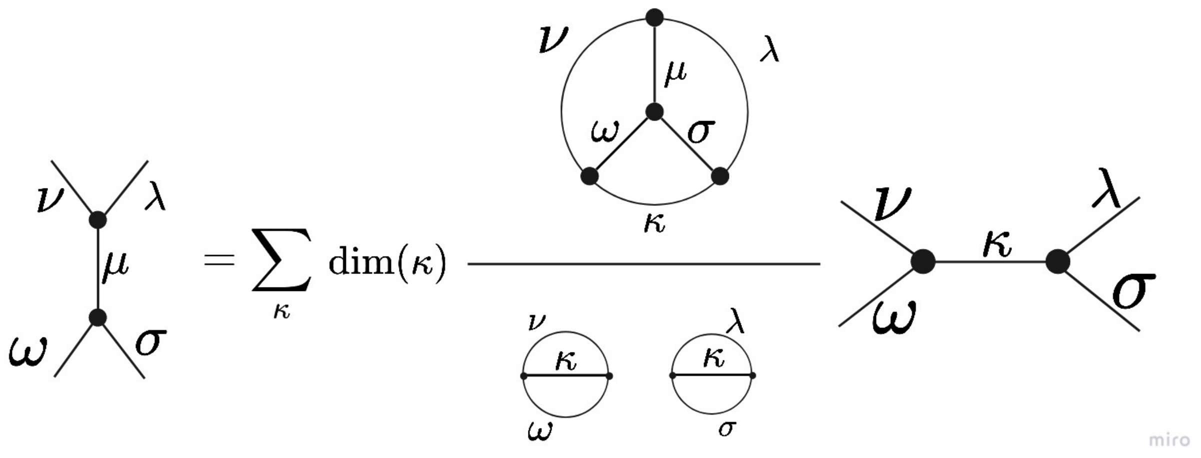


Figure 21: Recoupling relation

Definition 3.2 (6j).
The numerator of the fraction in fig.(21) is what we will call a ***6j****. It is a contraction of four 3-vertices.*

4 Motivation

4.1 Comprehensiveness of 3js and 6js

We're now in a position where we can understand the motivation for calculating 3js and 6js: we will use orthogonality (fig.18) and recoupling (fig.21) to express any Wigner 3n-j symbol as a product of 3js and 6js.

How to reduce a 3n-j:

1) Pick a loop
2) Reduce the number of vertices of that loop with recoupling until there are but two vertices left.
3) When there are but two vertices left on the loop, we can 'pop' the bubble with (fig.18).

The above reduces a 3n-j to a sum of 3(n-1)-js weighted by 3js and 6js. I will illustrate this on the simpler example of a 6j, however our interest lies in reducing more complicated expressions (9js and higher)

Example 4.1. *In a 6j, the smallest loop has 3 vertices. We apply the recoupling relation to the highlighted vertices to obtain:*

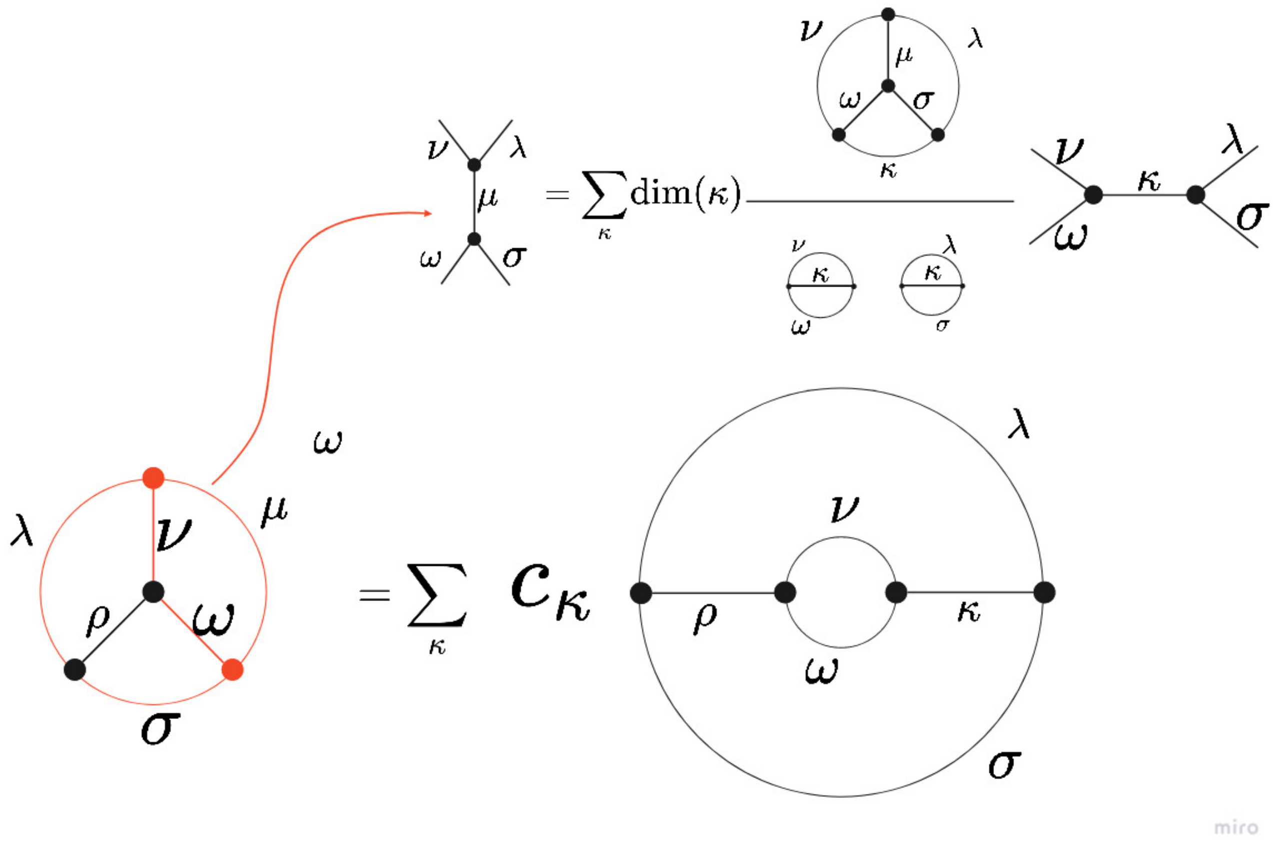


Figure 22: Steps 1 and 2 in reducing a 3n-j

We may now 'pop' the central bubble with (fig.18):

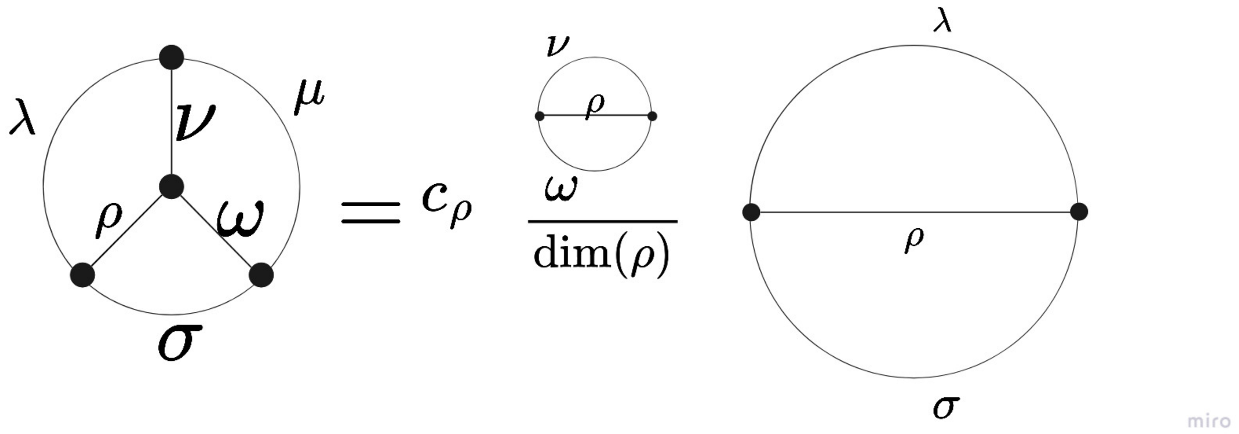


Figure 23: 'Reduction' of a 6j

The reader may notice that the final equality is trivial (a good way to check our equalities hold !). However the procedure has done what it is supposed to do: on the RHS of the equality we have a sum (just one term in this case) of products of 3js and 6js on the left, and a 3(n-1)-j (for us a 3j) on the right.

4.2 Example application: Wigner-Eckart theorem

Theorem 4.1 (Wigner-Eckart)**.**
Any invariant tensor can be reduced to a sum over the projection operators onto the irreducible subspaces weighted by Wigner 3n-j symbols (given by 'reduced matrix elements').

For the explicit form and a proof in birdtrack notation, see ref.[3].

The above theorem is as pleasing as it is useful, it allows for the complete reduction of any invariant tensor into the invariant subspaces. We may then understand the action of an invariant tensor as the direct sum of its action onto each irreducible subspace.

5 Calculation of 3js and 6js: background

5.1 Symmetrisers and Antisymmetrisers

In birdtrack notation, a permutation may be written:

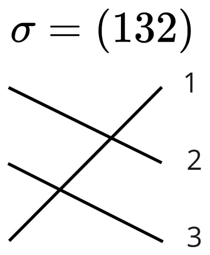


Figure 24: A permutation in birdtrack notation

In the above example, index 1 is now connected to what used to be connected to index 3.

Definition 5.1 (Symmetriser)**.**
A ***symmetriser****, represented by a white box, is the normalised sum of all possible permutations, i.e. the sum of all elements of S_n, with n the number of objects we're permuting:*

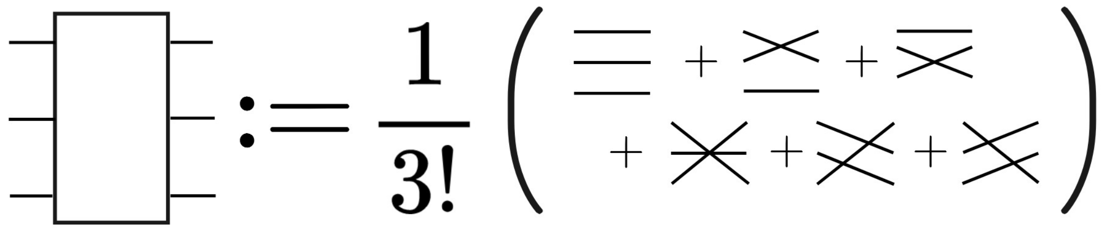


Figure 25: A symmetriser of length 3

where we have divided the sum by the number of terms, $n!$.

We may swap nodes on a symmetriser as we wish:

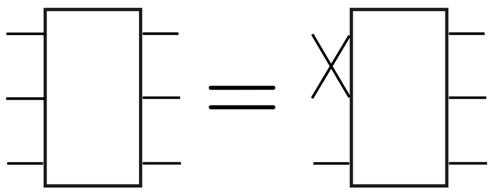


Figure 26: Swapping nodes on a symmetriser

this is an expression of the group property:

$$g' \sum_{g \in G} g = \sum_{g \in G} g \tag{5}$$

Definition 5.2 (Antisymmetriser).
An ***antisymmetriser****, represented by a black box, is the normalised* ***signed*** *sum of all permutations:*

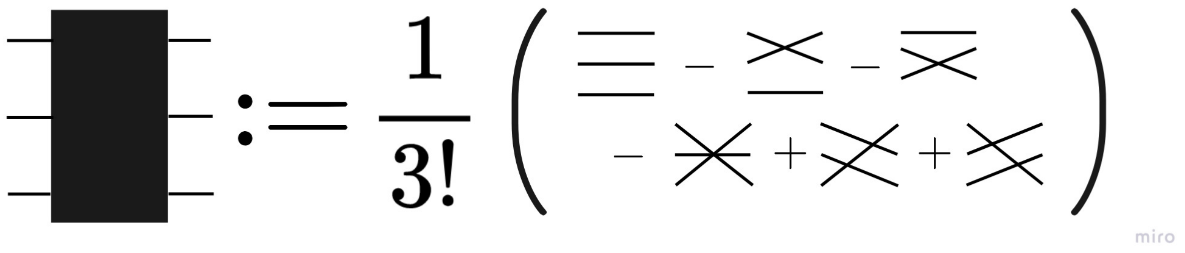


Figure 27: An antisymmetriser of length 3

Swapping two nodes on an antisymmetriser induces a minus sign:

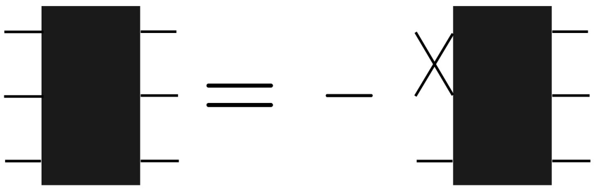


Figure 28: Swapping nodes on an antisymmetriser

this expresses the fact:

$$g' \sum_{g \in S_n} \mathrm{sgn}(g) * g = \mathrm{sgn}(g') \sum_{g \in G} \mathrm{sgn}(g) * g$$

From the properties of figs.(26, 28), it is immediate that connecting any two nodes of a symmetriser to any two nodes of an antisymmetriser results in zero:

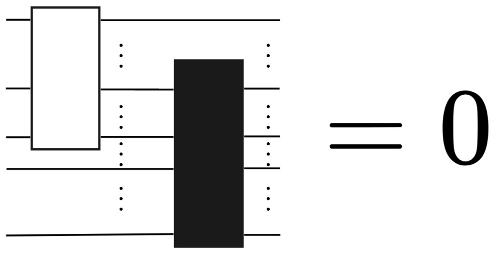


Figure 29: Contracting symmetrisers with antisymmetrisers yields zero

5.2 Representation theory reminders

Definition 5.3 (Representation)**.**
A representation ρ of a group G is a homomorphism from its objects g to automorphisms $\rho(g)$ of a vector space, the representation space.

A representation makes a member g of an abstract group G concrete by mapping it to a matrix $\rho(g)$ on a vector space which obeys the same multiplication rules:

$$\rho(g \circ g') = \rho(g) \circ \rho(g')$$

We remind ourselves of some facts:

- Every representation is a sum of irreducible representations (often called 'irreps'):

  $$\rho(G) = \rho_1(G) \oplus ... \oplus \rho_2(G)$$

  where each $\rho_i(G)$ is irreducible
- A finite group has as many irreducible representations as it has conjugacy classes.

5.3 The regular representation

Given a group G, we can construct a vector space $V_G(K)$ by considering elements of the form

$$\sum_{g \in G} c_g g$$

where the c_g belong to some field K. The group elements act as basis vectors.

Definition 5.4 (The left regular representation)**.**
Define a group G to act upon the basis vectors of $V_G(K)$ by left group multiplication. This furnishes a representation called the left regular representation.

The regular representation contains each irreducible representation dim(irrep) times:

$$\rho_{\text{reg}}(G) = \dim(\rho_1) \cdot \rho_1(G) \oplus ... \oplus \dim(\rho_n) \cdot \rho_n(G)$$

An element other than 0 or e of V_G that is idempotent, i.e.

$$P = \sum_{g \in G} p_g g, \qquad P^2 = P$$

generates a non-trivial subrepresentation of the regular representation. If we can find a complete orthogonal set of such elements, i.e. $\sum_i \dim(\rho_i)$ such elements, then we have fully reduced the regular representation. In the following sections, we will hunt for such idempotents to fully reduce the regular representation of S_n.

Remark 5.1 (Projection operators and idempotents)**.**
P is such that any element of V_G acting on P belongs to the non-trivial subspace V_P generated by P:

$$gP = Pg \in V_P, \qquad \forall g \in G$$

As such, P ***projects*** *onto V_P.*

Remark 5.2 (Group Algebra)**.**
By multiplying together elements of V_G, we have extended V_G to a ***group algebra****. Its definition is fairly natural, so I decided to avoid unnecessary detail on the group algebra. Should the reader wish to know more, I have placed a discussion in the appendix (sec.A). There is also an interesting discussion in ref.[4].*

5.4 S(n), Young diagrams, Young tableaux

We remind ourselves of the following fact:

Theorem 5.1 (Conjugacy classes of S_n)**.**
The conjugacy classes of S_n are given by the cycle types. These are in turn given by the partitions of n.

Definition 5.5 (Young diagram)**.** *A Young diagram labels a partition of n as follows:*

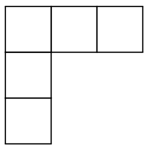
1) Take a partition of n and order it in decreasing order:

$$n = n_1 + ... + n_m, \qquad n_1 \geq ... \geq n_m$$

2) The Young diagram is given by a left-aligned diagram with n_1 boxes in the first row, n_2 in the second and so on...

For our purposes a Young diagram labels a conjugacy class of S_n, and we will see how it also labels an irrep of S_n.

Example 5.1. *Take the partition $5 = 3 + 1 + 1$. This represents elements of S_n of cycle type (...)(.)(.) = (...). Its Young diagram is given by*



Definition 5.6 (Young tableau)**.**
*A **Young tableau** is an n-box Young diagram where the numbers $1, ..., n$ have been placed into the boxes.*

Definition 5.7 (Standard Young tableau)**.**
*A **standard Young tableau** is a Young tableau such that each number is both greater than the number on its right and the number below it (if there are such numbers).*

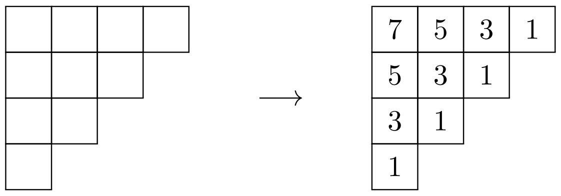
Example 5.2. *Among the 3 Young tableaux below, the first is standard while the next two are not:*

$$\begin{array}{|c|c|c|}\hline 1 & 3 & 4 \\ \hline 2 & \multicolumn{2}{c}{} \\ \cline{1-1} 5 & \multicolumn{2}{c}{} \\ \cline{1-1}\end{array} \qquad \begin{array}{|c|c|c|}\hline 1 & 5 & 3 \\ \hline 2 & \multicolumn{2}{c}{} \\ \cline{1-1} 4 & \multicolumn{2}{c}{} \\ \cline{1-1}\end{array} \qquad \begin{array}{|c|c|c|}\hline 1 & 3 & 5 \\ \hline 4 & \multicolumn{2}{c}{} \\ \cline{1-1} 2 & \multicolumn{2}{c}{} \\ \cline{1-1}\end{array}$$

Definition 5.8 (Hook number)**.**
*The **Hook number** of a **Young diagram** is the number of standard tableaux with the same shape as the Young diagram.*

Theorem 5.2 (Hook length formula)**.**
The Hook number of an arbitrary Young diagram with n boxes can be computed as follows:

- *For each box in the diagram, count the box itself, the number of boxes below in the same column, and the number of boxes to the right in the same column. Insert this number into the box:*



- *Multiply the numbers in each box together, and divide $n!$ by this number. For our example, the Hook number is:*

$$\frac{10!}{7 * 5^2 * 3^3} = 768$$

Although simple in appearance, the Hook formula is notoriously difficult to prove intuitively. A relatively short proof is given by ref.[2].

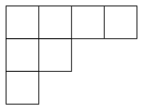
5.5 Young projection operators: uniqueness

Let a Young diagram represent an operator as follows:

For each row r_i, place a symmetriser S_i on the left of size the length of r_i.

For each column c_i, place an antisymmetriser A_i on the right of size the length of c_i.

The below Young diagram is represented on the right:



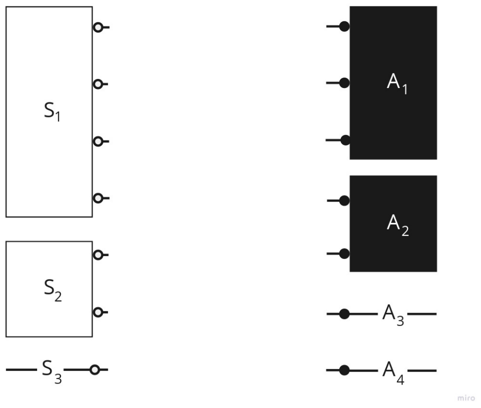


Figure 30: Operator constructed from the Young diagram

We will now go about connecting the symmetrisers to the antisymmetrisers:

Proposition 5.1 (Uniqueness)**.**
For a non-zero result, there is only one possible set of antisymmetrisers each symmetriser may connect to.

The proof is algorithmic:

- Consider S_1. It has size the number of columns and so the number of antisymmetrisers. For a non-zero result, each node of the symmetriser must connect to a different antisymmetriser. We choose the top available node throughout.
- Now consider S_2. It has size the number of antisymmetrisers with remaining nodes and so again there is only one possible choice of antisymmetrisers to connect to.
- We can continue in this way until all connections are made.

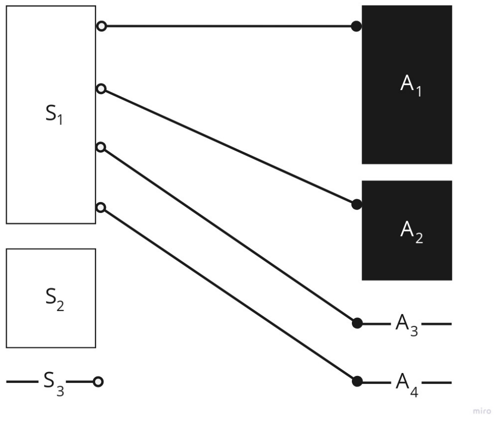


Figure 31: S_1 has been connected the only possible way. Now there is only one possibility for S_2.

The algorithm above describes how to construct what we will call the **standard connection**.

Remark 5.3.
Although the choice of antisymmetrisers to which each symmetriser connects to is unique, there is freedom in choosing which node of a symmetriser connects to which antisymmetriser. We may also choose which node within an antisymmetriser we are connecting to. However, since we may swap nodes of a symmetriser, and swap those of an antisymmetriser at the cost of a minus sign, the result is always the same up to a possible sign difference.

Prop.(5.1) is central to our discussion: it forms the basis to many of the proofs that follow.

5.6 Young projection operators: orthogonality

Now that we have established the standard connection for a Young diagram, we now define a **Young projection operator** for each standard Young tableau as follows: the number in the box of row i, column j passes through S_i and A_j:

Consider the tableau:

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

We may construct the following table to guide us:

	A_1	A_2	A_3	A_4
S_1	1	2	3	4
S_2	5	6		
S_3	7			

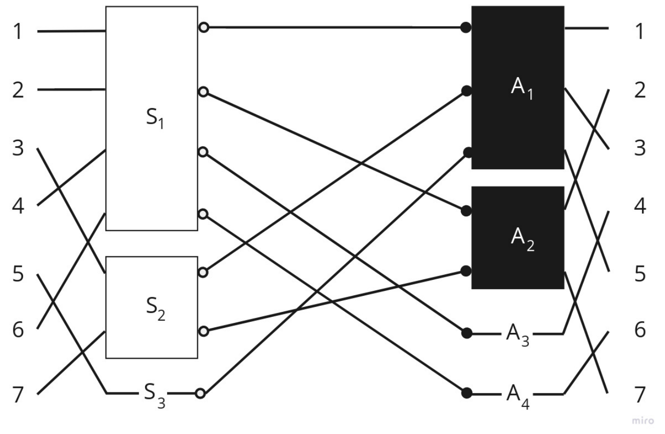


Figure 32: Young projection operator $Y(T)$ for the standard tableau T defined on the left

Using the uniqueness property, we will prove the standard tableaux form an orthogonal set.

Proposition 5.2. *The Young projection operators are orthogonal.*

Proof. Starting with the projector of fig.(32), consider attaching a projector corresponding to a standard Young tableau with the same shape.

$$T' = \begin{array}{|c|c|c|c|}\hline & & & \\ \hline & \\ \cline{1-2} \\ \cline{1-1}\end{array}$$

The algorithm is the same as in the proof of prop(5.1), simply with symmetrisers and antisymmetrisers reversed. Each antisymmetriser must connect to a different symmetriser, and so as before we place node

1 of A_1 to node 1 of S'_1, node 2 to node 1 of S'_2 and so on:

$$T' = \begin{array}{|c|c|c|c|}\hline 1 & & & \\ \hline 3 & & \multicolumn{2}{c}{} \\ \cline{1-2} 5 & \multicolumn{3}{c}{} \\ \cline{1-1}\end{array}$$

Seeing the algorithm through to the end, the only $Y(T')$ we can attach to $Y(T)$ is $Y(T)$ itself, and we get the result:

$$\boxed{Y(T) \circ Y(T') = \delta_{T,T'} Y(T)^2} \tag{6}$$

□

Remark 5.4 (Normalisation).
It can be shown that $Y(T_\lambda)^2 = \dfrac{1}{k_\lambda} Y(T_\lambda)$*, where* k_λ*, which depends only on the shape of the tableau, is given by:*

$$k_\lambda = \frac{\left(\prod_{i=1}^{s} |S_i|!\right)\left(\prod_{j=1}^{t} |A_j|!\right)}{|\lambda|}$$

where $|Y|$ *is the Hook number, and* $|S_i|$*,* $|A_j|$ *is the length of the symmetriser* i *(row* i*), the antisymmetriser* j *(column* j*).*

I omit the proof, however one may be found in ref.[3].

We denote our **normalised Young projection operator** by:

$$\boxed{P_{Y(T)} = k_T Y(T)} \tag{7}$$

Remark 5.5. *The numerator of* k_T *is the total number of terms in* $Y(T)$*'s expansion:*

$$k_{T_\lambda} = \frac{|Perms(Y(T))|}{|\lambda|} \tag{8}$$

5.7 The Garnir algorithm: standard tableaux as the basis vectors of the irreducible representations of S(n)

Before giving the explicit form of $S(n)$'s representation, we begin with a warning:

Warning 5.1 (Double usage of tableaux)**.**
Previously we have used ***standard tableaux*** *to denote* ***different Young projection operators****. In this section and the next, we will stick to a* ***single*** *(indeterminate)* ***Young projection operator*** $Y(T)$*. We will allow elements of* S_n *to act upon this tableau. To describe the resulting representation, we use Young tableaux to denote the resulting objects* $\sigma \circ Y(T)$*,* $\sigma \in S_n$*.*

We will consider the representation furnished by a standard tableau T. We define $\sigma \in S_n$ to act on $Y(T)$ on the left:

$$\sigma * Y(T) = \sigma \circ Y(T)$$

In the following, we will represent $\sigma * Y(T) = \sigma \circ Y(T)$ by T', where T' is the result of acting upon the elements of T by the permutation σ.

Example 5.3. *In the representation of* S_3 *furnished by* $\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 \\ \cline{1-1}\end{array}$*,*

$$\begin{array}{|c|c|}\hline 3 & 1 \\ \hline 2 \\ \cline{1-1}\end{array} := (132) \circ Y\left(\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 \\ \cline{1-1}\end{array}\right)$$

Remark 5.6. *Under this formalism, the order in which numbers appear in a row does not matter, as any row permutation can be absorbed into the corresponding symmetriser by eq.(5).*

Before stating one of the crowning results of the representation theory of S_n, we give one final definition:

Definition 5.9 (Standard permutation)**.**
Given a Young tableau T*, we define a* ***standard permutation*** *to be a permutation such that its action on the tableau's entries yields a standard tableau.*

And now we're ready:

Proposition 5.3. *In this formalism, every tableau can be expressed as a sum of standard tableaux. In other words:*

$$\sigma \circ Y(T) = \sum_{i=1}^{H(T)} c_i \sigma^i_{standard} \circ Y(T), \qquad c_i \in \mathbb{Z} \tag{9}$$

for any $\sigma \in S_n$*, where* $\sigma_{standard}$ *is a* ***standard permutation*** *and* $H(T)$ *is the Hook number (def.5.8) of the (Young diagram of) the tableau* T*.*

From this we conclude the dimension of $\rho_T(S_n)$ *is the Hook number of (the Young diagram of)* T*.*

In the next section, we prove the above by showing how we can express non-standard tableaux in terms of standard tableaux with the Garnir algorithm.

5.8 The Garnir algorithm

5.8.1 Algorithm

Consider a tableau that is not standard. We may reorder rows:

$$\begin{array}{|c|c|c|c|}\hline 2&1&4&3\\ \hline 8&5&10\\ \cline{1-3} 6&7&9\\ \cline{1-3}\end{array} \quad\longrightarrow\quad \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&8&10\\ \cline{1-3} 6&7&9\\ \cline{1-3}\end{array}$$

In our example, the tableau is still not standard. I will now illustrate the Garnir algorithm:

- Moving from left to right, top to bottom, highlight the first number larger than the one below, as well as the number below it:

$$\begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&{\color{red}8}&10\\ \cline{1-3} 6&{\color{red}7}&9\\ \cline{1-3}\end{array}$$

- Also highlight numbers to the right of the top number, and those to the left of the bottom number

$$\begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&{\color{red}8}&{\color{red}10}\\ \cline{1-3} {\color{red}6}&{\color{red}7}&9\\ \cline{1-3}\end{array}$$

- The sum of all partitions of the highlighted numbers into the two rows of highlighted spaces comes to zero (a birdtrack proof is given in sec.(6.1), for a proof in conventional notation, see for example ref.[8], although beware of differing conventions):

$$\begin{aligned} 0 \;=\; & \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&{\color{red}8}&{\color{red}10}\\ \cline{1-3} {\color{red}6}&{\color{red}7}&9\\ \cline{1-3}\end{array} + \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&{\color{red}6}&{\color{red}7}\\ \cline{1-3} {\color{red}8}&{\color{red}10}&9\\ \cline{1-3}\end{array} + \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&{\color{red}6}&{\color{red}8}\\ \cline{1-3} {\color{red}7}&{\color{red}10}&9\\ \cline{1-3}\end{array} \\ & + \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&{\color{red}6}&{\color{red}10}\\ \cline{1-3} {\color{red}7}&{\color{red}8}&9\\ \cline{1-3}\end{array} + \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&{\color{red}7}&{\color{red}8}\\ \cline{1-3} {\color{red}6}&{\color{red}10}&9\\ \cline{1-3}\end{array} + \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&{\color{red}7}&{\color{red}10}\\ \cline{1-3} {\color{red}6}&{\color{red}8}&9\\ \cline{1-3}\end{array} \end{aligned}$$

  (what order we put the numbers in within a row doesn't matter as we can always reorder rows).

- We may therefore write:

$$\begin{aligned} \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&8&10\\ \cline{1-3} 6&7&9\\ \cline{1-3}\end{array} \;=\; & - \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&6&7\\ \cline{1-3} 8&9&10\\ \cline{1-3}\end{array} - \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&6&8\\ \cline{1-3} 7&9&10\\ \cline{1-3}\end{array} - \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&6&{\color{red}10}\\ \cline{1-3} 7&8&{\color{red}9}\\ \cline{1-3}\end{array} \\ & - \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&7&8\\ \cline{1-3} 6&9&10\\ \cline{1-3}\end{array} - \begin{array}{|c|c|c|c|}\hline 1&2&3&4\\ \hline 5&7&{\color{red}10}\\ \cline{1-3} 6&8&{\color{red}9}\\ \cline{1-3}\end{array} \end{aligned} \qquad (10)$$

  where I have reordered rows into ascending order.

Notice we still have 2 non-standard tableaux. However, the offending number is further along. By recursively applying the Garnir algorithm, one always arrives at a sum of standard tableaux. I have placed a proof of this in the appendix (sec.B). There is also an example of the algorithm at work in sec.(C).

5.9 Summary: The irreducible representations of S(n)

Let us summarise the properties of $Y(T)$ as constructed in sec.(5.6):

- They are, up to a scalar, idempotent, and thus can be considered to be projection operators for the regular representation.
- Given n, all the $Y(T)$ (where T is an n-box standard tableau) are mutually orthogonal. This was essentially given by the uniqueness property (prop.5.1).
- By the Garnir algorithm, the dimension of the representation furnished by $Y(T)$ is the number of standard tableaux with the same shape. We can thus label the basis vectors of a representation space by the standard Young tableaux.

We can summarise this as follows:

Theorem 5.3. *The Young projection operators completely reduce the regular representation of S_n.*

Indeed for each conjugacy class c of S_n, i.e. each n-box Young diagram λ, we have $H(\lambda)$ equivalent irreducible representations (one for each standard tableau T_λ). The basis vectors of each of these representations can also be labelled by the respective standard tableaux and so they have dimension $H(\lambda_T)$. We have an orthogonal set with each irrep appearing dim(irrep) times, the complete reduction of $S(n)$.

Remark 5.7. *The Garnir algorithm actually only tells us that the dimension of the representation furnished by $Y(T_\lambda)$ is* ***at most*** *the number of standard tableaux with the same shape $H(\lambda)$. However, since for each λ we found $H(\lambda)$ orthogonal irreps within the regular representation, we can conclude that* $\dim(Y(T_\lambda)) \geq H(\lambda)$ *and so indeed* $\dim(Y(T_\lambda)) = H(\lambda)$.

Remark 5.8 (Completeness)**.** *Since the Young projection operators form a complete orthogonal set, they must satisfy the completeness relation:*

$$\sum Y(T) = \mathbb{1} \tag{11}$$

where the sum is taken over all n-box standard tableaux.

5.10 U(n)

Now that we have a solid grasp of Young projection operators, we state a few facts about the representation theory of $U(n)$. We start with the remarkable result:

Theorem 5.4 (Schur-Weyl duality).
The Young projection operators may be used to project onto the irreps of $U(n)$

The irreps of U(n) are labelled by the k-box standard tableaux with no more than n rows. Instead of acting upon the numbers $1, ..., k$, in this context the Young projection operators act upon the k indices of a tensor.

We simply state the dimension formula as given in ref.[3]:

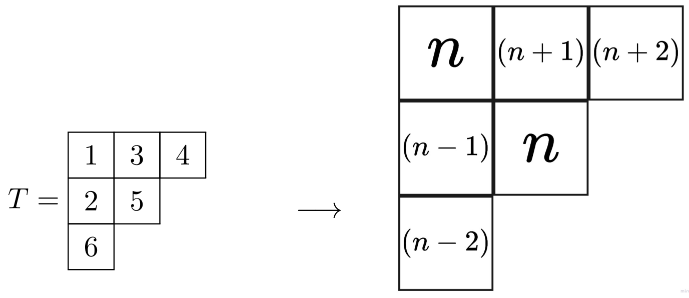
Theorem 5.5 (Dimension of a U(n) irrep).
For an irrep of $U(n)$ given by the Young tableau T_λ,

$$d_Y(T_\lambda) = \frac{f_\lambda(n)}{|\lambda|}$$

where $f(n)$ is a polynomial in n obtained from the Young diagram λ by multiplying the numbers written in the boxes of λ, according to the following rules:

- *The upper left box contains an n.*
- *The numbers in a row increase by one when reading from left to right.*
- *The numbers in a column decrease by one when reading from top to bottom.*

Example 5.4.



$$\longrightarrow \quad \dim(\rho_{Y(T)}) = \frac{n^2(n^2-1)(n^2-4)}{|\lambda_T|}$$

Figure 33: Dimension formula for $U(n)$

5.11 U(n) 3j and 6j coefficients

Thanks to Schur-Weyl duality, we are now in a position to display the 3js and 6js of $U(n)$ in birdtrack notation. For irreps X, Y, Z of $U(n)$ labelled by Young tableaux T_X, T_Y, T_Z, with the number of boxes of T_X and T_Z summing to the number of boxes of T_Y, a U(n) 3j is given by:

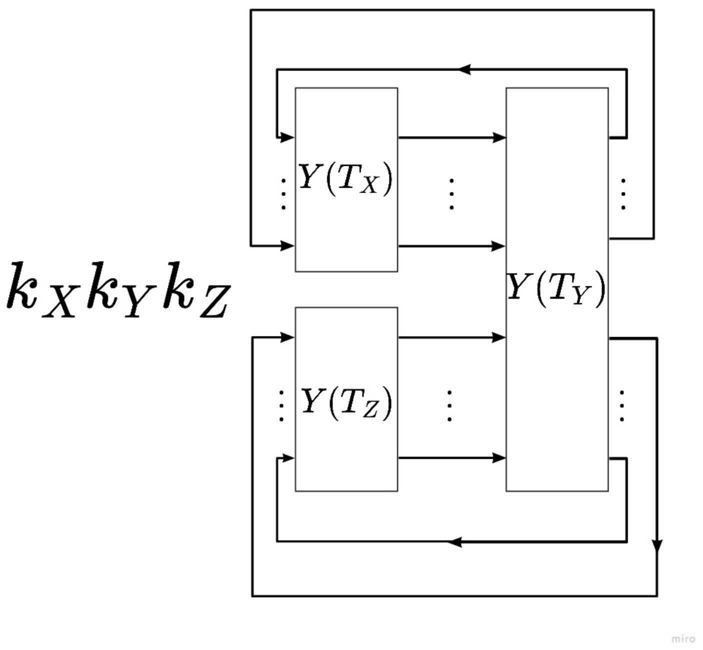


Figure 34: A $U(n)$ 3j in birdtrack notation

From now on, we'll simply denote $Y(T_X)$ by X in diagrams. For 6js, we get:

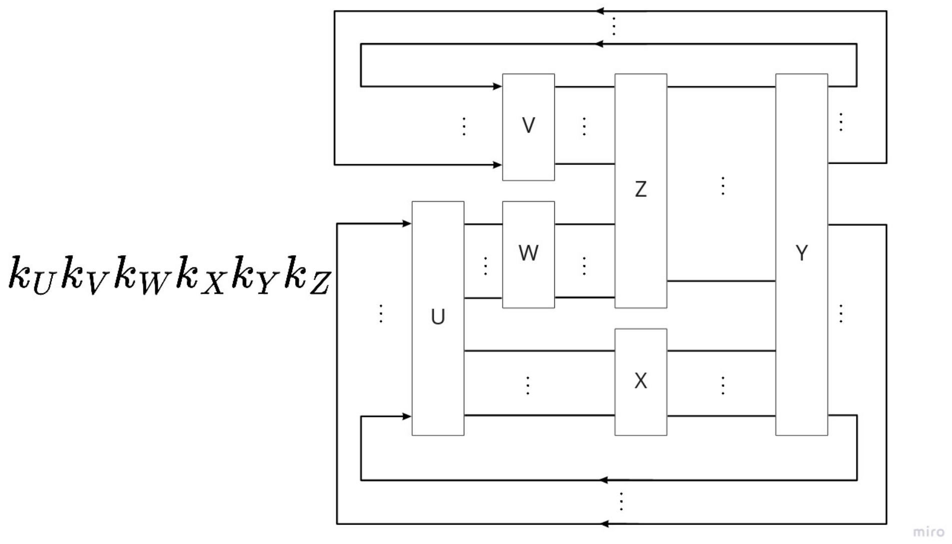


Figure 35: A $U(n)$ 6j in birdtrack notation

for irreps U, V, W, X, Y, Z whose dimensions fit as shown in the diagram.

5.12 Elvang's sandwich theorem ref.[9]

5.12.1 m(sigma)

We once again use the fact there is a unique possible connection between the symmetrisers and antisymmetrisers of a Young tableau. We may conclude that, for any permutation σ,

$$Y(T) \circ \sigma \circ Y(T) = m(\sigma) Y(T)^2 \tag{12}$$

where $m = -1, 0, 1$ depending on whether we can rearrange the permutation into the standard connection, and whether or not we get a minus sign as we do so.

Consider these two examples for the tableau

1	2
3	5
4	

:

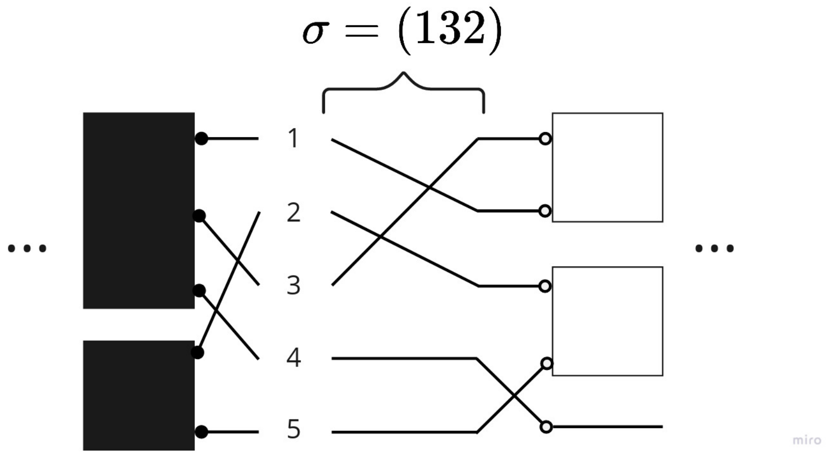


Figure 36: Example 1 in calculating $m(\sigma)$

By inspection, we notice lines 1 and 3 connect the same symmetriser to the same antisymmetriser and so $m(\sigma) = 0$

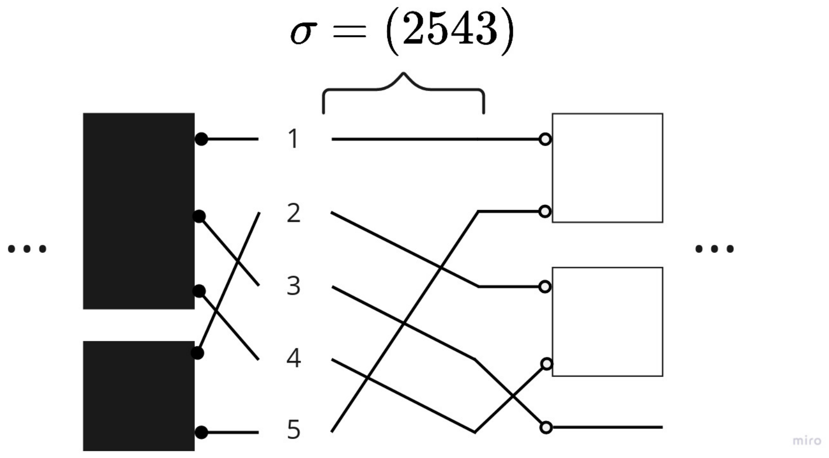


Figure 37: Example 2 in calculating $m(\sigma)$

This is already more complicated, you could work out that a swap on the 2nd symmetriser, a swap on the bottom two nodes of the top antisymmetriser, and a swap on the bottom antisymmetriser gives us back the standard form, and so $m(\sigma) = (1)^2(-1)^2 = 1$

In section 6.2, I will propose an algorithm for the reliable calculation of $m(\sigma)$ factors.

5.12.2 The sandwich trick

We are now ready to illustrate the sandwich trick:

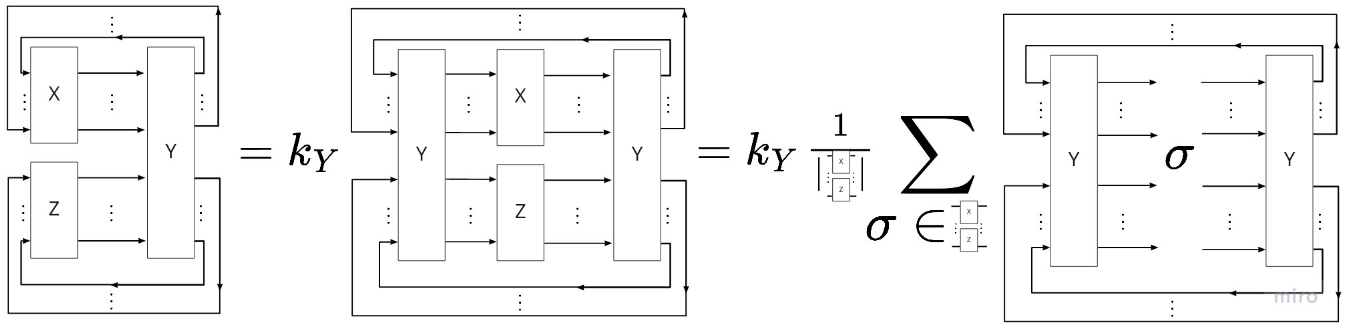


Figure 38: The sandwich trick

We can use the fact $Y^2 = k_Y Y$ to add in a second Y operator. The 2nd operator has been re-positioned along the arrows to make the "sandwich" more obvious. The middle operators can then be expressed as a sum of permutations (weighted by the total number of permutations, which is the fraction in the 3rd term), and by eq.(5.12.1), we get the result:

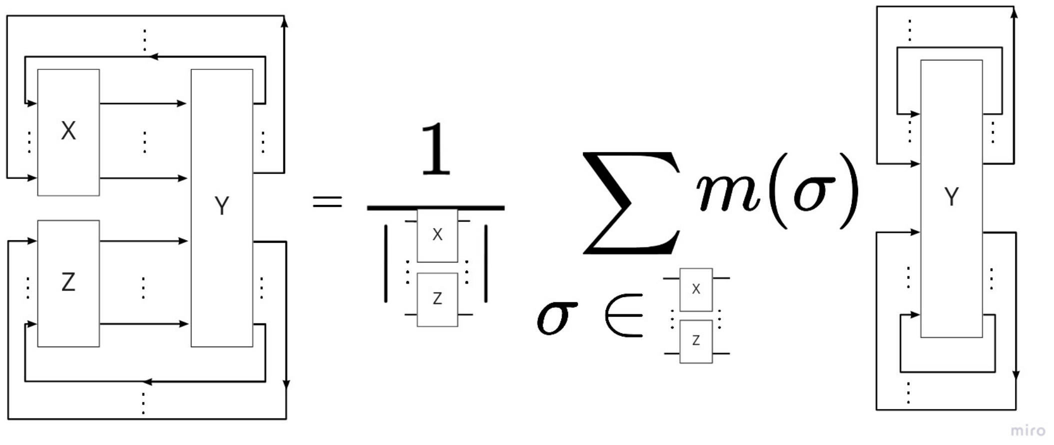


Figure 39: The sandwich trick part 2

where the object on the right is the trace of the operator. And so, remembering to re-introduce normalisation factors:

$$3j(X, Z; Y) = \mathrm{Tr}(Y) k_X k_Y k_Z \frac{1}{|\mathrm{Perms}(X, Z)|} \sum_{\sigma \in \mathrm{Perms}(X,Z)} m(\sigma) \tag{13}$$

Since $\mathrm{Tr}(Y) = \frac{1}{k_Y}\mathrm{Tr}(P_Y)$:

$$3j(X,Z;Y) = \mathrm{Tr}(Y)k_X k_Y k_Z \frac{1}{|\mathrm{Perms}(X,Z)|} \sum_{\sigma \in \mathrm{Perms}(X,Z)} m(\sigma) \tag{14}$$

and after further simplification using eq.(8):

$$\boxed{3j(X,Z;Y) = \dim(Y)\frac{1}{|X|}\frac{1}{|Z|} \sum_{\sigma \in \mathrm{Perms}(X,Z)} m(\sigma)} \tag{15}$$

where again $|X|, |Z|$ are Hook numbers.

Using the sandwich trick, we get for 6js:

$$\boxed{6j(U,V,W,X,Z;Y) = \dim(Y)\frac{1}{|U||V||W||X||Z|} \sum_{\sigma \in \mathrm{Perms}(U,V,W,X,Z)} m(\sigma)} \tag{16}$$

6 Efficient computation of 3j and 6j symbols

The computation of 3j and 6j symbols outlined in this paper heavily relies on the Garnir algorithm. Though there are many proofs in traditional notation (see for example ref.[8]), I will provide a proof in birdtrack notation as this makes it very easy to see why the sum (10) vanishes.

6.1 Garnir birdtrack proof

Consider attaching a symmetriser to the highlighted numbers. These numbers belong to two rows, so we're connecting a symmetriser to part or all of two symmetrisers. Following the previous example:

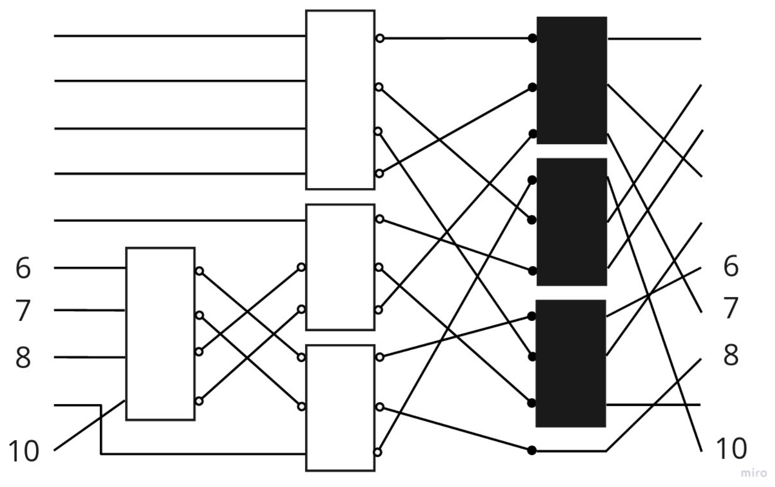


Figure 40: Attaching a symmetriser to highlighted numbers

No matter the permutation in the expansion of the symmetrisers in the middle, the $r + 1$ nodes (where r is the length of the row of the offending number) of the added symmetriser must connect to r antisymmetrisers:

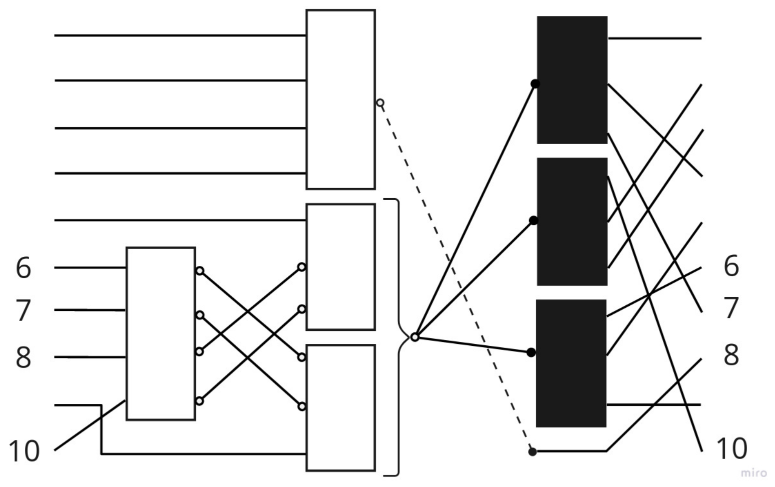


Figure 41: Not enough antisymmetrisers

and so we have shown that the sum we are dealing with comes to zero.

The sum of all permutations of the highlighted numbers in the tableau is zero. Since we may reorder rows as we please, we need only choose one representative of each partition of the highlighted numbers into the highlighted rows, as re-orderings of each partition appear the same number of times. We choose the highlighted numbers to be in ascending order, and this completes the proof.

Remark 6.1. *To be exact, each partition appears $u! * l!$ times, where u is the number of highlighted numbers in the upper row, and $l!$ is the number of highlighted numbers in the lower row.*

6.2 Algorithm for m(sigma)

As we saw in section (5.12.1), calculating $m(\sigma)$ can often be done by inspection. However, when many swaps need to be made this quickly gets messy. I propose the following algorithm:

- Starting with a Young tableau, construct a table showing which symmetriser each entry is connected to:

$$\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 5 \\ \hline 4 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \quad \longrightarrow$$

n	1	2	3	4	5
Symmetriser (row)	1	1	2	3	2

- Act upon the list of symmetrisers with σ^{-1}. Considering our two examples:

$$(123) \circ [1,1,2,3,2] = [1,2,1,3,2]$$

n	1	2	3	4	5
Symmetriser	1	2	1	3	2

$$(2345) \circ [1,1,2,3,2] = [1,2,3,2,1]$$

n	1	2	3	4	5
Symmetriser	1	2	3	2	1

- Enter this into our tableau:

$$\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 1 & 2 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \qquad\qquad \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 1 \\ \hline 2 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array}$$

$$m(\sigma_1) = 0 \qquad\qquad m(\sigma_2) = (-1)^2 = 1$$

The columns now show which symmetriser each node of the antisymmetriser is connected to. For the unique standard connection, columns are in ascending order.

- We can now calculate m:
  - If a column contains the same number twice, this means several nodes of an antisymmetriser are connected to the same symmetriser and so $m = 0$.
  - Otherwise m is the product of the signatures of each column (i.e. +1/-1 if the column is an even/odd permutation of numbers $1, ..., l$ where l is the length of the column).

7 Numerical implementation, U(n)

In this section we give the suggested formulas to be used for computer calculation of $U(n)$ 3js and 6js. Some formulas useful to checking such programs can be found in sec.(7.3).

7.1 3js

Let us remind ourselves of the formula:

$$3j(X,Z;Y) = \dim(Y)\frac{1}{|X|}\frac{1}{|Z|}\sum_{\sigma\in\text{Perms}(X,Z)} m(\sigma)$$

We will use the Garnir algorithm to further simplify the sandwich trick (fig.38).

By the Garnir algorithm, we can express the Young operator X (which we remember is a large sum of permutations) acting on the Young operator Y as $X \circ Y = \sum_i x^i \sigma^i_{\text{standard}} \circ Y, \quad x^i \in \mathbb{Z}$.
Similarly, $Z \circ Y = \sum_i z^i \sigma^i_{\text{standard}} \circ Y$.

Notice we may write:

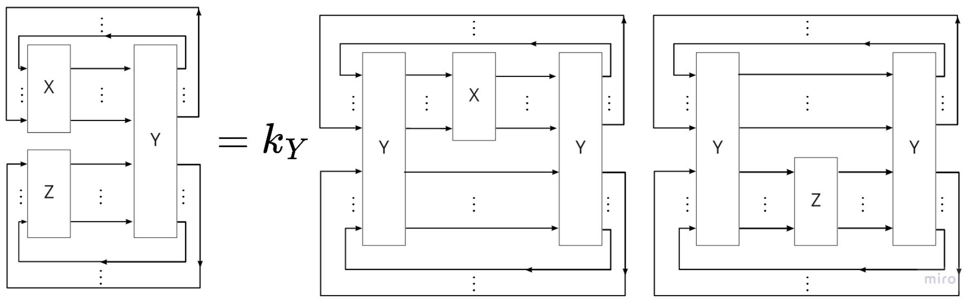


Figure 42: Factoring out X, Z

as X and Z commute (they are attached to different legs and are permuting different objects). We can now write:

$$\boxed{3j(X,Z;Y) = \dim(Y)\frac{1}{|X|}\frac{1}{|Z|}\left(\sum_i x^i m(\sigma^i_{\text{standard}})\right)\left(\sum_j z^j m(\sigma^j_{\text{standard}})\right)} \tag{17}$$

This greatly reduces the number of permutations we are dealing with. In particular, we only need to calculate $m(\sigma)$ for standard permutations.

7.2 6js

For 6js, we need to go a little further. As the sandwiched operators no longer all commute, we calculate their matrix representation in the representation given by the largest operator Y. The representation of $X = \sum_i x^i \sigma^i$, for instance, is given by

$$\rho_Y(X) = \sum_i x^i \rho_Y(\sigma^i)$$

where each $x^i = \pm 1$. In other words it is simply the sum of the matrix representations of all permutations in the expansion (weighted by their respective signs). This allows us to express $(U \circ V \circ W \circ Z \circ X) \circ Y$ (there is some freedom in the order here as some of these operators commute) as a sum of standard tableaux:

$$(U \circ V \circ W \circ Z \circ X \circ Y) = \sum_i c^i_{uvwzx} \sigma^i_{\text{stand}} \circ Y$$

Remember that Y is a basis vector of the representation, and so it is represented as a vector (with length the number of possible standard tableaux with the same shape as Y) with a single 1, with all other entries 0.

And thus:

$$\boxed{6j(U, V, W, X, Z; Y) = \dim(Y) \frac{1}{|U||V||W||X||Z|} \sum_i c^i_{uvwzx} m(\sigma^i_{\text{stand}})} \tag{18}$$

7.3 Checks: 3j and 6j sum rules

We include a couple formulas useful to checking computer programs which calculate 3js and 6js. These have been taken from ref.([3]), where further useful formulas can be found.

For 3js, given an operator Y,

$$\sum_{X,Z} 3j(X, Z; Y) = (N_b(Y) - 1) \dim(Y) \tag{19}$$

where $N_b(Y)$ is the number of boxes of Y, and the sum is taken over all operators X, Z that fit into the diagram (fig.34), i.e. for all $m = 1, ..., N_b(Y)$, all standard tableaux X with numbers $1, ..., m$ and all standard tableaux Z with numbers $m + 1, ..., N_B(Y)$.

The equivalent sum for 6js is:

$$\sum_{U,V,W,X,Z} 3j(X, Z; Y) = \frac{1}{2}(N_b(Y) - 1)(N_b(Y) - 2) \dim(Y) \tag{20}$$

where again the sum is over all possible Young operators that will fit into the diagram fig.(35) for given Y.

7.4 Speeding up calculations: early detection of zeros

Many 3js and 6js are zero. This may be deduced by inspection when calculating a single 3j or 6j at a time, however, for expressions involving many 3js and 6js, it is useful to implement checks such that a program can instantly recognise a 3j or 6j is zero.

A Young projector whose greatest antisymmetriser has length greater than the number of symmetrisers of the projector upon which it acts will yield zero, as several nodes will have to connect to the same symmetriser [1]. Similarly, a projector with fewer antisymmetrisers than the length of the greatest symmetriser of the projector upon which it acts will also be zero.

The above obviously does not cover all cases, however it is a simple check to implement as it usually only involves comparing the size of the first row and column of both operators.

For example, for the operator

$$Y = \begin{array}{|c|c|c|}\hline 1 & 2 & 5 \\ \hline 3 & 4 & 6 \\ \hline 7 \\ \cline{1-1} \end{array}$$

there 366 combinations of operators X, Z that fit (as in the 3j sum rule, eq.(19). 324 of these are zero, and the above check catches 166 of these. What's more, computation time in checking the sum rule was reduced by around two thirds (from 31s-33s to 10s-11s), I attribute this to the fact the check removes operators with particularly large rows or columns, which introduce a large $n!$ term in the total number of permutations.

[1]If there turn out to be as many symmetrisers as the length of the greatest antisymmetriser, we can then compare the 2nd greatest antisymmetriser to the number of symmetrisers remaining when each symmetriser has used up one node, and so on

7.5 Performance of my code

With my code, calculation time of a 3j rises exponentially with the number of boxes of the largest diagram:

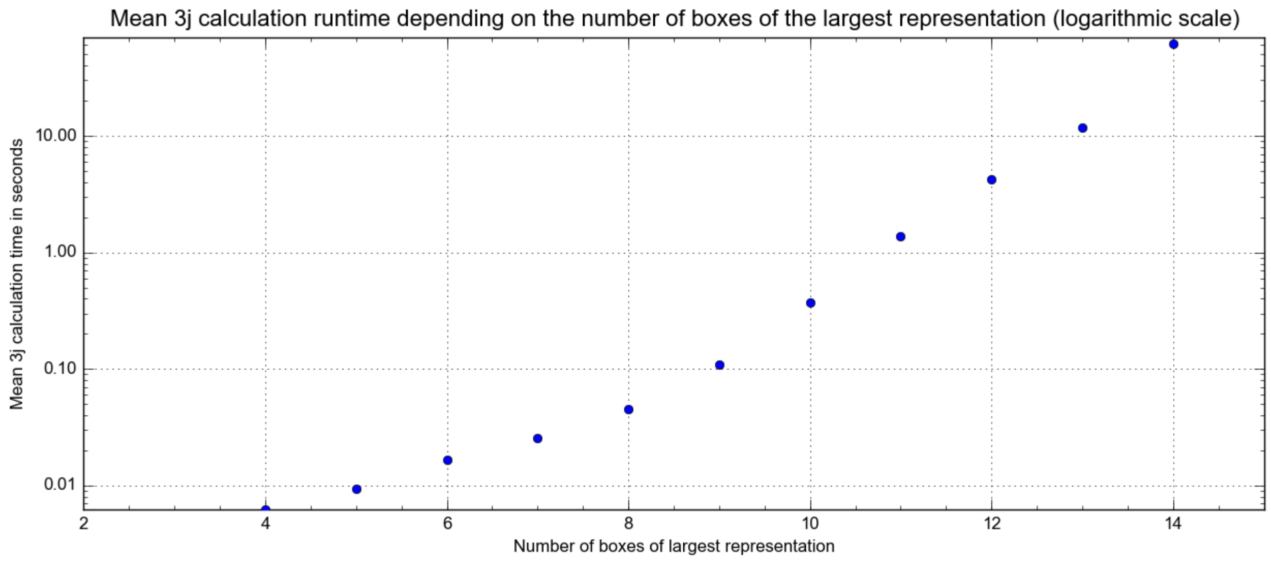


Figure 43: Performance of my code

Calculations of 6js use the same processes in my code and follow a similar exponential trend, although they are more costly. Run time exceeds a second from around 7 boxes onwards and quickly explodes.

This exponential growth is an improvement on the $n!$ factorial scaling one would expect from a combinatorial problem. To put this into perspective, a 9-box 3j calculation takes a tenth of a second, while generating S(9) takes approximately 13 seconds.

However improvements need to be made if we wish to calculate 3js and 6js with many boxes. A next step could be to implement an algorithm that efficiently generates S(n) from a generating set. This would mean we would only need to generate representation matrices for those elements in the generating set. The representations of all other elements would then be computed through appropriate matrix multiplication. This would drastically reduce the number of times the Garnir algorithm would need to be carried out. The Bruhat order of the symmetric group, on which there is an interesting discussion in ref.([5]), may be relevant to this task. A brief profiling of my code is given in sec.(D).

8 Conclusion

As mentioned, although use of the sandwich trick and the Garnir algorithm have allowed for reasonable computation times at low order (under a second per 3j for 10 boxes or less, 6 boxes or less for 6js), my code needs further optimising before it can calculate higher order terms. The most immediate step might be to create an algorithm to efficiently generate elements of $S(n)$. For this task, a study of the Bruhat order of the symmetric group may be of use.

There are many interesting avenues to explore which follow on from this paper. Expressing Wigner 3n-j coefficients in terms of 3js and 6js as in sec.(4.1) also presents an interesting graph manipulation problem. Creating a program that recursively reduces 3n-j graphs will be an interesting problem in computer science, as there will be several paths to the same result.

In this vein, the results of ref.([7]) on hermitian Young operators may warrant further study. Hermitian operators allow us to remove arrows from birdtracks and simplify graph reduction.

Finally, this paper discusses 3js and 6js for the unitary group $U(n)$. It is surely worth exploring whether similar tricks to the sandwich theorem exist for other groups of interest in physics such as the orthogonal group $O(n)$.

A The group algebra

Definition A.1 (Group algebra)**.**
The group algebra $A_G(K)$ of a group G over a field K is the vector space V_G over K, with group elements considered to be basis vectors, endowed with the additional structure of group multiplication.

The regular action of G on $V_G(K)$ induces a decomposition of V_G into irreducible subspaces (each appearing dim(irrep) times). This reduces the identity element into a sum of the identity elements of each subspace:

$$e = e_1^1 \oplus ... \oplus e_n^n$$

Definition A.2 (Ideal of an algebra)**.**
An ideal $\mathcal{L}$ of an algebra A is a subalgebra such that for all $a \in A, \quad i \in \mathcal{L}$,

$$a \circ i \in \mathcal{L}$$

Consider an element $i = \sum_{g \in G} c_g g \in V_G \backslash \{0, e\}, \quad$ such that i^2, defined by:

$$i^2 = \sum_{g \in G} c_g \left(\rho_{reg}(g) \circ \sum_{g' \in G} c_{g'} g' \right) = \sum_{g \in G} \sum_{g' \in G} c_g c_{g'} g \circ g' \tag{21}$$

satisfies $i^2 = i$. Then i is some sum of $1 \cdot e_i^j$ and thus generates an ideal of the group algebra, which is in this context the equivalent of an invariant subspace when considering the representation of the group.

In summary, an idempotent element of the group algebra generates a subrepresentation of the regular representation.

For a more complete treatment of the group algebra, see for example ref.([4]).

B Proof: the Garnir algorithm eventually yields standard tableaux

In this section we show that recursively applying this algorithm does indeed always eventually yield standard tableaux. To do this, we prove that, reading from top to bottom, left to right, each number up to and including the box of the previously offending number is greater than both its left-hand neighbour and its downstairs neighbour, and thus any remaining offending numbers are further along. In other words, applying the Garnir algorithm to a tableau whose nth box is the firs to break requirements yields a sum of tableaux whose first n boxes are guaranteed to satisfy requirements.

Proof. 1) Since each number before the offending number is smaller than its right hand neighbours, and smaller than its downstairs neighbour, it can be shown through a chain of inequalities that all highlighted numbers are greater than any number before the offending box.

2) Now consider the possible arrangements of highlighted numbers into the two rows. Since we've chosen rows to be ascending, it can again be shown that there is only a single possibility for the offending box to remain offending due to a chain of strict comparisons:

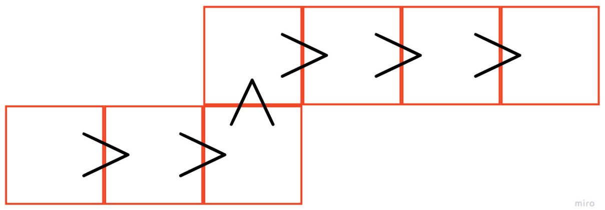


Figure 44: Strict chain of inequalities

which is the tableau we started with. We place this tableau on one side of the equation, as in eq.(10), the remaining tableaux, for which we have shown the offending box has been corrected, are placed on the other side of the equation. □

C Checks: The Garnir algorithm

The Garnir algorithm can be checked by verifying it does indeed result in a representation of S(n), i.e. by verifying:

$$\rho(\sigma_1 \cdot \sigma_2) = \rho(\sigma_1) \cdot \rho(\sigma_2)$$

It may also be worth getting a computer algorithm that conducts the Garnir decomposition to check itself

by brute force.

For example, in the rep. $\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array}$,

$$\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \quad = \quad (e + (12)(e - (13)) \quad = \quad e + (12) - (13) - (132)$$

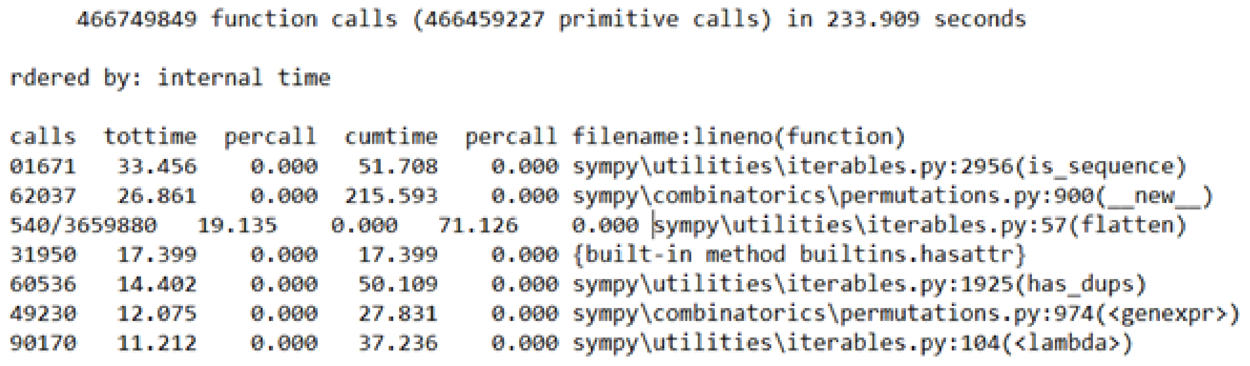
$$\begin{array}{|c|c|}\hline 2 & 3 \\ \hline 1 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \quad = \quad (123)\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \quad = \quad (123) + (13) - (23) - e$$

The Garnir algorithm should yield:

$$\begin{aligned}
\begin{array}{|c|c|}\hline 2 & 3 \\ \hline 1 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \quad &= \quad - \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \quad - \quad \begin{array}{|c|c|}\hline 1 & 3 \\ \hline 2 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \\
&= \quad - \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \quad - \quad (23)\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1}\end{array} \\
&= -(e + (12) - (13) - (132)) - ((23) + (132) - (123) - (12)) \\
&= -e + (13) - (23) + (123)
\end{aligned}$$

D Optimising my code

After finishing my code, I started off by profiling 3j calculation. After several small improvements such as minimising the use of deepcopy which seemed to take up an unusual amount of time, the top few functions looked like this:



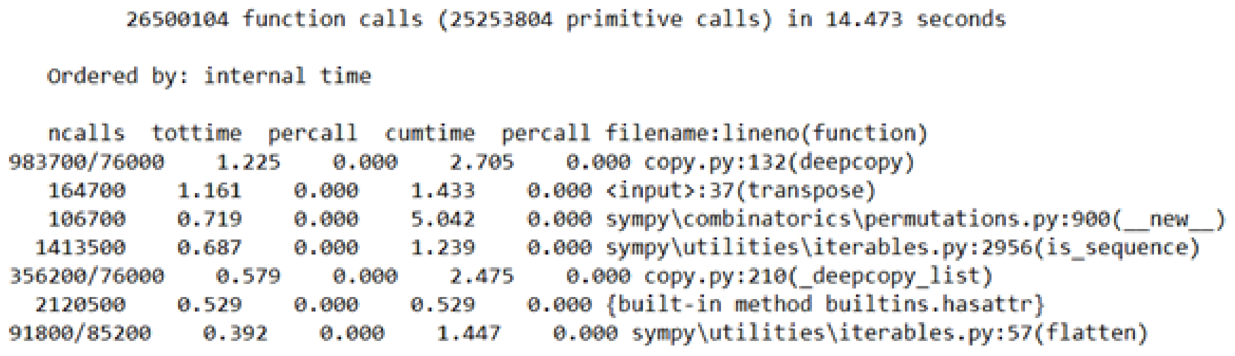
```
      466749849 function calls (466459227 primitive calls) in 233.909 seconds

rdered by: internal time

calls  tottime  percall  cumtime  percall filename:lineno(function)
01671   33.456    0.000   51.708    0.000 sympy\utilities\iterables.py:2956(is_sequence)
62037   26.861    0.000  215.593    0.000 sympy\combinatorics\permutations.py:900(__new__)
540/3659880   19.135    0.000   71.126    0.000 sympy\utilities\iterables.py:57(flatten)
31950   17.399    0.000   17.399    0.000 {built-in method builtins.hasattr}
60536   14.402    0.000   50.109    0.000 sympy\utilities\iterables.py:1925(has_dups)
49230   12.075    0.000   27.831    0.000 sympy\combinatorics\permutations.py:974(<genexpr>)
90170   11.212    0.000   37.236    0.000 sympy\utilities\iterables.py:104(<lambda>)
```

Figure 45: Initial profiling: Calculating a 9-box 3j 10 times

A lot of time seemed to be spent on generating permutations (not hugely surprising as the program does involve a lot of permutations). In a separate test, it seemed declaring the operators took as long as calculating the 3j, which did not seem right.

I realised generating all elements of S(n) was very computationally expensive, and obviously scales with $n!$. I saw big improvements after avoiding generating S(n) in its entirety where it was not needed. For example, when generating a representation, I used to generated a dictionary with the elements of S(n) as keys with None values to be filled in. I now add they keys only when I generate their values, which avoids generating placeholders that won't be used. I was also able to minimise its use during the Garnir algorithm by directly generating the required ordered partitions, instead of sifting through S(n).

After these changes, there was huge improvement (time reduced by a factor of over 100, my supervisor was right in saying this would be much more satisfying than writing better code the first time), and declaring the operators (not shown here) was now negligible next to the 3j calculation:



```
         26500104 function calls (25253804 primitive calls) in 14.473 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
983700/76000    1.225    0.000    2.705    0.000 copy.py:132(deepcopy)
   164700    1.161    0.000    1.433    0.000 <input>:37(transpose)
   106700    0.719    0.000    5.042    0.000 sympy\combinatorics\permutations.py:900(__new__)
  1413500    0.687    0.000    1.239    0.000 sympy\utilities\iterables.py:2956(is_sequence)
356200/76000    0.579    0.000    2.475    0.000 copy.py:210(_deepcopy_list)
  2120500    0.529    0.000    0.529    0.000 {built-in method builtins.hasattr}
91800/85200    0.392    0.000    1.447    0.000 sympy\utilities\iterables.py:57(flatten)
```

Figure 46: After minimising generation of S(n): Calculating a 9-box 3j 100 times

The top functions are now as I'd expect from the code I've written. I use deepcopy in several places, to generate standard tableaux for instance, and many functions work more conveniently on the transpose. Functions related to permutations are obviously not a surprise either. I have not found redundancy in the use of the listed functions in my code, although a skilled program would likely find more efficient methods than mine.

References

[1] Andrey Khusid. *Miro*. Mar. 21, 2022. URL: `https://www.miro.com`.

[2] Jason Bandlow. “An elementary proof of the hook formula”. In: *Electron. J. Comb.* 15.1 (Mar. 2008), R45.

[3] Predrag Cvitanović. *Group Theory : Birdtracks, Lie’s, and Exceptional Groups.* Princeton University Press, 2010. ISBN: 9781400837670.

[4] Morton Hamermesh. *Group theory and its application to physical problems.* Addison-Wesley, 1973.

[5] J.E. Humphreys. *Reflection Groups and Coxeter Groups.* Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1990. ISBN: 9780521436137. URL: `https://books.google.co.uk/books?id=ODfjmOeNLMUC`.

[6] Stefan Keppeler. “Birdtracks for SU(N)”. In: (2017). DOI: `10.21468/SciPostPhysLectNotes.3`. eprint: `arXiv:1707.07280`.

[7] Stefan Keppeler and Malin Sjödahl. “Hermitian Young operators”. In: *Journal of Mathematical Physics* 55.2 (2014), p. 021702. DOI: `10.1063/1.4865177`. URL: `https://doi.org/10.1063/1.4865177`.

[8] Redmond McNamara. “Irreducible Representations of the Symmetric Group”. In: University of Chicago. 2013.

[9] A.Kennedy P.Cvitanović H.Elvang. *Group Theory : Birdtracks, Lie’s, and Exceptional Groups.* Princeton University Press, 2010. Chap. 9: Unitary Groups. ISBN: 9781400837670.

Diagrams in this text were created with Miro([1])