SYMMETRIC FUNCTIONS IN GEOMETRY

ANTON MELLIT

Faculty of Mathematics, University of Vienna,
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

Abstract. Short transcript of the course on symmetric functions at the Univeristy of Vienna, SS 2018. Comments/Suggestions/Corrections are welcome!

Contents

E-mail address: anton.mellit@univie.ac.at.
Date: June 22, 2018.

1. Introduction

One way to do mathematics (and sometimes also physics) is to start with a space, extract some numerical invariants from it, then package these numbers into functions, and study those functions using algebra. This course should illustrate this point of view.

2. First examples (Practice 1-2)

We would like to calculate some examples to get experience.

For any $m \in \mathbb{Z}$ and $n \in \mathbb{Z}_{>0}$, we would like to calculate

$$f_m(x_1, \dots, x_n) = \sum_{i=1}^{n} \frac{x_i^m}{\prod_{\substack{j=1 \\ j \neq i}}^{n} (x_i - x_j)}.$$

Note first, that for all $m \geq 0$ f_m must be a polynomial in $x_1, \dots, x_n$. Indeed, let

$$P(x_1, \dots, x_n) = f_m(x_1, \dots, x_n) \prod_{1 \leq i < j \leq n} (x_i - x_j).$$

Then P is a polynomial in $x_1, \dots, x_n$. Moreover, it is antisymmetric (permuting any two variables changes its sign). This implies that P vanishes when any two variables coincide. This implies that P is divisible by $x_i - x_j$. So P must be divisible by the product of all the differences $x_i - x_j$ (because the polynomial ring is a unique factorization domain). Therefore

$$f_m(x_1, \dots, x_n) = \frac{P(x_1, \dots, x_n)}{\prod_{1 \leq i < j \leq n} (x_i - x_j)}$$

is a polynomial.

For $0 \leq m \leq n-2$ we must have $f_m = 0$ because otherwise the degree of f_m is $m - n + 1 < 0$, which is impossible for a polynomial.

Note also that

$$f_m(x_1^{-1}, \dots, x_n^{-1}) = (-1)^{n-1} \left(\prod_{i=1}^{n} x_i \right) f_{n-2-m},$$

so if we calculate f_m for $m \geq n-1$, we obtain f_m for $m \leq -1$, and vice versa.

We now can use two approaches.

2.1. **Approach 1.** The first approach uses complex analysis and goes by computing residues of the following function, viewed as a function of t.

$$\varphi_m(x_1,\dots,x_n;t)=\frac{t^m}{\prod_{i=1}^n(t-x_i)}.$$

The total sum of the residues must be zero. The sum of the residues in $t=x_1,\dots,x_n$ gives f_m, so the residues at 0 and ∞ allow us to compute f_m.

2.2. **Approach 2.** The second approach uses only algebra. The main idea of the second approach is to expand the following function as a power series in t:

$$\varphi(x_1,\dots,x_n;t)=\sum_{i=1}^n\frac{1}{(x_i-t)\prod_{\substack{j=1\\j\neq i}}^n(x_i-x_j)}=\sum_{m=0}^\infty t^m f_{-m-1},$$

and show, on the other hand, that

$$\varphi(x_1,\dots,x_n;t)=\frac{(-1)^{n-1}}{\prod_{i=1}^n(x_i-t)}.$$

This gives the formula

$$f_{-m-1}(x_1,\dots,x_n)=(-1)^{n-1}\left(\prod_{i=1}^n x_i\right)^{-1}\sum_{\substack{a_1,\dots,a_n\geq 0\\ \sum a_i=m}}x_1^{-a_1}\cdots x_n^{-a_n},$$

so we have f_m for $m\leq -1$.

2.3. **Result.**

$$f_m(x_1,\dots,x_n)=\begin{cases}h_{m-n+1}(x_1,\dots,x_n) & (m\geq n-1),\\ 0 & (0\leq m\leq n-2),\\ (-1)^{n-1}\left(\prod_{i=1}^n x_i\right)^{-1}h_{-m-1}(x_1^{-1},\dots,x_n^{-1}) & (m\leq -1).\end{cases}$$

2.4. **Calculations in SAGE.** During the course, we would like to do some examples of calculations in SAGE (`http://www.sagemath.org/`). I prefer to use SAGE in command line, but it is possible to use web-interface, and even the free online inteface at `http://sagecell.sagemath.org/`.

To begin, let us first verify our formula for the functions f_m. Take $n=4$.

```
sage: R.<x1,x2,x3,x4>=QQ[]
sage: xx=[x1,x2,x3,x4]
sage: def f(m):
....:     return sum(xx[i]^m/prod(xx[i]-xx[j] for j in
range(4) if j!=i) for i in range(4))
....:
```

```
sage: f(1)
0
sage: f(2)
0
sage: f(3)
1
sage: f(4)
x1 + x2 + x3 + x4
sage: f(5)
x1^2 + x1*x2 + x2^2 + x1*x3 + x2*x3 + x3^2 + x1*x4 + x2*x4
+ x3*x4 + x4^2
sage: f(-1)
(-1)/(x1*x2*x3*x4)
sage: f(-2)
(-x1*x2*x3 - x1*x2*x4 - x1*x3*x4 - x2*x3*x4)
/(x1^2*x2^2*x3^2*x4^2)
sage: f(-2).subs(x1=1/x1,x2=1/x2,x3=1/x3,x4=1/x4)
-x1^2*x2*x3*x4 - x1*x2^2*x3*x4 - x1*x2*x3^2*x4
- x1*x2*x3*x4^2
sage: factor(_)
(-1) * x4 * x3 * x2 * x1 * (x1 + x2 + x3 + x4)
sage: f(-3).subs(x1=1/x1,x2=1/x2,x3=1/x3,x4=1/x4)
-x1^3*x2*x3*x4 - x1^2*x2^2*x3*x4 - x1*x2^3*x3*x4
- x1^2*x2*x3^2*x4 - x1*x2^2*x3^2*x4 - x1*x2*x3^3*x4
- x1^2*x2*x3*x4^2 - x1*x2^2*x3*x4^2 - x1*x2*x3^2*x4^2
- x1*x2*x3*x4^3
sage: factor(_)
(-1) * x4 * x3 * x2 * x1 * (x1^2 + x1*x2 + x2^2 + x1*x3
+ x2*x3 + x3^2 + x1*x4 + x2*x4 + x3*x4 + x4^2)
sage:
```

Next we try to express $x_1^5 + x_2^5$ in terms of e_1, e_2 to illustrate the proof of Theorem 3.3. Note that SAGE automatically displays polynomials in such a way that the main term comes first.

```
sage: R.<x1,x2>=QQ[]
sage: e1=x1+x2
sage: e2=x1*x2
```

```
sage: p5=x1^5+x2^5
sage: p5
x1^5 + x2^5
sage: e1^5
x1^5 + 5*x1^4*x2 + 10*x1^3*x2^2 + 10*x1^2*x2^3 + 5*x1*x2^4
+ x2^5
sage: p5-e1^5
-5*x1^4*x2 - 10*x1^3*x2^2 - 10*x1^2*x2^3 - 5*x1*x2^4
sage: e2*e1^3
x1^4*x2 + 3*x1^3*x2^2 + 3*x1^2*x2^3 + x1*x2^4
sage: p5-e1^5+e2*e1^3*5
5*x1^3*x2^2 + 5*x1^2*x2^3
sage: p5-e1^5+e2*e1^3*5-5*e2^2*e1
0
```

The same formula does not work for three variables, so we need more terms, involving e_3.

```
sage: R.<x1,x2,x3>=QQ[]
sage: p5=x1^5+x2^5+x3^5
sage: e1=x1+x2+x3
sage: e2=x1*x2+x1*x3+x2*x3
sage: e3=x1*x2*x3
sage: p5-e1^5+e2*e1^3*5-5*e2^2*e1
5*x1^3*x2*x3 + 5*x1^2*x2^2*x3 + 5*x1*x2^3*x3
+ 5*x1^2*x2*x3^2 + 5*x1*x2^2*x3^2 + 5*x1*x2*x3^3
sage: e3*e1^2
x1^3*x2*x3 + 2*x1^2*x2^2*x3 + x1*x2^3*x3 + 2*x1^2*x2*x3^2
+ 2*x1*x2^2*x3^2 + x1*x2*x3^3
sage: p5-e1^5+e2*e1^3*5-5*e2^2*e1-5*e3*e1^2
-5*x1^2*x2^2*x3 - 5*x1^2*x2*x3^2 - 5*x1*x2^2*x3^2
sage: p5-e1^5+e2*e1^3*5-5*e2^2*e1-5*e3*e1^2+5*e3*e2
0
```

SAGE can be used, for instance, to list all partitions.

```
sage: Partitions(5)
Partitions of the integer 5
sage: list(Partitions(5))
```

[[5], [4, 1], [3, 2], [3, 1, 1], [2, 2, 1], [2, 1, 1, 1], [1, 1, 1, 1, 1]]

3. Basic algebra of symmetric functions (Lecture 1)

The base field is $\mathbb{C}$, but in most cases can be anything.

Definition 3.1. Let $n \in \mathbb{Z}_{\geq 0}$. A *symmetric function in n variables* is a polynomial $f \in \mathbb{C}[x_1, x_2, \ldots, x_n]$ satisfying any of the following equivalent conditions:

(i) $f(x_1, \ldots, x_{i-1}, x_{i+1}, x_i, x_{i+2}, \ldots, x_n) = f(x_1, \ldots, x_n)$ for $i = 1, 2, \ldots, n-1$.
(ii) $f(x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n) = f(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n)$ for any pair $1 \leq i < j \leq n$.
(iii) $f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}) = f(x_1, \ldots, x_n)$ for any permutation $\pi \in S_n$.

The space of symmetric functions in n variables is denoted by Sym_n.

Suppose $a_1, \ldots, a_n \in \mathbb{Z}_{\geq 0}$. Denote by $\mathrm{coeff}_{a_1,\ldots,a_n} f$ the coefficient of the monomial $x_1^{a_1} \ldots x_n^{a_n}$ in f. This gives a linear map

$$\mathrm{coeff}_{a_1,\ldots,a_n} : \mathrm{Sym}_n \to \mathbb{C}.$$

For any permutation $\pi \in S_n$ and any $f \in \mathrm{Sym}_n$, we have

$$\mathrm{coeff}_{a_1,\ldots,a_n} f = \mathrm{coeff}_{a_{\pi(1)},\ldots,a_{\pi(n)}} f.$$

So it is enough to consider $a_1 \geq a_2 \geq \ldots \geq a_n$.

One can write any $f \in \mathrm{Sym}_n$ uniquely as follows:

$$f = f^{(0)} + f^{(1)} + f^{(2)} + \cdots,$$

where $f^{(d)}$ contains only monomials of total degree d. Denote the space of symmetric functions of degree d by Sym_n^d.

3.1. Monomial basis.

Proposition 3.2. *Let $P(n,d)$ be the set of n-tuples of integers $a = (a_1, \ldots, a_n)$ satisfying $a_i \geq 0$ and $\sum_{i=1}^n a_i = d$. $|P(n,d)|$ denotes the number of such tuples. Then the map*

$$\mathrm{Sym}_n^d \to \mathbb{C}^{|P(n,d)|}, \qquad f \to (\mathrm{coeff}_a f \mid a \in P(n,d))$$

is an isomorphism of vector spaces.

Proof. Injectivity is clear. Surjectivity follows from the following construction. Let $a \in P(n,d)$. Denote

$$m_a(x_1, \ldots, x_n) = \sum_{a'_1,\ldots,a'_n} x_1^{a'_1} \cdots x_n^{a'_n},$$

where the sum goes over all *distinct* permutations of the sequence $(a_1, \dots, a_n)$. Then we have for all $b \in P(n,d)$

$$\operatorname{coeff}_b m_a = \begin{cases} 1 & (b = a) \\ 0 & (b \neq a). \end{cases}$$

□

The above proposition in particular means that m_a where a goes over the set of sequences $a = (a_1 \geq a_2 \geq \cdots \geq a_n \geq 0)$ form a basis of Sym_n.

3.2. **Elementary symmetric polynomials.** These are defined for $k = 1, 2, \dots, n$ by

$$e_k(x_1, \dots, x_n) = \sum_{1 \leq i_1 < i_2 < \dots < i_k \leq n} x_{i_1} x_{i_2} \cdots x_{i_k}.$$

We have

Theorem 3.3. *The ring of symmetric functions is isomorphic to the ring of polynomials in the elementary symmetric polynomials. In other words, for any $f \in \mathrm{Sym}_n$ there exists a unique polynomial in n variables g such that*

$$f(x_1, \dots, x_n) = g(e_1(x_1, \dots, x_n), e_2(x_1, \dots, x_n), \dots, e_n(x_1, \dots, x_n)).$$

Proof. Without loss of generality assume $0 \neq f \in \mathrm{Sym}_n^d$. The main idea is to look at the *main term*. Write f as the sum of *terms*, where each term is a monomial in $x_1, \dots, x_n$ with some non-zero coefficient in $\mathbb{C}$. The main term is defined as follows:

Choose all terms where the power of x_1 is maximal, among those choose all terms where the power of x_2 is maximal, and so on. In the end of this procedure only one term will be left. This is is the main term.

In other words, the main term is the first term in the *reverse lexicographic order*.

The strategy is to cook up a product of e_i is such a way that the main term will be the same as the one of f, and then subtract it from f. Then proceed by "killing" the remaining main term and so on until we get zero.

We can find that

$$\text{main term}(e_1^{b_1} e_2^{b_2} \cdots e_n^{b_n}) = x_1^{b_1 + b_2 + \cdots + b_n} x_2^{b_2 + \cdots b_n} \cdots x_n^{b_n}.$$

So to kill the main term with $x_1^{a_1} \cdots x_n^{a_n}$ we use $b_1 = a_1 - a_2$, $\dots b_i = a_i - a_{i+1}$,$\dots b_n = a_n$.

Another way to look at this construction is as follows. Let us form a matrix by decomposing the products $e_1^{b_1} \cdots e_n^{b_n}$ in the monomial basis. Order the elements of $P(n,d)$ in the reverse lexicographic order, so that we obtain a sequence of tuples $a^{(1)}, a^{(2)}, \dots$. For each i compute the tuple $b^{(i)}$ by the rule above. The matrix is defined by

$$M_{i,j} = \operatorname{coeff}_{a^{(j)}} e_1^{b_1^{(i)}} \cdots e_n^{b_n^{(i)}}.$$

The matrix is upper triangluar with 1 on the diagonal. Therefore it is a linear isomorphism, which is equivalent to the statement of the theorem. □

4. Symmetric functions in infinitely many variables (Lecture 2)

4.1. **Construction.** For a fixed $d \in \mathbb{Z}_{\geq 0}$ consider the sequence of spaces and maps between them:

$$\mathrm{Sym}_1^d \leftarrow \mathrm{Sym}_2^d \leftarrow \mathrm{Sym}_3^d \leftarrow \cdots,$$

where the maps are given as follows (res stands for "restriction"):

$$\mathrm{res}_n : \mathrm{Sym}_n^d \to \mathrm{Sym}_{n-1}^d, \qquad \mathrm{res}_n f(x_1, \ldots, x_n) = f(x_1, \ldots, x_{n-1}, 0).$$

Definition 4.1. A symmetric function of degree d in infinitely many variables is an infinite sequence $f_1, f_2, \ldots$ where $f_n \in \mathrm{Sym}_n^d$ such that for any n we have

$$f_{n-1} = \mathrm{res}_n f_n.$$

The space of symmetric functions of degree d in infinitely many variables is denoted by Sym_∞^d.

Proposition 4.2. *If $n \geq d$ then $\mathrm{Sym}_\infty^d = \mathrm{Sym}_n^d$.*

Proof. Look how res_n looks like in the monomial basis:

$$\mathrm{res}_n m_{a_1,\ldots,a_n} = \begin{cases} 0 & (a_n \neq 0) \\ m_{a_1,\ldots,a_{n-1}} & (a_n = 0). \end{cases}$$

So we have that res_n is surjective for any n, and its kernel is the span of $m_{a_1,\ldots,a_n}$ where $a_1 \geq a_2 \geq \cdots \geq a_n > 0$. In particular, if $n > d$ the the kernel is zero, so res_n is an isomorphism. For any $n \geq d$ an element $f \in \mathrm{Sym}_\infty^d$ is completely determined by its component f_n. □

Sometimes we consider symmetric functions of "mixed degree":

Definition 4.3. A symmetric function in infinitely many variables is a sum for some d

$$f = f^{(0)} + f^{(1)} + \cdots f^{(d)},$$

where $f^{(i)} \in \mathrm{Sym}_\infty^i$.

4.2. **Monomial and elementary symmetric functions.**

4.2.1. *Notations.* A *partition* is a sequence of integers $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m > 0$. We call m the *length* and denote it by $l(\lambda)$. The sum $\sum_{i=1}^{l(\lambda)} \lambda_i$ is called the *size* and is denoted by $|\lambda|$. The set of partitions is denoted by $\mathcal{P}$. The notation $\lambda \vdash n$ means that λ is a partition of size n.

For any $\lambda \in \mathcal{P}$, the corresponding monomial symmetric function in infinitely many variables is defined by

$$m_\lambda(x_1, \ldots, x_n) = \begin{cases} m_{\lambda_1,\ldots,\lambda_{l(\lambda)},0,\cdots,0}(x_1, \ldots, x_n) & (n \geq l(\lambda)) \\ 0 & (n < l(\lambda)). \end{cases}$$

It is easy to check, that setting the last variable x_n equal to zero, we obtain the corresponding function in $n-1$ variables, so the condition of Definition 4.1 is satisfied. Thus we obtain a symmetric function in infinitely many variables.

Using Propositions 4.2 and 3.2, we obtain

Proposition 4.4. *The elements $m_\lambda \in \mathrm{Sym}_\infty$ where λ goes over the set of all partitions $\mathcal{P}$ form a basis of Sym_∞.*

We also define for $k = 1, 2, \ldots$

$$e_k(x_1, \ldots, x_n) = \begin{cases} e_k(x_1, \ldots, x_n) & (n \geq k), \\ 0 & (n < k). \end{cases}$$

Again, this gives a well-defined element of Sym_∞.

Using Proposition 4.2, we extend Theorem 3.3 to symmetric functions in infinitely many variables

Theorem 4.5. *Any element $f \in \mathrm{Sym}_\infty$ can be uniquely written as a polynomial of $e_1, e_2, \ldots$.*

It is convenient when writing symmetric functions in infinitely many variables not to mention the number of variables at all.

5. Relations between e_k, h_k, p_k (Lecture 2)

We also introduce the *complete homogeneous symmetric polynomials*

$$h_k(x_1, x_2, \ldots) = \sum_{i_1 \leq i_2 \leq \cdots \leq i_k} x_{i_1} x_{i_2} \cdots x_{i_k},$$

and the *power sums*

$$p_k(x_1, x_2, \ldots) = \sum_i x_i^k.$$

The following relations are quite easy to prove:

$$p_k = m_{(k)},$$

where (k) denotes the sequence consisting of a single number k.

$$e_k = m_{\underbrace{(1, 1, \ldots, 1)}_{k \text{ ones}}}.$$

$$h_k = \sum_{\lambda \vdash k} m_\lambda.$$

To show more algebraic relations connecting e_k, h_k, p_k it is convenient to use the formal power series ring $\mathrm{Sym}_\infty[[t]]$. Then we have

Proposition 5.1.

$$1+\sum_{k=1}^{\infty} h_k t^k = \frac{1}{1+\sum_{k=1}^{\infty}(-1)^k e_k t^k} = \exp\left(\sum_{k=1}^{\infty}\frac{p_k t^k}{k}\right),$$

$$1+\sum_{k=1}^{\infty}(-1)^k e_k t^k = \frac{1}{1+\sum_{k=1}^{\infty} h_k t^k},$$

$$\sum_{k=1}^{\infty}\frac{p_k t^k}{k} = \log\left(1+\sum_{k=1}^{\infty} h_k t^k\right) = -\log\left(1+\sum_{k=1}^{\infty}(-1)^k e_k t^k\right).$$

Proof. Each of these identities corresponds to an infinite sequence of identities, and to show each of them, by Proposition 4.2, it is enough to check the corresponding identity for finitely many variables. We clearly have

$$\prod_{i=1}^{n}(1-tx_i) = 1+\sum_{k=1}^{\infty}(-1)^k e_k(x_1,\dots,x_n)t^k$$

and

$$\prod_{i=1}^{n}\frac{1}{1-tx_i} = 1+\sum_{k=1}^{\infty} h_k(x_1,\dots,x_n)t^k. \tag{5.1}$$

This implies the first identity. Furthermore, we have

$$\log\left(\prod_{i=1}^{n}\frac{1}{1-tx_i}\right) = \sum_{i=1}^{n}\log\left(\frac{1}{1-tx_i}\right) = \sum_{i=1}^{n}\sum_{m=1}^{\infty}\frac{x_i^m t^m}{m} = \sum_{m=1}^{\infty}\frac{p_m t^m}{m}.$$

Putting these together and using the fact that exp is inverse to log we deduce all these identities. □

These identities allow us to express h_k, p_k, e_k in terms of each other. Therefore we have

Corollary 5.2. *Any element $f \in \mathrm{Sym}_\infty$ can be uniquely written as a polynomial of $h_1, h_2, \dots$.*

Corollary 5.3. *Any element $f \in \mathrm{Sym}_\infty$ can be uniquely written as a polynomial of $p_1, p_2, \dots$.*

6. Algebraic manipulations (Practice 3-4)

6.1. **Conversion from monomial to elementary symmetric functions.** We continue to illustrate proof of Theorem 3.3. Let us compute the matrix

$$M_{i,j} = \mathrm{coeff}_{a^{(j)}}\, e_1^{b_1^{(i)}}\cdots e_n^{b_n^{(i)}}.$$

We pick $n = 4$ (the size of the partitions) and list all partitions n in the reverse lexicographic order:

sage: n=4

```
sage: ps=list(Partitions(n))
sage: ps
[[4], [3, 1], [2, 2], [2, 1, 1], [1, 1, 1, 1]]
sage: sorted(ps)
[[1, 1, 1, 1], [2, 1, 1], [2, 2], [3, 1], [4]]
sage: list(reversed(sorted(ps)))
[[4], [3, 1], [2, 2], [2, 1, 1], [1, 1, 1, 1]]
sage: ps=list(reversed(sorted(ps)))
```

Next we write a function that turns a partition λ into the corresponding sequence a:

```
sage: def lambda2a(lam):
....:     res = []
....:     for i in range(len(lam)-1):
....:         res.append(lam[i]-lam[i+1])
....:     res.append(lam[-1])
....:     return res
....:
sage: lambda2a([3,2])
[1, 2]
sage: lambda2a([3,1])
[2, 1]
```

Remember that this means that the main term of $e_1e_2^2$ is $m_{3,2}$, the main term of $e_1^2e_2$ is $m_{3,1}$. Let us verify this. We implement elementary symmetric functions:

```
sage: R=PolynomialRing(QQ, names=['x'+str(i) for i in range(n)])
sage: R
Multivariate Polynomial Ring in x0, x1, x2, x3 over Rational Field
sage: def elementary(k,n):
....:     if n<k:
....:         return 0
....:     if n==k:
....:         return prod(R.gen(i) for i in range(n))
....:     if k==1:
....:         return sum(R.gen(i) for i in range(n))
....:     return elementary(k, n-1) + R.gen(n-1)*elementary(k-1,n-1)
....:
```

```
sage: elementary(1,3)
x0 + x1 + x2
sage: elementary(2,3)
x0*x1 + x0*x2 + x1*x2
sage: elementary(3,3)
x0*x1*x2
sage: elementary(3,4)
x0*x1*x2 + x0*x1*x3 + x0*x2*x3 + x1*x2*x3
sage: elementary(2,4)
x0*x1 + x0*x2 + x1*x2 + x0*x3 + x1*x3 + x2*x3
```

Now we compute the list of a-sequences and for each a-sequence the corresponding product of the elementary symmetric functions:

```
sage: aa_list=[lambda2a(lam) for lam in ps]
sage: aa_list
[[4], [2, 1], [0, 2], [1, 0, 1], [0, 0, 0, 1]]
sage: el_list=[prod(elementary(i+1,n)^ai for i,ai in enumerate(aa))
   for aa in aa_list]
sage: el_list

[x0^4 + 4*x0^3*x1 + 6*x0^2*x1^2 + 4*x0*x1^3 + x1^4 + 4*x0^3*x2 + 12*x0
   ^2*x1*x2 + 12*x0*x1^2*x2 + 4*x1^3*x2 + 6*x0^2*x2^2 + 12*x0*x1*x2^2
   + 6*x1^2*x2^2 + 4*x0*x2^3 + 4*x1*x2^3 + x2^4 + 4*x0^3*x3 + 12*x0^2*
   x1*x3 + 12*x0*x1^2*x3 + 4*x1^3*x3 + 12*x0^2*x2*x3 + 24*x0*x1*x2*x3
   + 12*x1^2*x2*x3 + 12*x0*x2^2*x3 + 12*x1*x2^2*x3 + 4*x2^3*x3 + 6*x0
   ^2*x3^2 + 12*x0*x1*x3^2 + 6*x1^2*x3^2 + 12*x0*x2*x3^2 + 12*x1*x2*x3
   ^2 + 6*x2^2*x3^2 + 4*x0*x3^3 + 4*x1*x3^3 + 4*x2*x3^3 + x3^4,
x0^3*x1 + 2*x0^2*x1^2 + x0*x1^3 + x0^3*x2 + 5*x0^2*x1*x2 + 5*x0*x1^2*
   x2 + x1^3*x2 + 2*x0^2*x2^2 + 5*x0*x1*x2^2 + 2*x1^2*x2^2 + x0*x2^3 +
    x1*x2^3 + x0^3*x3 + 5*x0^2*x1*x3 + 5*x0*x1^2*x3 + x1^3*x3 + 5*x0
   ^2*x2*x3 + 12*x0*x1*x2*x3 + 5*x1^2*x2*x3 + 5*x0*x2^2*x3 + 5*x1*x2
   ^2*x3 + x2^3*x3 + 2*x0^2*x3^2 + 5*x0*x1*x3^2 + 2*x1^2*x3^2 + 5*x0*
   x2*x3^2 + 5*x1*x2*x3^2 + 2*x2^2*x3^2 + x0*x3^3 + x1*x3^3 + x2*x3^3,
```

```
x0^2*x1^2 + 2*x0^2*x1*x2 + 2*x0*x1^2*x2 + x0^2*x2^2 + 2*x0*x1*x2^2 +
    x1^2*x2^2 + 2*x0^2*x1*x3 + 2*x0*x1^2*x3 + 2*x0^2*x2*x3 + 6*x0*x1*x2
    *x3 + 2*x1^2*x2*x3 + 2*x0*x2^2*x3 + 2*x1*x2^2*x3 + x0^2*x3^2 + 2*x0
    *x1*x3^2 + x1^2*x3^2 + 2*x0*x2*x3^2 + 2*x1*x2*x3^2 + x2^2*x3^2,
x0^2*x1*x2 + x0*x1^2*x2 + x0*x1*x2^2 + x0^2*x1*x3 + x0*x1^2*x3 + x0^2*
    x2*x3 + 4*x0*x1*x2*x3 + x1^2*x2*x3 + x0*x2^2*x3 + x1*x2^2*x3 + x0*
    x1*x3^2 + x0*x2*x3^2 + x1*x2*x3^2,
x0*x1*x2*x3]
sage:
```

We can see that the leading terms x_0^4, $x_0^3x_1$, ... are precisely given by the partitions $(4), (3,1), \ldots$. Next let us compute the matrix of all coefficients of the products of elementary symmetric functions. We create the list of all monomials:

```
sage: mon_list=[prod(R.gen(i)^ai for i,ai in enumerate(lam)) for lam
    in ps]
sage: mon_list
[x0^4, x0^3*x1, x0^2*x1^2, x0^2*x1*x2, x0*x1*x2*x3]
```

And then we create a matrix

```
sage: M=matrix(len(ps), len(ps))
sage: M


[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
```

And finally we put the coefficients into the matrix:

```
sage: for i in range(len(ps)):
....:     for j in range(len(ps)):
....:         M[i,j] = el_list[i].monomial_coefficient(mon_list[j])
....:
sage: M


[ 1  4  6 12 24]
[ 0  1  2  5 12]
```

```
[ 0  0  1  2  6]
[ 0  0  0  1  4]
[ 0  0  0  0  1]
```

As expected, the resulting matrix is upper-triangular with 1 on the diagonal, so it is invertible.

```
sage: M.inverse()

[ 1 -4  2  4 -4]
[ 0  1 -2 -1  4]
[ 0  0  1 -2  2]
[ 0  0  0  1 -4]
[ 0  0  0  0  1]
```

We can do the same for $n = 5$:

```
sage: n=5
sage: ps=list(Partitions(n))
sage: ps=list(reversed(sorted(ps)))
sage: ps
[[5], [4, 1], [3, 2], [3, 1, 1], [2, 2, 1], [2, 1, 1, 1], [1, 1, 1, 1,
    1]]
sage: aa_list=[lambda2a(lam) for lam in ps]
sage: R=PolynomialRing(QQ, names=['x'+str(i) for i in range(n)])
sage: el_list=[prod(elementary(i+1,n)^ai for i,ai in enumerate(aa))
   for aa in aa_list]
sage: mon_list=[prod(R.gen(i)^ai for i,ai in enumerate(lam)) for lam
   in ps]
sage: el_list

[x0^5 + 5*x0^4*x1 + 10*x0^3*x1^2 + 10*x0^2*x1^3 +...
sage: mon_list

[x0^5,
x0^4*x1,
x0^3*x1^2,
x0^3*x1*x2,
x0^2*x1^2*x2,
```

```
x0^2*x1*x2*x3,
x0*x1*x2*x3*x4]
sage: M=matrix(len(ps), len(ps))
sage: M

[0 0 0 0 0 0 0]
[0 0 0 0 0 0 0]
[0 0 0 0 0 0 0]
[0 0 0 0 0 0 0]
[0 0 0 0 0 0 0]
[0 0 0 0 0 0 0]
[0 0 0 0 0 0 0]
sage: for i in range(len(ps)):
....:     for j in range(len(ps)):
....:         M[i,j] = el_list[i].monomial_coefficient(mon_list[j])
....:
sage: M

[  1   5  10  20  30  60 120]
[  0   1   3   7  12  27  60]
[  0   0   1   2   5  12  30]
[  0   0   0   1   2   7  20]
[  0   0   0   0   1   3  10]
[  0   0   0   0   0   1   5]
[  0   0   0   0   0   0   1]
sage: M.inverse()

[ 1 -5  5  5 -5 -5  5]
[ 0  1 -3 -1  5  1 -5]
[ 0  0  1 -2 -1  5 -5]
[ 0  0  0  1 -2 -1  5]
[ 0  0  0  0  1 -3  5]
[ 0  0  0  0  0  1 -5]
[ 0  0  0  0  0  0  1]
```

Recall that the inverse matrix can be used to express the elements of the monomial basis in terms of e_i. For instance let us use the first row of the matrix M^{-1}:

```
sage: el_list[0]-5*el_list[1]+5*el_list[2]+5*el_list[3]-5*el_list
    [4]-5*el_list[5]+5*el_list[6]
x0^5 + x1^5 + x2^5 + x3^5 + x4^5
```

6.2. **Using SymmetricFunctions in SAGE.** SAGE has a library for symmetric functions in infinitely many variables, so we do not need to do the computations by hand. Here are some examples:

```
sage: Sym=SymmetricFunctions(QQ)
sage: Sym
Symmetric Functions over Rational Field
sage: ee=Sym.elementary()
sage: ee
Symmetric Functions over Rational Field in the elementary basis
sage: mm=Sym.monomial()
sage: mm
Symmetric Functions over Rational Field in the monomial basis
sage: mm[3,2]
m[3, 2]
sage: mm[5]
m[5]
sage: ee(mm[5])
e[1, 1, 1, 1, 1] - 5*e[2, 1, 1, 1] + 5*e[2, 2, 1] + 5*e[3, 1, 1] - 5*e
    [3, 2] - 5*e[4, 1] + 5*e[5]
sage: ee(mm[4,1])
e[2, 1, 1, 1] - 3*e[2, 2, 1] - e[3, 1, 1] + 5*e[3, 2] + e[4, 1] - 5*e
    [5]
```

6.3. **Power series identities.** SAGE can compute with formal power series (to a given finite number of terms), as we had in Section 5. We check the identities of Proposition 5.1:

```
sage: Rt.<t>=PowerSeriesRing(QQ)
sage: Rt
Power Series Ring in t over Rational Field
sage: 1/(1-t)
```

```
1 + t + t^2 + t^3 + t^4 + t^5 + t^6 + t^7 + t^8 + t^9 + t^10 + t^11 +
    t^12 + t^13 + t^14 + t^15 + t^16 + t^17 + t^18 + t^19 + O(t^20)
sage: Rt.<t>=PowerSeriesRing(R)
sage: Rt
Power Series Ring in t over Multivariate Polynomial Ring in x0, x1, x2
    , x3, x4 over Rational Field
sage: E=sum(elementary(i,n)*t^i for i in range(1,6))+1
sage: 1/E+O(t^4)
1 + (-x0 - x1 - x2 - x3 - x4)*t + (x0^2 + x0*x1 + x1^2 + x0*x2 + x1*x2
     + x2^2 + x0*x3 + x1*x3 + x2*x3 + x3^2 + x0*x4 + x1*x4 + x2*x4 + x3
    *x4 + x4^2)*t^2 + (-x0^3 - x0^2*x1 - x0*x1^2 - x1^3 - x0^2*x2 - x0*
    x1*x2 - x1^2*x2 - x0*x2^2 - x1*x2^2 - x2^3 - x0^2*x3 - x0*x1*x3 -
    x1^2*x3 - x0*x2*x3 - x1*x2*x3 - x2^2*x3 - x0*x3^2 - x1*x3^2 - x2*x3
    ^2 - x3^3 - x0^2*x4 - x0*x1*x4 - x1^2*x4 - x0*x2*x4 - x1*x2*x4 - x2
    ^2*x4 - x0*x3*x4 - x1*x3*x4 - x2*x3*x4 - x3^2*x4 - x0*x4^2 - x1*x4
    ^2 - x2*x4^2 - x3*x4^2 - x4^3)*t^3 + O(t^4)
sage: log(E+O(t^4))
(x0 + x1 + x2 + x3 + x4)*t + (-1/2*x0^2 - 1/2*x1^2 - 1/2*x2^2 - 1/2*x3
    ^2 - 1/2*x4^2)*t^2 + (1/3*x0^3 + 1/3*x1^3 + 1/3*x2^3 + 1/3*x3^3 +
    1/3*x4^3)*t^3 + O(t^4)
```

6.4. **More combinatorics.** The entries of matrix M are positive integers, and so they should have some combinatorial interpretation. As an exercise, we suggest to find such an interpretation.

Let us look at M for $n = 6$:

$$\begin{pmatrix}
1 & 6 & 15 & 30 & 20 & 60 & 120 & 90 & 180 & 360 & 720 \\
0 & 1 & 4 & 9 & 6 & 22 & 48 & 36 & 78 & 168 & 360 \\
0 & 0 & 1 & 2 & 2 & 8 & 18 & 15 & 34 & 78 & 180 \\
0 & 0 & 0 & 1 & 0 & 3 & 10 & 6 & 18 & 48 & 120 \\
0 & 0 & 0 & 0 & 1 & 3 & 6 & 6 & 15 & 36 & 90 \\
0 & 0 & 0 & 0 & 0 & 1 & 3 & 3 & 8 & 22 & 60 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 9 & 30 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 6 & 20 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 15 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

We can notice 2 extra zeroes: one in position $M_{4,5}$, and another one on $M_{7,8}$. This correspond to pairs of partitions $(4,1,1),(3,3)$ and $(3,1,1,1),(2,2,2)$. Explanation for these zeroes is given by the *dominance order* on partitions.

Definition 6.1. If μ and λ are partitions of n, we say that μ *dominates* λ (denoted $\mu \succeq \lambda$) if for all i we have

$$\sum_{j=1}^{i} \mu_j \geq \sum_{j=1}^{i} \lambda_j.$$

We treat μ_j as zero if $j > l(\mu)$.

It turns out that

Proposition 6.2. *$M_{i,j} \neq 0$ if and only if $\lambda^{(i)} \succeq \lambda^{(j)}$, where $\lambda^{(i)}$ denotes the partition corresponding to the row i of the matrix M.*

If $\lambda^{(i)} \succeq \lambda^{(j)}$, then $\lambda^{(i)}$ greater than $\lambda^{(j)}$ lexicographically, so $i \leq j$, but the inverse statement is not true. Indeed, we have $(4,1,1)$ is greater than $(3,3)$ lexicographically, but not in the dominance order. This explains the zeroes.

In the rest of the class we try to sketch proof of Proposition 6.2.

7. Schur functions (Lecture 3)

Let us begin by recalling the Vandermonde formula:

Proposition 7.1. *For any $n \geq 1$ we have*

$$\det(x_i^{n-j})_{i,j=1}^{n} = \prod_{1 \leq i < j \leq n} (x_i - x_j).$$

Proof. The matrix on the left hand side looks like

$$D(x) = \begin{pmatrix} x_1^{n-1} & x_1^{n-2} & \cdots & 1 \\ x_2^{n-1} & x_2^{n-2} & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ x_n^{n-1} & x_n^{n-2} & \cdots & 1 \end{pmatrix}$$

First observe that setting $x_i = x_j$ for any $i \neq j$ makes the determinant zero. Therefore the determinant, as a polynomial in $x_1, \ldots, x_j$ must be divisible by $x_i - x_j$. Since all differences $x_i - x_j$ are relatively prime, it must be divisible by the product

$$\Delta(x) := \prod_{1 \leq i < j \leq n} (x_i - x_j).$$

So we have $D(x) = \Delta(x)U(x)$ for some polynomial $U(x)$. We have that $\Delta(x)$ and $D(x)$ are both homogeneous of degrees $\binom{n}{2}$. So U must be constant. Finally we notice that the main term of

both sides is given by

$$x_1^{n-1}x_2^{n-2}\cdots x_{n-1}.$$

So we necessarily have $U(x)=1$. □

We will use the notation

$$\Delta(x) := \prod_{1\le i<j\le n}(x_i - x_j).$$

It is interesting to consider the generalized Vandermonde determinant for any sequence $a = (a_1,\dots,a_n)$, $a_1 > a_2 > \ldots > a_n \ge 0$:

$$D_a(x) := \det(x_i^{a_j})_{i,j=1}^n = \det\begin{pmatrix} x_1^{a_1} & x_1^{a_2} & \cdots & x_1^{a_n} \\ x_2^{a_1} & x_2^{a_2} & \cdots & x_2^{a_n} \\ \vdots & \vdots & & \vdots \\ x_n^{a_1} & x_n^{a_2} & \cdots & x_n^{a_n} \end{pmatrix} \tag{7.1}$$

Let us first list some properties of $D_a(x)$ that are clear from the definition:

(i) $D_a(x)$ is a homogeneous polynomial of degree $\sum_{i=1} a_i$.
(ii) For any $i<j$ we have $D_a(x_1,\dots,x_i,\dots,x_j,\dots,x_n) = -D_a(x_1,\dots,x_j,\dots,x_i,\dots,x_n)$, i.e. $D_a(x)$ is antisymmetric.
(iii) In particular, if we set $x_i = x_j$ for $i\ne j$, then $D_a(x)$ vanishes.
(iv) The main term of $D_a(x)$ is $x_1^{a_1}\cdots x_n^{a_n}$.

From the property (iii) we know that $D_a(x)$ is divisible by $\Delta(x)$. Let us denote

$$U_a(x) := \frac{D_a(x)}{\Delta(x)}.$$

Now we have $U_a(x) \in \mathrm{Sym}_n^{\sum_{i=1}^n a_i - \binom{n}{2}}$. Notice that for a partition λ we can define a sequence a by $a_i = \lambda_i + n - i$ (we define $\lambda_i = 0$ if $i > l(\lambda)$). Then we obtain a sequence satisfying $a_1 > a_2 > \ldots > a_n >= 0$. In fact, this is a bijection between partitions of length $<= n$ and strictly decreasing sequences of length n.

Definition 7.2. For any partition λ we define the *Schur function* in n variables by

$$s_\lambda(x) := U_{\lambda_1+n-1,\lambda_2+n-2,\dots,\lambda_n}(x).$$

Proposition 7.3. *(i) For any partition λ of length $\le n$ we have $s_\lambda(x) \in \mathrm{Sym}_n^{|\lambda|}$.*

(ii) For any $\lambda\in\mathcal{P}$ the following rule defines a symmetric function in infinitely many variables, for any $n\ge 1$ we set

$$s_\lambda(x_1,\dots,x_n) = \begin{cases} s_\lambda(x_1,\dots,x_n) & (n\ge l(\lambda)) \\ 0 & (n < l(\lambda)). \end{cases}$$

(iii) If $n\le l(\lambda)$, the main term of $s_\lambda(x_1,\dots,x_n)$ is $x_1^{\lambda_1}\cdots x_{l(\lambda)}^{\lambda_{l(\lambda)}}$.

Proof. The property (i) follows from the fact that both $D_a(x)$ and $\Delta(x)$ are anti-symmetric.

To prove (ii) we need to show that for λ of length $\leq n$ we have

$$s_\lambda(x_1,\dots,x_{n-1},0)=\begin{cases} s_\lambda(x_1,\dots,x_{n-1}) & (l(\lambda)<n)\\ 0 & \text{otherwise.}\end{cases}$$

Suppose $l(\lambda)\geq n$. Then for the corresponding sequence a we have $a_n>0$, so the determinant (7.1) vanishes. If on the other hand $l(\lambda)<n$, then $a_n=0$. In this case we obtain

$$D_a(x_1,\dots,x_{n-1},0)=x_1\cdots x_n D_{a_1-1,a_2-1,\dots,a_{n-1}-1}(x_1,\dots,x_{n-1}),$$

which leads to the right formula also in this case.

To prove (iii) we just divide the main term of $D_a(x)$ by the main term of $\Delta(x)$. □

Now it makes sense to ask how to express s_λ in terms of other symmetric functions. We have

Theorem 7.4 (Jacobi-Trudi formula)**.** *For any partition* λ

$$s_\lambda=\det(h_{\lambda_i+j-i})_{i,j=1}^{l(\lambda)}, \tag{7.2}$$

where we set $h_0=1$ *and* $h_i=0$ *for* $i<0$.

Proof. Let us expand $U_a(x)$ in x_1. First we expand the determinant (7.1) in the first row:

$$D_a(x)=\sum_{i=1}^n(-1)^{i-1}x_1^{a_i}D_{a_1,\dots,\widehat{a_i},\dots,a_n}(x_2,\dots,x_n).$$

Here we denote by $a_1,\dots,\widehat{a_i},\dots,a_n$ the sequence obtained by throwing a_i away. We also write

$$\Delta(x_1,\dots,x_n)=\Delta(x_2,\dots,x_n)\prod_{i=2}^n(x_1-x_i).$$

Dividing one by the other we obtain a formula that allows us to compute U_a recursively in the n:

$$U_a(x)=\frac{\sum_{i=1}^n(-1)^{i-1}x_1^{a_i}U_{a_1,\dots,\widehat{a_i},\dots,a_n}(x_2,\dots,x_n)}{\prod_{i=2}^n(x_1-x_i)}.$$

The difficulty is that we have to divide one polynomial by another. To calculate this division we use the following idea: we first replace x_1 by z^{-1} and then expand everything as a power series in z. Then we will use the fact that the result of the division is a polynomial in z^{-1}.

$$U_a(z^{-1},x_2,\dots,x_n)=\left(\sum_{i=1}^n(-1)^{i-1}z^{-a_i}U_{a_1,\dots,\widehat{a_i},\dots,a_n}(x_2,\dots,x_n)\right)\frac{z^{n-1}}{\prod_{i=2}^n(1-x_iz)}.$$

Using (5.1) we can write

$$\frac{1}{\prod_{i=2}^n(1-x_iz)}=\sum_{k=0}^\infty h'_kz^k,$$

where we have denoted $h'_k = h_k(x_2, \dots, x_n)$. So we have the following power series expansion of $U_a(z^{-1}, \dots)$:

$$U_a(z^{-1}, x_2, \dots, x_n) = \sum_{i=1}^{n} \sum_{k=0}^{\infty} (-1)^{i-1} z^{n-1-a_i+k} U_{a_1,\dots,\widehat{a_i},\dots,a_n}(x_2, \dots, x_n) h'_k.$$

On the other hand we know, that this expression must be a polynomial in $z^{-1}, x_2, \dots, x_n$. So the terms with $n - 1 - a_i + k > 0$ must cancel out. We obtain

$$U_a(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} \sum_{k=0}^{a_i+1-n} x_1^{a_i+1-n-k} U_{a_1,\dots,\widehat{a_i},\dots,a_n}(x_2, \dots, x_n) h'_k.$$

Using the formula $h_k = \sum_{i=0}^{k} h'_k x_1^{k-i}$ we can simplify it to

$$U_a(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} (-1)^{i-1} h_{a_i+1-n} U_{a_1,\dots,\widehat{a_i},\dots,a_n}(x_2, \dots, x_n). \tag{7.3}$$

Now we can complete the proof by induction on n. We want to show that $U_a(x) = \det(h_{a_i+j-n})_{i,j=1}^n$. So we can assume that this identity holds for the terms $U_{a_1,\dots,\widehat{a_i},\dots,a_n}(x_2, \dots, x_n)$. Then (7.3) can be recognized as the expansion of a following determinant in the first row:

$$U_a(x_1, \dots, x_n) = \det \begin{pmatrix} h_{a_1+1-n} & h'_{a_1+2-n} & \dots & h'_{a_1+n-n} \\ h_{a_2+1-n} & h'_{a_2+2-n} & \dots & h'_{a_2+n-n} \\ \vdots & \vdots & & \vdots \\ h_{a_n+1-n} & h'_{a_n+2-n} & \dots & h'_{a_n+n-n} \end{pmatrix}.$$

This is almost what we need to show the formula for U_a. The only problem is that we need to replace h' by h. We have $h_k = h'_k + x_1 h_{k-1}$ for any $k' in \mathbb{Z}$. So multiplying the first column by x_1 and adding it to the second will not change the determinant, and transform h' into h. Proceeding in this way column by column we obtain the identity $U_a(x) = \det(h_{a_i+j-n})_{i,j=1}^n$, which directly translates into the required identity for $s_\lambda(x)$. □

Remark 7.5. Notice that in Theorem 7.4 we can replace λ by a sequence $\lambda_1, \dots, \lambda_{l(\lambda)}, 0, \dots, 0$ with arbitrary many zeros. Indeed, if $\lambda_n = 0$ then the last row of the matrix in (7.2) looks like $(0, 0, \dots, 0, 1)$, so the determinant can be replaced by a smaller one.

8. Experiments with Schur functions (Practice 5)

As before, we try to implement Schur functions by hand first, and later we will use the corresponding SAGE library.

The number of variables is $N = 4$.

```
sage: N=4
sage: R = PolynomialRing(QQ, names=['x'+str(i) for i in range(N)])
```

```
sage: R
Multivariate Polynomial Ring in x0, x1, x2, x3 over Rational Field
sage: xs = R.gens()
sage: xs
(x0, x1, x2, x3)
```

We write the function `schur` step by step to see the intermediate results

```
sage: def schur(lam):
....:     lam = list(lam)
....:     while len(lam)<N:
....:         lam.append(0)
....:     for i in range(N):
....:         lam[i] += N-1-i
....:     print lam
....:
sage: schur([2])
[5, 2, 1, 0]
sage: def schur(lam):
....:     lam = list(lam)
....:     while len(lam)<N:
....:         lam.append(0)
....:     for i in range(N):
....:         lam[i] += N-1-i
....:     a = lam
....:     M = matrix(R, N, N)
....:     for i in range(N):
....:         for j in range(N):
....:             M[i,j]=xs[i]^a[j]
....:     print M
....:
sage: schur([2])
[x0^5 x0^2  x0   1]
[x1^5 x1^2  x1   1]
[x2^5 x2^2  x2   1]
[x3^5 x3^2  x3   1]
sage: def schur0(lam):
```

```
....:     lam = list(lam)
....:     while len(lam)<N:
....:         lam.append(0)
....:     for i in range(N):
....:         lam[i] += N-1-i
....:     a = lam
....:     M = matrix(R, N, N)
....:     for i in range(N):
....:         for j in range(N):
....:             M[i,j] = xs[i]^a[j]
....:     return M.det()
....:
sage: schur0([2])
x0^5*x1^2*x2 - x0^2*x1^5*x2 - x0^5*x1*x2^2 + x0*x1^5*x2^2 + x0^2*x1*x2
    ^5 - x0*x1^2*x2^5 - x0^5*x1^2*x3 + x0^2*x1^5*x3 + x0^5*x2^2*x3 - x1
    ^5*x2^2*x3 - x0^2*x2^5*x3 + x1^2*x2^5*x3 + x0^5*x1*x3^2 - x0*x1^5*
    x3^2 - x0^5*x2*x3^2 + x1^5*x2*x3^2 + x0*x2^5*x3^2 - x1*x2^5*x3^2 -
    x0^2*x1*x3^5 + x0*x1^2*x3^5 + x0^2*x2*x3^5 - x1^2*x2*x3^5 - x0*x2
    ^2*x3^5 + x1*x2^2*x3^5
sage: factor(_)
(-1) * (x2 - x3) * (-x1 + x2) * (x1 - x3) * (-x0 + x2) * (-x0 + x1) *
    (x0 - x3) * (x0^2 + x0*x1 + x1^2 + x0*x2 + x1*x2 + x2^2 + x0*x3 +
    x1*x3 + x2*x3 + x3^2)
```

What we call `schur0` is not the Schur function yet, it is the determinant, we still need to divide by $\Delta(x)$:

```
sage: def schur(lam):
....:     res = schur0(lam)
....:     for j in range(N):
....:         for i in range(j):
....:             res /= xs[i]-xs[j]
....:     return res
....:
sage: schur([2])
x0^2 + x0*x1 + x1^2 + x0*x2 + x1*x2 + x2^2 + x0*x3 + x1*x3 + x2*x3 +
    x3^2
```

```
sage: schur([1])
x0 + x1 + x2 + x3
sage: schur([0])
1
sage: schur([])
1
sage: schur([1,1])
x0*x1 + x0*x2 + x1*x2 + x0*x3 + x1*x3 + x2*x3
sage: schur([3])
x0^3 + x0^2*x1 + x0*x1^2 + x1^3 + x0^2*x2 + x0*x1*x2 + x1^2*x2 + x0*x2
    ^2 + x1*x2^2 + x2^3 + x0^2*x3 + x0*x1*x3 + x1^2*x3 + x0*x2*x3 + x1*
    x2*x3 + x2^2*x3 + x0*x3^2 + x1*x3^2 + x2*x3^2 + x3^3
```

So we implemented the Schur function and computed a few examples. For instance, we can recognize that $s_3 = h_3$, $s_2 = h_2$, $s_1 = h_1$, $s_{1,1} = e_2$.

After we feel that we understand the definition of Schur functions, we can use the SAGE library. We create various symmetric function bases:

```
Sym = SymmetricFunctions(QQ)
ss = Sym.schur()
ss
hh = Sym.homogeneous()
ee = Sym.elementary()
mm = Sym.monomial()
```

Then we try to expand Schur functions in various bases:

```
sage: ss[2]
s[2]
sage: ss[3]
s[3]
sage: hh(ss[3])
h[3]
sage: hh(ss[4])
h[4]
sage: hh(ss[3,1])
h[3, 1] - h[4]
sage: hh(ss[4,1])
```

```
h[4, 1] - h[5]
sage: hh(ss[4,2])
h[4, 2] - h[5, 1]
sage: hh(ss[3,2])
h[3, 2] - h[4, 1]
```

We see that when the partition is given by a single number, we have $s_{(k)} = h_k$. When the partition has two numbers, it is a 2×2 determinant consisting of h_k. This illustrates Theorem 7.4.

```
sage: ee(ss[1,1])
e[2]
sage: ee(ss[1,1,1])
e[3]
sage: ee(ss[1,1,1,1])
e[4]
sage: ee(ss[2,1,1])
e[3, 1] - e[4]
```

Here we can notice that the Schur functions $s_{(1,\dots,1)}$ coincide with e_k, and if we have 2 then we probably obtain a 2×2 determinant. In fact, the following is true. Let $\omega : \mathrm{Sym} \to \mathrm{Sym}$ be the algebra homomorphism that interchanges e_k and h_k for each k (by Proposition 5.1, if it sends e_k to h_k then automatically h_k will go to e_k). Then we have $\omega s_\lambda = s_{\lambda'}$ for any partition λ where λ' is the conjugate partition. The conjugate partition is defined by representing λ as a diagram consisting of rows of boxes of lengths $\lambda_1, \lambda_2, \dots$, and then applying the symmetry with respect to the diagonal line. For instance, $\underbrace{(1,1,\dots,1)}_{k \text{ ones}}$ goes to (k). $(2,1,1)$ goes to $(3,1)$, which explains the last computation. So there is a second Jacobi-Trudi identity:

$$s_\lambda = \det(h_{\lambda'_i+j-i})_{i,j=1}^{l(\lambda')},$$

Next, we express in the monomial basis. In the first two evaluations we match the computation using the library with our own computation. Then we try to experiment.

```
sage: mm(ss[2,1,1])
3*m[1, 1, 1, 1] + m[2, 1, 1]
sage: schur([2,1,1])
x0^2*x1*x2 + x0*x1^2*x2 + x0*x1*x2^2 + x0^2*x1*x3 + x0*x1^2*x3 + x0^2*
    x2*x3 + 3*x0*x1*x2*x3 + x1^2*x2*x3 + x0*x2^2*x3 + x1*x2^2*x3 + x0*
    x1*x3^2 + x0*x2*x3^2 + x1*x2*x3^2
sage: mm(ss[3,1])
```

```
3*m[1, 1, 1, 1] + 2*m[2, 1, 1] + m[2, 2] + m[3, 1]
sage: mm(ss[3,2])
5*m[1, 1, 1, 1, 1] + 3*m[2, 1, 1, 1] + 2*m[2, 2, 1] + m[3, 1, 1] + m
    [3, 2]
sage: mm(ss[3,3])
5*m[1, 1, 1, 1, 1, 1] + 3*m[2, 1, 1, 1, 1] + 2*m[2, 2, 1, 1] + m[2, 2,
     2] + m[3, 1, 1, 1] + m[3, 2, 1] + m[3, 3]
sage: mm(ss[3,2,1])
16*m[1, 1, 1, 1, 1, 1] + 8*m[2, 1, 1, 1, 1] + 4*m[2, 2, 1, 1] + 2*m[2,
     2, 2] + 2*m[3, 1, 1, 1] + m[3, 2, 1]
```

We see that the main term (appearing last) of s_λ is m_λ, as we have proved. The coefficients are always positive integers, which suggests that they may be counting some combinatorial objects.

It is not interesting to compute multiplication in the h, p, e bases:

```
sage: hh[2]*hh[3]
h[3, 2]
sage: hh[2]*hh[3]*hh[1]
h[3, 2, 1]
```

In the Schur basis it gets more interesting. First we try to multiply by $s_1 = e_1 = h_1 = p_1$:

```
sage: ss[2]*ss[1]
s[2, 1] + s[3]
sage: ss[3,2,1]*ss[1]
s[3, 2, 1, 1] + s[3, 2, 2] + s[3, 3, 1] + s[4, 2, 1]
sage: ss[3,2]*ss[1]
s[3, 2, 1] + s[3, 3] + s[4, 2]
sage: ss[4,2]*ss[1]
s[4, 2, 1] + s[4, 3] + s[5, 2]
```

It is surprising that the coefficients are always 1. Upon further investigation we notice, that the partitions that appear in the expansion of $s_\lambda s_{(1)}$ are exactly the partitions whose diagram contains λ and one extra box. This is the first example of *Pieri rules*. We next try to multiply by $s_{(2)} = h_2$, $s_{(3)} = h_3$, etc.:

```
sage: ss[4,2]*ss[2]
s[4, 2, 2] + s[4, 3, 1] + s[4, 4] + s[5, 2, 1] + s[5, 3] + s[6, 2]
sage: ss[4,2,1]*ss[2]
```

```
s[4, 2, 2, 1] + s[4, 3, 1, 1] + s[4, 3, 2] + s[4, 4, 1] + s[5, 2, 1,
    1] + s[5, 2, 2] + s[5, 3, 1] + s[6, 2, 1]
sage: ss[4,2,1]*ss[3]
s[4, 3, 2, 1] + s[4, 4, 1, 1] + s[4, 4, 2] + s[5, 2, 2, 1] + s[5, 3,
    1, 1] + s[5, 3, 2] + s[5, 4, 1] + s[6, 2, 1, 1] + s[6, 2, 2] + s[6,
     3, 1] + s[7, 2, 1]
sage: ss[3,2,1]*ss[3]
s[3, 3, 2, 1] + s[4, 2, 2, 1] + s[4, 3, 1, 1] + s[4, 3, 2] + s[5, 2,
    1, 1] + s[5, 2, 2] + s[5, 3, 1] + s[6, 2, 1]
sage: ss[3,2,1]*ss[4]
s[4, 3, 2, 1] + s[5, 2, 2, 1] + s[5, 3, 1, 1] + s[5, 3, 2] + s[6, 2,
    1, 1] + s[6, 2, 2] + s[6, 3, 1] + s[7, 2, 1]
sage: ss[3,2]*ss[2]
s[3, 2, 2] + s[3, 3, 1] + s[4, 2, 1] + s[4, 3] + s[5, 2]
```

Again we see that all coefficients are 1. How to describe the partitions that appear in the expansion of $s_\lambda s_{(k)}$? This is given by the full set of Pieri rules. The expansion contains all the partitions that contain λ and k extra boxes, satisfying the following condition: no two extra boxes are on top of one another. We do not prove Pieri rules yet. Notice that the $e \leftrightarrow h$ symmetry implies existence of dual Pieri rules, with respect to multiplication by e_k.

9. Representations of finite groups (Lecture 4)

We recall the main steps in the theory of representations of finite groups. Assume G is a finite group of size $|G|$.

Definition 9.1. A representation of a group G is a vector space V with an operation $g, x \to g(x)$ for $g \in G$, $x \in V$ such that

(i) for a fixed $g \in G$ the map $x \to gx$ is linear,
(ii) for a fixed $x \in V$ we have $1(x) = x$, where $1 \in G$ is the unit element, and $(g_1 g_2)(x) = g_1(g_2(x))$ for any $g_1, g_2 \in G$.

If V is a representation of G and $U \subset V$ is a sub-vector space of V such that $GU = U$, then U is called a *sub-representation*. A sub-representation is said to be *non-trivial* if $U \neq \{0\}$ and $U \neq V$. If U_1, U_2 are representations of G then the direct sum $U_1 \bigoplus U_2$ with the diagonal action of G is also a representation.

Definition 9.2. A representation V is *irreducible* if there is no non-trivial sub-representation $U \subset V$.

We will only consider finite-dimensional representations.

The theory builds on two main lemmas:

Lemma 9.3 (Maschke's theorem)**.** *If V is a representation of G and $U \subset V$ is a sub-representation, then there exists another sub-representation $U' \subset V$ such that $U \cap U' = \{0\}$ and $U + U' = V$, equivalently $V \cong U \oplus U'$.*

Proof. First construct an arbitrary linear operator $\varphi_0 : V \to V$ such that $\varphi_0|_U = \mathrm{Id}_U$ and $\operatorname{Im}\varphi_0 = U$. Then make it commute with the action of G by the following trick. Define $\varphi : V \to V$ by

$$\varphi(x) = \frac{1}{|G|}\sum_{g\in G} g(\varphi_0(g^{-1}x)) \qquad (x \in V).$$

Now we have $\varphi(x) = x$ for all $x \in U$ and $\operatorname{Im}\varphi = U$. Moreover $\varphi(g(x)) = g(\varphi(x))$ for any $g \in G$, $x \in V$. Let $U' = \operatorname{Ker}\varphi$. Then U' is a sub-representation satisfying the required properties. □

Corollary 9.4. *Every finite-dimensional representation can be represented as a direct sum of irreducible representations.*

Definition 9.5. If V and V' are representations of G then the space of *intertwiners* $\mathrm{Hom}_G(V, V')$ is defined as the subspace

$$\mathrm{Hom}_G(V, V') = \{f \in \mathrm{Hom}(V, V') : \ f(g(x)) = g(f(x)) \quad \text{for all } g \in G, x \in V.\}$$

Lemma 9.6 (Schur's lemma)**.** *If V, V' are irreducible representations of G then*

(i) $\mathrm{Hom}_G(V, V') = \{0\}$ *if V and V' are not isomorphic,*

(ii) $\mathrm{Hom}_G(V, V) = \{\lambda \,\mathrm{Id}_V : \lambda \in \mathbb{C}\}$.

Proof. Suppose $\varphi \in \mathrm{Hom}_G(V, V')$, $\varphi \neq 0$. Notice that $\operatorname{Ker}\varphi \subset V$ is a sub-representation. Since $\varphi \neq 0$, $\operatorname{Ker}\varphi \neq V$. Hence $\operatorname{Ker}\varphi = \{0\}$ and φ is injective. Similarly, $\operatorname{Im}\varphi \subset V'$ is a sub-representation. Since $\varphi \neq 0$, $\operatorname{Im}\varphi \neq \{0\}$. Hence $\operatorname{Im}\varphi = V'$, so φ is surjective. So φ must be an isomorphism and we proved (i). Now suppose V and V' are isomorphic, so we can assume $V' = V$. Let $\varphi \in \mathrm{Hom}_G(V, V)$. If φ is not a multiple of the identity matrix, by Jordan normal form theorem there is an eigenvalue $\lambda \in \mathbb{C}$ such that the λ-eigenspace is non-trivial:

$$\operatorname{Ker}(\varphi - \lambda \,\mathrm{Id}_V) = \{x \in V : \ \varphi x = \lambda x\} \neq \{0\}, \neq V.$$

On the other hand, $\operatorname{Ker}(\varphi - \lambda \,\mathrm{Id}_V)$ is clearly a sub-representation. This is contradiction with the assumption that V is irreducible. □

For a representation V and an element $g \in G$ we consider

$$\mathrm{Tr}(g|V) \in \mathbb{C},$$

the trace of g, viewed as an endomorphism of V. For a fixed V the resulting function on G

$$g \to \mathrm{Tr}(g|V)$$

is called the *character* of V.

Recall that the *group algebra* $\mathbb{C}[G]$ is defined as the space of formal linear combinations $\sum_{g\in G} f_g g$, with $f_g \in \mathbb{C}$ for any $g \in G$. The multiplication is defined by expanding the brackets:

$$\left(\sum_{g\in G} f_g g\right)\left(\sum_{g\in G} f'_g g\right) = \sum_{g,h\in G} (f_g f'_g)(gh) = \sum_{g\in G}\left(\sum_{h\in G} f_h f'_{h^{-1}g}\right) g.$$

We will now prove the *first orthogonality relation*:

Proposition 9.7. *Let V, V' be irreducible representations of G. Then we have*

$$\frac{1}{|G|}\sum_{g\in G} \operatorname{Tr}(g^{-1}|V)\operatorname{Tr}(g|V') = \begin{cases} 1 & \text{if } V,\ V' \text{ are isomorphic} \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Compute the dimension of the space $\operatorname{Hom}_G(V, V')$ in two ways. On the one hand, by Schur lemma

$$\dim \operatorname{Hom}_G(V, V') = \begin{cases} 1 & \text{if } V,\ V' \text{ are isomorphic} \\ 0 & \text{otherwise.} \end{cases}$$

On the other hand, consider $\operatorname{Hom}(V, V')$ as a representation of G where for each $g \in G$ and $\varphi \in \operatorname{Hom}(V, V')$

$$(g\varphi)(x) = g(\varphi(g^{-1}x)) \qquad (x \in V).$$

Then $\varphi \in \operatorname{Hom}_G(V, V')$ if and only if $\varphi = g\varphi$ for all $g \in G$. Consider the operator

$$\pi = \frac{1}{|G|}\sum_{g\in G} g$$

acting on $\operatorname{Hom}(V, V')$. We have $\pi^2 = \pi$. Therefore the space $\operatorname{Hom}(V, V')$ is a direct sum of the 0-eigenspace and the 1-eigenspace for π. We can see that $\varphi \in \operatorname{Hom}_G(V, V')$ if and only if $\pi\varphi = \varphi$. Therefore the subspace $\operatorname{Hom}_G(V, V')$ is precisely the 1-eigenspace of π. So we can calculate its dimension by taking the trace

$$\dim \operatorname{Hom}_G(V, V') = \operatorname{Tr}(\pi|\operatorname{Hom}(V, V')) = \frac{1}{|G|}\sum_{g\in G} \operatorname{Tr}(g|\operatorname{Hom}(V, V')).$$

For any $A \in \operatorname{Hom}(V, V)$ and $B \in \operatorname{Hom}(V', V')$ the trace of the operator $\operatorname{Hom}(V, V') \to \operatorname{Hom}(V, V')$ which sends φ to $B\varphi A$ equals $\operatorname{Tr} A \operatorname{Tr} B$. Hence we have

$$\operatorname{Tr}(g|\operatorname{Hom}(V, V')) = \operatorname{Tr}(g|V')\operatorname{Tr}(g^{-1}|V).$$

So we obtain

$$\dim \operatorname{Hom}_G(V, V') = \operatorname{Tr}(\pi|\operatorname{Hom}(V, V')) = \frac{1}{|G|}\sum_{g\in G} \operatorname{Tr}(g^{-1}|V)\operatorname{Tr}(g|V').$$

□

Let $\mathrm{Irr}(G)$ be the set of irreducible representations of G up to isomorphism. We construct now a linear map from the center of the group algebra $Z(\mathbb{C}[G])$ to the vector space of maps from $\mathrm{Irr}(G)$ to $\mathbb{C}$

$$\Phi : Z(\mathbb{C}[G]) \to \mathrm{Maps}(\mathrm{Irr}(G), \mathbb{C}), \qquad \Phi\left(\sum_{g\in G} f_g g\right)(V) = \sum_{g\in G} f_g \operatorname{Tr}(g|V).$$

Of course, we can also construct Φ as the restriction to $Z(\mathbb{C}[G])$ of the map $\mathbb{C}[G] \to \mathrm{Maps}(\mathrm{Irr}(G), \mathbb{C})$, defined by the same formula. The main theorem in the representation theory of finite groups is

Theorem 9.8. *The map $\Phi : Z(\mathbb{C}[G]) \to \mathrm{Maps}(\mathrm{Irr}(G), \mathbb{C})$ is an isomorphism of vector spaces.*

Proof. For any irreducible representation V define $\pi_V \in \mathbb{C}[G]$ by

$$\pi_V = \frac{1}{|G|} \sum_{g\in G} \operatorname{Tr}(g^{-1}|V) g. \tag{9.1}$$

Then by Proposition 9.7, $\Phi(\pi_V)$ has value 1 on V and 0 on any other irreducible representation. This implies that the set of irreducible representations is finite and Φ is surjective.

To prove injecivity of Φ suppose $f \in \operatorname{Ker}\Phi$. On each irreducible representation V the action of f commutes with the action of any $g \in G$. By Schur lemma, it must be given by some scalar operator $\lambda_{f,V} \operatorname{Id}_V$. By the assumption, $\operatorname{Tr}(f|V) = 0$. This gives

$$0 = \operatorname{Tr}(f|V) = \lambda_{f,V} \dim V.$$

Therefore $\lambda_{f,V} = 0$. So we see that f acts as the zero operator on any irreducible representation. Since any representation can be represented as a direct sum of irreducibles, we have that f acts as the zero operator on any representation, in particular on the representation $\mathbb{C}[G]$. Hence $f = 0$. □

An element $f = \sum_{g\in G} f_g g$ belongs to $Z(\mathbb{C}[G])$ if and only if we have $f_g = f_{hgh^{-1}}$ for any $g, h \in G$, in other words, if $g \to f_g$ is a function on the set of conjugacy classes of G. So the theorem above identifies two vector spaces $\mathrm{Maps}(\mathrm{Irr}(G), \mathbb{C})$ and $\mathrm{Maps}(\mathrm{Conj}(G), \mathbb{C})$, where we denote by $\mathrm{Conj}(G)$ the set of conjugacy classes of G. Hence these vector spaces must have the same dimension:

Corollary 9.9. *The number of irreducible representations equals to the number of conjugacy classes for any finite group G.*

10. Examples of groups and irreducible representations (Practice 6)

We talk about abelian groups, cyclic groups, the smallest non-abelian group is S_3. We describe all irreducible representations in the case of a cyclic group. We introduce character tables and show how the character table looks like for a cyclic group. Then we construct all 3 irreducible representations of S_3 and compute the character table

11. Scalar products and the Frobenius character (Lecture 5)

We continue studying the isomorphism

$$\Phi : Z(\mathbb{C}[G]) \to \mathrm{Maps}(\mathrm{Irr}(G), \mathbb{C}), \qquad \Phi\left(\sum_{g\in G} f_g g\right)(V) = \sum_{g\in G} f_g \operatorname{Tr}(g|V).$$

For each $V \in \mathrm{Irr}(G)$ let $\delta_V \in \mathrm{Maps}(\mathrm{Irr}(G), \mathbb{C})$ be defined by

$$\delta_V(V') = \begin{cases} 1 & \text{if } V \cong V', \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, the elements δ_V form a basis of $\mathrm{Maps}(\mathrm{Irr}(G), \mathbb{C})$. So we introduce a symmetric bilinear form on $\mathrm{Maps}(\mathrm{Irr}(G), \mathbb{C})$ in such a way that $\{\delta_V\}$ form an orthonormal basis:

$$(\delta_V, \delta_{V'}) == \begin{cases} 1 & \text{if } V \cong V', \\ 0 & \text{otherwise.} \end{cases}$$

Next we will transport the form to $Z(\mathbb{C}[G])$ via Φ and compute it there:

Proposition 11.1. *Let $(\cdot,\cdot)$ be the symmetric bilinear form on $Z(\mathbb{C}[G])$ defined by*

$$(f, f') = (\Phi(f), \Phi(f')) \qquad (f, f' \in Z(\mathbb{C}[G])).$$

Then we have for any f, f'

$$(f, f') = |G| \operatorname{coeff}_1(f f').$$

Proof. It is enough to verify the statement for $f = \pi_V$, $f' = \pi_{V'}$ for any $V, V' \in \mathrm{Irr}(G)$ (see (9.1)). Recall that $\Phi(\pi_V) = \delta_V$. So we need to check

$$|G| \operatorname{coeff}_1(\pi_V \pi_{V'}) = \begin{cases} 1 & \text{if } V \cong V', \\ 0 & \text{otherwise.} \end{cases}$$

For any other representation V'' we have

$$\pi_V | V'' = \begin{cases} \frac{1}{\dim V} \operatorname{Id}_{V''} & \text{if } V \cong V'', \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, if $V \ncong V'$ then $\pi_V \pi_{V'} = 0$. If $V \cong V'$, we have

$$\pi_V^2 = \frac{1}{\dim V} \pi_V.$$

Directly from (9.1), $\operatorname{coeff}_1(\pi_V) = \frac{1}{|G|} \dim V$. Hence

$$|G| \operatorname{coeff}_1(\pi_V \pi_V) = \frac{|G|}{\dim V} \operatorname{coeff}_1(\pi_V) = 1.$$

□

11.1. **Symmetric group.** Suppose $G = S_n$, the group of permutations of $1, 2, \dots, n$, also called the *symmetric group*. We will connect $Z(\mathbb{C}[S_n])$ with the space of symmetric functions of degree n. The conjugacy classes of S_n correspond to partitions as follows. Let $\sigma \in S_n$ be a permutation. Draw a graph with vertices $1, \dots, n$ and arrows going from i to $\sigma(i)$ for each i. This graph is a disjoint union of cycles whose lengths we record in a list $\lambda_1, \lambda_2, \dots$ such that $\lambda_1 \geq \lambda_2 \cdots$. This is a partition of size n, which we denote by $\mathrm{type}(\sigma)$ and call the *cycle type* of σ. Conjugation of permutations corresponds to relabeling of the vertices of the graph. This idea shows that the map $\sigma \to \mathrm{type}(\sigma)$ gives a bijection from $\mathrm{Conj}(S_n)$ to the set of partitions of size n.

Let $\mathrm{Frob}_0 : Z(\mathbb{C}[S_n]) \to \mathrm{Sym}^n$ be the map sending

$$\sum_{g \in S_n} f_g g \to \sum_{g \in S_n} f_g p_{\mathrm{type}\, g} \in \mathrm{Sym}^n,$$

where for any partition λ we denote $p_\lambda = \prod_i p_{\lambda_i}$.

Then we have for any $V \in \mathrm{Irr}(S_n)$

$$\mathrm{Frob}_0\, \Phi^{-1} \delta_V = \frac{1}{n!} \sum_{\sigma \in S_n} \mathrm{Tr}(\sigma^{-1}|V) p_{\mathrm{type}\, \sigma}.$$

Note that for $\sigma \in S_n$ we have $\mathrm{type}\, \sigma = \mathrm{type}\, \sigma^{-1}$, so σ and σ^{-1} belong to the same conjugacy class and have the same trace.

More generally, for any representation V of S_n we define the *Frobenius character* by

$$\mathrm{Frob}\, V = \frac{1}{n!} \sum_{\sigma \in S_n} \mathrm{Tr}(\sigma|V) p_{\mathrm{type}\, \sigma}. \tag{11.1}$$

This is an important invariant of representations of S_n which we will study. In particular, we will show that the irreducible representations go to the Schur functions.

We define the *Hall scalar product* on Sym^n by transporting the scalar product from $Z(\mathbb{Z}[S_n])$ via Frob_0. Let us compute the scalar product in the power sum basis.

Proposition 11.2. *For any $f, f' \in \mathrm{Sym}^n$ let $(f, f') = (\mathrm{Frob}_0^{-1} f, \mathrm{Frob}_0^{-1} f')$. Then for any $\lambda, \mu \vdash n$ we have*

$$(p_\lambda, p_\mu) = \begin{cases} 0 & \text{if } \lambda \neq \mu \\ z_\lambda & \text{if } \lambda = \mu, \end{cases}$$

where

$$z_\lambda = \prod_{i=1}^{l(\lambda)} \lambda_i \prod_m \#\{j : \lambda_j = m\}!,$$

z_λ is the number of permutations σ' that commute with σ where σ is any permutation of type λ.

Proof. Let $c_\lambda = \{\sigma \in S_n : \mathrm{type}\, \sigma = \lambda\}$. We have

$$\mathrm{Frob}_0^{-1} p_\lambda = \frac{1}{|c_\lambda|} \sum_{\sigma \in c_\lambda} \sigma.$$

Using Proposition 11.1 we obtain

$$(p_\lambda, p_\mu) = \frac{|G|}{|c_\lambda| \cdot |c_\mu|} \operatorname{coeff}_1 \left(\sum_{\sigma \in c_\lambda} \sigma \sum_{\sigma' \in c_\mu} \sigma' \right).$$

It is clear that for $\lambda \neq \mu$ we get 0. If $\lambda = \mu$, we obtain that for each σ there is a unique $\sigma' = \sigma^{-1}$ that contributes to the coefficient of 1. Therefore,

$$(p_\lambda, p_\mu) = \frac{|G|}{|c_\lambda|} = z_\lambda.$$

The last identity follows from the fact that G acts on c_λ transitively by conjugation and the size of the stabilizer of each element $\sigma \in c_\lambda$ is z_λ.

The formula for z_λ follows from the graphical representation of the permutation σ such that $\operatorname{type} \sigma = \lambda$. Namely, z_λ is the number of automorphisms of the oriented graph which consists of oriented cycles of sizes $\lambda_1, \lambda_2, \ldots$. The automorphisms can permute cycles of the same lengths, and each cycle of length λ_i can be rotated in λ_i ways. □

12. Character tables of finite groups (Practice 7)

A finite group in SAGE is represented as a *permutation group*, which means a subgroup of the symmetric group S_n for some n. A subgroup is given by a list of generators. Each generator is a permutation, which is given in the so-called *cycle notation*, that is the list of cycles. Cycles of length 1 are omitted. Thus $(1, 2)$ is the permutation which permutes 1 and 2 and leaves the other elements in place.

We start by implementing S_3 by hand and computing its conjugacy classes and the character table.

```
sage: G=PermutationGroup([ (0,1), (1,2) ])
sage: list(G)
[(), (1,2), (0,1), (0,2,1), (0,1,2), (0,2)]
sage: G.conjugacy_classes()

[Conjugacy class of () in Permutation Group with generators [(1,2),
    (0,1)],
Conjugacy class of (1,2) in Permutation Group with generators [(1,2),
    (0,1)],
Conjugacy class of (0,1,2) in Permutation Group with generators [(1,2)
    , (0,1)]]
sage: G.character_table()
```

```
[ 1 -1  1]
[ 2  0 -1]
[ 1  1  1]
```

SAGE has many groups already built in. For instance, we have the symmetric groups:

```
sage: groups.permutation.Symmetric(3)
Symmetric group of order 3! as a permutation group
sage: groups.permutation.Symmetric(4)
Symmetric group of order 4! as a permutation group
sage: G=groups.permutation.Symmetric(4)
sage: G.character_table()

[ 1 -1  1  1 -1]
[ 3 -1 -1  0  1]
[ 2  0  2 -1  0]
[ 3  1 -1  0 -1]
[ 1  1  1  1  1]
sage: G.conjugacy_classes()

[Conjugacy class of cycle type [1, 1, 1, 1] in Symmetric group of
   order 4! as a permutation group,
Conjugacy class of cycle type [2, 1, 1] in Symmetric group of order 4!
    as a permutation group,
Conjugacy class of cycle type [2, 2] in Symmetric group of order 4! as
    a permutation group,
Conjugacy class of cycle type [3, 1] in Symmetric group of order 4! as
    a permutation group,
Conjugacy class of cycle type [4] in Symmetric group of order 4! as a
   permutation group]
```

To answer a question from the student we calculate the number of conjugacy classes in S_n and then look it up in OEIS.

```
sage: [len(list(groups.permutation.Symmetric(n).conjugacy_classes()))
   for n in range(10)]
[1, 1, 2, 3, 5, 7, 11, 15, 22, 30]
```

There is the Rubik's cube group, but it is too large:

```
sage: G=groups.permutation.RubiksCube()
sage: G.order()
43252003274489856000
sage: G.order().factor()
2^27 * 3^14 * 5^3 * 7^2 * 11
```

As a next example we pick some interesting group, for instance PGL(2, 5), which is the group of 2×2 matrices over $\mathbb{F}_5$, the field of residues modulo 5. Looking at the character table we notice a similarity with S_5. It turns out, the two groups are isomorphic!

```
sage: G=groups.permutation.PGL(2,5)
sage: G
Permutation Group with generators [(3,6,5,4), (1,2,5)(3,4,6)]
sage: G.order()
120
sage: G.character_table()
[ 1  1  1  1  1  1  1]
[ 1 -1  1  1 -1  1 -1]
[ 4  0  0 -1 -2  1  1]
[ 4  0  0 -1  2  1 -1]
[ 5 -1  1  0  1 -1  1]
[ 5  1  1  0 -1 -1 -1]
[ 6  0 -2  1  0  0  0]
sage: G.is_simple()
False
sage: G2=groups.permutation.Symmetric(5)
sage: G2.character_table()
[ 1 -1  1  1 -1 -1  1]
[ 4 -2  0  1  1  0 -1]
[ 5 -1  1 -1 -1  1  0]
[ 6  0 -2  0  0  0  1]
[ 5  1  1 -1  1 -1  0]
[ 4  2  0  1 -1  0 -1]
[ 1  1  1  1  1  1  1]
sage: G2.is_isomorphic(G)
True
```

As a next exercise we recall the character table for S_3 and the row- and column-orthogonality. We compute the Frobenius character and obtain the Schur polynomial:

```
sage: Sym=SymmetricFunctions(QQ)
sage: f=(Sym.power()[1,1,1]-Sym.power()[3])/3
sage: f
1/3*p[1, 1, 1] - 1/3*p[3]
sage: Sym.homogeneous()(f)
h[2, 1] - h[3]
sage: Sym.schur()(f)
s[2, 1]
```

13. Schur functions are Frobenius characters of irreducible representations of symmetric groups (Lecture 6)

At this moment we have constructed the Frobenius character Frob, which associates to a representation of S_n a symmetric function of degree n. We also have a scalar product on Sym^n, explicitly defined in the power sum basis, and we know that Frobenius characters of irreducible representations are orthonormal. The number of irreducible representations is equal to the number of conjugacy classes, which is the number of partitions of size n. Therefore, the characters of irreducible representations must form an orthonormal basis.

First we have to construct at least some representations of S_n. For each $\lambda \vdash n$ we will construct two representations Ind_λ^+ and Ind_λ^-. First we define Ind_λ^+ by

$$\mathrm{Ind}_\lambda^+ = \mathbb{C}[S_n/S_\lambda],$$

where $S_\lambda = S_{\lambda_1} \times S_{\lambda_2} \times \cdots \times S_{\lambda_{l(\lambda)}} \subset S_n$ is the subgroup which preserves the decomposition of the sequence $1, 2, \ldots, n$ into consecutive blocks of lengths $\lambda_1, \lambda_2, \ldots$.

More explicitly, let a_λ be the sequence of length n

$$a_\lambda = (1, \ldots, 1, 2, \ldots, 2, \ldots),$$

where each i appears λ_i times. The group S_n acts on sequences of length n by permuting the entries, and the stabilizer of a_λ is S_λ:

$$\sigma(a_\lambda) = a_\lambda \quad \Leftrightarrow \quad \sigma \in S_\lambda.$$

Thus S_n/S_λ can be identified with the set of all sequences that can obtained from a_λ by permutations.

The representation $\mathbb{C}[S_n/S_\lambda]$ is defined as the vector space with basis $\{e_a | a \in S_n/S_\lambda\}$, and the group S_n acts on S_n/S_λ and therefore on the basis elements:

$$\sigma(e_a) = e_{\sigma(a)} \quad (a \in S_n).$$

The representaton Ind_λ^- is defined by

$$\mathrm{Ind}_\lambda^- = \mathrm{Ind}_{\lambda'}^+ \otimes \mathrm{sign},$$

or more explicitly as a vector space with basis $\{e_a | a \in S_n/S_{\lambda'}\}$, but the action is "twisted":

$$\sigma(e_a) = \mathrm{sign}(\sigma) e_{\sigma(a)} \quad (a \in S_n/S_\lambda).$$

We will prove

Theorem 13.1. *(i) For each $\lambda \vdash n$ there exists a unique irreducible representation V_λ of S_n which appears in both Ind_λ^+ and Ind_λ^-. Moreover, V_λ appears in both Ind_λ^+ and Ind_λ^- with multiplicity 1.*

(ii) If $\lambda > \mu$ in the lexicographic order, then the representations Ind_λ^+ and Ind_μ^- do not have any irreducible sub-representations in common.

The proof occupies the rest of the section. Our strategy is to reduce the statements to statements about scalar products. The first statement is equivalent to the following:

$$(\mathrm{Frob}(\mathrm{Ind}_\lambda^+), \mathrm{Frob}(\mathrm{Ind}_\lambda^-)) = 1. \tag{13.1}$$

The second statement is equivalent to

$$(\mathrm{Frob}(\mathrm{Ind}_\mu^+), \mathrm{Frob}(\mathrm{Ind}_\lambda^-)) = 0 \quad (\mu > \lambda). \tag{13.2}$$

So we first compute $\mathrm{Frob}(\mathrm{Ind}_\lambda^+), \mathrm{Frob}(\mathrm{Ind}_\lambda^-)$ in terms of well-known symmetric functions. Note that for $\lambda = (n)$ the representation $\mathrm{Ind}_{(n)}^+$ is the trivial representation.

Proposition 13.2. *The Frobenius character of the trivial representation of S_n is h_n. We have*

$$\sum_{\lambda \vdash n} \frac{p_\lambda}{z_\lambda} = h_n.$$

Proof. From the definition (11.1), all the traces being 1 we obtain

$$\mathrm{Frob}(\mathrm{Ind}_{(n)}^+) = \frac{1}{n!} \sum_{\sigma \in S_n} p_{\mathrm{type}(\sigma)}.$$

For each λ the number of permutations of type λ is given by $|c_\lambda| = \frac{n!}{z_\lambda}$. So we have

$$\mathrm{Frob}(\mathrm{Ind}_{(n)}^+) = \sum_{\lambda \vdash n} \frac{p_\lambda}{z_\lambda}.$$

Consider the summand $\frac{p_\lambda}{z_\lambda}$ for a fixed λ. Let $n_m = \#\{j : \lambda_j = m\}$. Then we have

$$\frac{p_\lambda}{z_\lambda} = \prod_{m : n_m \neq 0} \frac{1}{n_m!} \left(\frac{p_m}{m}\right)^{n_m}.$$

Now for the generating function we can write

$$1+\sum_{n=1}^{\infty} t^n \operatorname{Frob}(\operatorname{Ind}_{(n)}^{+}) = \sum_{\vec{n}} \prod_{m:n_m\neq 0} \frac{t^{mn_m}}{n_m!}\left(\frac{p_m}{m}\right)^{n_m}, \tag{13.3}$$

where the summation goes over the set of infinite vectors $\vec{n}=(n_1,n_2,\ldots)$ of non-negative integers with only finitely positive entries. Indeed, each such vector corresponds to a unique partition λ with $n_m=\#\{j:\lambda_j=m\}$.

Next we recognize that (13.3) is the expansion of the product

$$\prod_{m=1}^{\infty}\sum_{n=0}^{\infty}\frac{t^{mn}}{n!}\left(\frac{p_m}{m}\right)^n = \prod_{m=1}^{\infty}\exp(t^m p_m/m) = \exp\left(\sum_{m=1}^{\infty}\frac{t^m p_m}{m}\right),$$

which equals $\sum_{m=0}^{\infty} h_m t^m$ by Proposition 5.1. □

The case of a general induced representation is not far:

Proposition 13.3. *The Frobenius character of* $\operatorname{Ind}_{\lambda}^{+}$ *is*

$$\operatorname{Frob}(\operatorname{Ind}_{\lambda}^{+}) = h_\lambda = \prod_i h_{\lambda_i}.$$

Proof. Since each $\sigma\in S_n$ acting on $\operatorname{Ind}_{\lambda}^{+}$ permutes the basis elements, its trace equals to the number of basis elements that are fixed by σ. So we have

$$\operatorname{Frob}(\operatorname{Ind}_{\lambda}^{+}) = \frac{1}{n!}\sum_{\sigma\in S_n} p_{\operatorname{type}(\sigma)} \operatorname{Tr}(\sigma|\operatorname{Ind}_{\lambda}^{+}) = \frac{1}{n!}\sum_{\sigma\in S_n,\, a\in S_n/S_\lambda,\, \sigma a=a} p_{\operatorname{type}(\sigma)}.$$

The contribution of each a to the sum is the same, so we can replace all of them by the class of the identity element. Then the condition $\sigma a=a$ translates to $\sigma\in S_\lambda$:

$$\operatorname{Frob}(\operatorname{Ind}_{\lambda}^{+}) = \frac{|S_n/S_\lambda|}{n!}\sum_{\sigma\in S_\lambda} p_{\operatorname{type}\sigma}.$$

We have $|S_n/S_\lambda| = \frac{n!}{\prod_i \lambda_i!}$. Each $\sigma\in S_\lambda$ corresponds to a sequence of permutations $\sigma_i\in S_{\lambda_i}$ $(i=1,\ldots,l(\lambda))$, and the cycle type of σ is obtained by putting together the cycle types of σ_i. Thus the formula factors:

$$\operatorname{Frob}(\operatorname{Ind}_{\lambda}^{+}) = \frac{1}{\prod_i \lambda_i!}\sum_{\sigma_1\in S_{\lambda_1},\,\sigma_2\in S_{\lambda_2},\ldots} p_{\operatorname{type}\sigma_1}p_{\operatorname{type}\sigma_2}\ldots = \prod_{i=1}^{l(\lambda)}\frac{1}{\lambda_i!}\sum_{\sigma\in S_{\lambda_i}} p_{\operatorname{type}\sigma}.$$

By Proposition 13.2, the value of the i-th factor equals h_{λ_i}. □

Tensoring by sign has the effect of replacing each h_i by e_i:

Proposition 13.4. *The Frobenius character of* $\operatorname{Ind}_{\lambda}^{-}$ *is*

$$\operatorname{Frob}(\operatorname{Ind}_{\lambda}^{-}) = e_{\lambda'}.$$

Proof. More generally, suppose we know the Frobenius character of some S_n-representation V and we want to compute the Frobenius character of $V \otimes \mathrm{sign}$. For each $\sigma \in S_n$ the action of σ on $V \otimes \mathrm{sign}$ is the same as the action on V, but multiplied by the sign of σ. Therefore,

$$\mathrm{Tr}(\sigma | V \otimes \mathrm{sign}) = \mathrm{Tr}(\sigma | V)\,\mathrm{sign}(\sigma).$$

The sign of σ can be related to the type as follows. If $\mathrm{type}(\sigma) = \lambda$, then

$$\mathrm{sign}(\sigma) = (-1)^{\sum_{i=1}^{l(\lambda)} \lambda_i - 1}.$$

So let us define the operator $\omega : \mathrm{Sym} \to \mathrm{Sym}$ by sending any polynomial in $p_1, p_2, \ldots$ to the same polynomial evaluated at $\{(-1)^{k-1} p_k\}_{k>0}$. By the construction,

$$\mathrm{Frob}(V \otimes \sigma) = \omega\,\mathrm{Frob}(V).$$

Applying Proposition 5.1, we obtain $\omega h_k = e_k$ and more generally $\omega h_\lambda = e_\lambda$. Thus

$$\mathrm{Frob}(\mathrm{Ind}_\lambda^+ \otimes \mathrm{sign}) = e_\lambda,$$

and the statement follows. □

To complete the proof of (13.1) and (13.2) we will need the following:

Proposition 13.5. *For any partitions λ, μ of same size,*

$$(h_\lambda, m_\mu) = \begin{cases} 1 & \text{if } \lambda = \mu, \\ 0 & \text{otherwise.} \end{cases}$$

We will postpone its proof until the next lecture. Assuming Proposition 13.5, the statements (13.1) and (13.2) immediately follow from the expansion

$$\mathrm{Frob}(\mathrm{Ind}_\lambda^-) = e_{\lambda'} = m_\lambda + (\text{terms } m_\mu \text{ with } \mu < \lambda),$$

which is precisely what we have shown during the proof of Theorem 3.3.

Note that also we do not know the character of V_λ yet, we know that it satisfies the following property:

$$\mathrm{Frob}(V_\lambda) = e_{\lambda'} = m_\lambda + (\text{terms } m_\mu \text{ with } \mu < \lambda). \tag{13.4}$$

Proof. This holds because V_λ appears with multiplicity one in Ind_λ^+, and it does not appear in Ind_μ^+ for $\mu > \lambda$. □

The property (13.4) and the orthogonality uniquely determine $\mathrm{Frob}(V_\lambda)$ by the Gram-Schmidt orthogonalization argument. By Proposition 7.3, Schur polynomials satisfy (13.4). So, in order to show that $\mathrm{Frob}(V_\lambda) = s_\lambda$, it is enough to verify that Schur polynomials are orthogonal. This, together with Proposition 13.5 will be shown in the next lecture.

14. Examples for the induced representation (Practice 8)

We explicitly describe the induced representation $\mathrm{Ind}_{2,2}^{+}$. It is six-dimensional, we compute its character and the Frobenius character, and check that it is h_2^2. We have (by Pieri rules)

$$h_2^2 = s_{2,2} + s_{3,1} + s_4.$$

It is clear how to see the trivial representation as a sub-representation of $\mathrm{Ind}_{2,2}^{+}$. The remaining 5-dimensional piece should therefore split into two representations as $2+3$ corresponding to $s_{2,2}$ and $s_{3,1}$.

15. Cauchy product formula (Lecture 7)

In order to prove Proposition 13.5 and orthogonality of Schur polynomials, we will use a general technique of *reproducing kernels*.

Suppose X is a vector space of dimension m endowed with a symmetric bilinear form $(\cdot,\cdot)$.

Definition 15.1. Two bases $a_1, a_2, \ldots, a_m$ and $b_1, b_2, \ldots, b_m$ are *dual* if

$$(a_i, b_j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

The basic property is

Proposition 15.2. *For each basis $a_1, \ldots, a_m$ there is at most one dual basis $b_1, \ldots, b_m$. The form is non-degenerate if and only if there exists a pair of dual basis. If the form is non-degenerate, than any basis has a unique dual.*

Suppose the form is non-degenerate and choose any pair of dual bases $a_1, a_2, \ldots, a_m$ and $b_1, b_2, \ldots, b_m$. Define the *reproducing kernel* by

$$K = \sum_{i=1}^{m} a_i \otimes b_i. \tag{15.1}$$

Lemma 15.3. *The tensor K depends only on the bilinear form and does not depend on the choice of dual bases. Conversely, if for two sequences of vectors $a_1, \ldots, a_m$, $b_1, \ldots, b_m$ we have* (15.1)*, then they are dual bases.*

Proof. The main idea of the proof is to pick a third basis $c_1, \ldots, c_n$ and show that for arbitrary dual pair $a_1, \ldots, a_m$, $b_1, \ldots, b_m$, the matrix of K is the inverse of the matrix of the bilinear form. □

We will use the following idea. Suppose X is a finite dimensional subspace of the space of polynomials in variables $x_1, \ldots, x_n$. Then $X \otimes X$ can be identified with a subspace of the space of polynomials in $x_1, \ldots, x_n, y_1, \ldots, y_n$ by identifying

$$x_1^{q_1} x_2^{q_2} \ldots x_n^{q_n} \otimes x_1^{r_1} x_2^{r_2} \ldots x_n^{r_n} \in \mathbb{C}[x_1, \ldots, x_n] \otimes \mathbb{C}[x_1, \ldots, x_n]$$

with

$$x_1^{q_1}x_2^{q_2}\dots x_n^{q_n}y_1^{r_1}y_2^{r_2}\dots y_n^{r_n}\in\mathbb{C}[x_1,\dots,x_n,y_1,\dots,y_n].$$

Now let us consider the case $X=\mathrm{Sym}^d$ with the Hall scalar product. Let $n\geq d$. Then we have $\mathrm{Sym}^d_n\cong\mathrm{Sym}^d$ and we can view the reproducing kernel of the Hall scalar product on Sym^d as a function in variables $x_1,\dots,x_n,y_1,\dots,y_n$, which we denote by K_d. By Proposition 11.2 we have

$$K_d(x_1,\dots,x_n,y_1,\dots,y_n)=\sum_{\lambda\vdash d}\frac{p_\lambda(x_1,\dots,x_n)p_\lambda(y_1,\dots,y_n)}{z_\lambda}.$$

This function makes sense also if $n<d$. Consider the generating function

$$K(x_1,\dots,x_n,y_1,\dots,y_n;t)=1+\sum_{n=1}^{\infty}K_d(x_1,\dots,x_n,y_1,\dots,y_n)t^d.$$

Proposition 15.4 (Cauchy product formula)**.** *We have*

$$K(x_1,\dots,x_n,y_1,\dots,y_n;t)=\prod_{i,j=1}^{n}\frac{1}{1-x_iy_jt}.$$

Proof. Notice that

$$p_\lambda(x_1,\dots,x_n)p_\lambda(y_1,\dots,y_n)=p_\lambda(x_1y_1,x_1y_2,\dots,x_ny_n),$$

where we have n^2 arguments x_iy_j on the right hand side. Then we can use Proposition 13.2 to see that

$$K_d(x_1,\dots,x_n,y_1,\dots,y_n)=h_d(x_1y_1,x_1y_2,\dots,x_ny_n).$$

Finally, use (5.1) to write $\sum_d t^dh_d$ in the product form. □

Now we are ready to prove Proposition 13.5.

Proof of Proposition 13.5. Using Lemma 15.3, it is enough to show that

$$K(x_1,\dots,x_n,y_1,\dots,y_n;t)=1+\sum_{d=1}^{\infty}t^d\sum_{\lambda\vdash d}h_\lambda(x_1,\dots,x_n)m_\lambda(y_1,\dots,y_n). \tag{15.2}$$

By the product formula and (5.1), we have

$$K(x_1,\dots,x_n,y_1,\dots,y_n;t)=\prod_{j=1}^{n}\left(1+\sum_{k=1}^{\infty}(y_jt)^kh_k(x_1,\dots,x_n)\right).$$

Opening the parenthesis we see that the coefficient of $y_1^{a_1}y_2^{a_2}\cdots y_n^{a_n}t^{\sum_i a_i}$ is precisely

$$\prod_i h_{a_i}(x_1,\dots,x_n)=h_\lambda(x_1,\dots,x_n),$$

where λ is a partition obtained by sorting the sequence $a_1,\dots,a_n$ and removing zeros. So (15.2) is true. □

We prove orthogonality of Schur functions in a similar way

Proposition 15.5. *For any two partitions λ, μ of same size, we have*

$$(s_\lambda, s_\mu) = \begin{cases} 1 & \text{if } \lambda = \mu, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Similarly to the proof of Proposition 13.5, we need to show

$$K(x_1, \dots, x_n, y_1, \dots, y_n; t) = \sum_{d=0}^{\infty} t^d \sum_{\lambda \vdash d} s_\lambda(x_1, \dots, x_n) s_\lambda(y_1, \dots, y_n). \tag{15.3}$$

Denote the right hand side of (15.3) by RHS. Let $M(x)$ be the infinite matrix of height n whose i, j-entry is x_i^{j-1}:

$$M(x) = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & \cdots \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots \\ \vdots & \vdots & \vdots & \cdots & \\ 1 & x_n & x_n^2 & x_n^3 & \cdots \end{pmatrix}$$

Similarly, let $M(ty)$ be the infinite matrix whose i, j-entry is $(ty_i)^{j-1}$. Then we have (see Section 7)

$$\text{RHS } \Delta(x)\Delta(y)t^{\binom{n}{2}} = \sum_{0<a_1<a_2<\dots<a_n} \det M_a(x) \det M_a(ty),$$

where $M_a(x)$ is the sub-matrix of $M(x)$ formed by columns $a_1, \dots, a_n$, and similarly for $M_a(ty)$. By a linear algebra theorem, the right hand side can be written as the determinant of the product matrix $M(x)M(ty)^{\text{tr}}$. Although the product of infinite matrices is not well-defined, if we want to compute the coefficient of t^d for some fixed value of d, we need to consider only finitely many elements in these matrices. So we can replace the infinite matrices by the truncated $n \times N(d)$ matrices where $N(d)$ is big enough. The product matrix has entries $\frac{1}{1-x_i y_j t}$. Hence

$$\text{RHS } \Delta(x)\Delta(y)t^{\binom{n}{2}} = \det\left(\frac{1}{1 - x_i y_j t}\right)_{i,j=1}^{n}.$$

Now we have an identity of rational functions that we need to prove. We can replace y_i by y_i/t to get rid of y. So we are reduced to proving the following algebraic identity, which is left as an exercise:

$$\det\left(\frac{1}{1 - x_i y_j}\right)_{i,j=1}^{n} = \frac{\Delta(x)\Delta(y)}{\prod_{i,j=1}^{n}(1 - x_i y_j)}.$$

□

From the discussion in the end of Section 13, we conclude

Theorem 15.6. *For any partition λ, the Frobenius character of the irreducible representation V_λ is s_λ.*

16. Schur-Weyl duality (Lectures 8-9)

17. Practice 9

Decompositions of $E \otimes E$, $E \otimes E \otimes E$.

18. Practice 10

Some more discussion about the proof of Schur-Weyl duality and how to describe GL_n representations explicitly.

19. Bruhat decomposition of $GL_n(\mathbb{C})$ (Lecture 10)

We construct the Bruhat decomposition of $GL_n(\mathbb{C})$ and explain how it is related to understanding relative position of flags.

20. Algebraic manifold structure on the Grassmannian (Lecture 11)

We give a definition of an algebraic variety, similar to the definition of a smooth or complex manifold. We cover Grassmannian by charts and show that it is an algebraic manifold of dimension $k(n-k)$. We define Zariski open, Zariski closed sets and algebraic dimension.

21. Schubert cells (Lecture 12)

We construct Schubert cell decomposition and show that cells are Zariski locally closed. We also compute dimensions of cells.

22. Practice 11

We compute Schubert cells of $\mathrm{Gr}_2(\mathbb{C}^4)$. We show that some intersections of cells of dimension 2 are zero.

23. Poincaré duality and basic properties of intersections (Lecture 13)

Half of the lecture is used to explain the basic ideas behind homology, cohomology and Poincaré duality. Then we prove that for $|\lambda|+|\mu| = k(n-k)$ we have (c_λ, c_μ) is 1 if μ is the complementary partition to λ ($\mu = \bar{\lambda}$) and 0 otherwise. We also prove that the cells can intersect only if $\mu \subset \bar{\lambda}$.

24. Intersecting Schubert cells in SAGE (Practice 12)

```
sage: FF=QQ
sage: K=3
sage: N=6
sage: def jumps(lam):
....:     lam = list(lam)
```

```
....:         while len(lam)<K:
....:             lam.append(0)
....:         res = [N-1-(lam[i]+K-i-1) for i in range(K)]
....:         return res
....:
sage: jumps([])
[3, 4, 5]
sage: jumps([1])
[2, 4, 5]
sage: jumps([2])
[1, 4, 5]
sage: jumps([1,1])
[2, 3, 5]
sage: def generic_matrix(lam):
....:         a = jumps(lam)
....:         stars = []
....:         for i in range(K):
....:             for j in range(N):
....:                 if j<a[i] and j not in a:
....:                     stars.append((i,j))
....:         return stars
....:
sage: generic_matrix([])
[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2,
   2)]
sage: generic_matrix([1])
[(0, 0), (0, 1), (1, 0), (1, 1), (1, 3), (2, 0), (2, 1), (2, 3)]
sage: generic_matrix([2])
[(0, 0), (1, 0), (1, 2), (1, 3), (2, 0), (2, 2), (2, 3)]
sage: generic_matrix([1,1])
[(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1), (2, 4)]
sage: generic_matrix([2,1])
[(0, 0), (1, 0), (1, 2), (2, 0), (2, 2), (2, 4)]
sage: jumps([2,1])
[1, 3, 5]
```

```
sage: def generic_matrix(lam):
....:     a = jumps(lam)
....:     stars = []
....:     for i in range(K):
....:         for j in range(N):
....:             if j<a[i] and j not in a:
....:                 stars.append((i,j))
....:     names = ['m%d%d'%(stars[i][0], stars[i][1]) for i in range(
   len(stars))]
....:     return names
....:
sage: generic_matrix([2,1])
['m00', 'm10', 'm12', 'm20', 'm22', 'm24']
sage: def generic_matrix(lam):
....:     a = jumps(lam)
....:     stars = []
....:     for i in range(K):
....:         for j in range(N):
....:             if j<a[i] and j not in a:
....:                 stars.append((i,j))
....:     names = ['m%d%d'%(stars[i][0], stars[i][1]) for i in range(
   len(stars))]
....:     R = PolynomialRing(FF, names=names)
....:     M = matrix(R, K, N)
....:     for i in range(K):
....:         M[i,a[i]]=1
....:     for i in range(len(stars)):
....:         M[stars[i][0], stars[i][1]] = R.gen(i)
....:     return M, R
....:
sage: generic_matrix([2,1])
(
[m00   1   0   0   0   0]
[m10   0 m12   1   0   0]
```

```
[m20   0 m22   0 m24   1], Multivariate Polynomial Ring in m00, m10,
   m12, m20, m22, m24 over Rational Field
)
sage: def mat_to_equations(M, mu):
....:     res = []
....:     mu=list(mu)
....:     while len(mu)<K:
....:         mu.append(0)
....:     for i in range(K):
....:         size = K-i
....:         numcols = mu[i]+K-i-1
....:         for ss1 in Subsets(range(K), size):
....:             for ss2 in Subsets(range(numcols), size):
....:                 det=M.matrix_from_rows_and_columns(list(ss1),
   list(ss2)).det()
....:                 res.append(det)
....:     return res
....:
sage: mat_to_equations(M, [1])
[-m02*m11*m20 + m01*m12*m20 + m02*m10*m21 - m00*m12*m21 - m01*m10*m22
   + m00*m11*m22]
sage: mat_to_equations(M, [1,1])
[-m02*m11*m20 + m01*m12*m20 + m02*m10*m21 - m00*m12*m21 - m01*m10*m22
   + m00*m11*m22,
-m01*m10 + m00*m11,
-m01*m20 + m00*m21,
-m11*m20 + m10*m21]
sage: mat_to_equations(M, [3])
[-m02*m11*m20 + m01*m12*m20 + m02*m10*m21 - m00*m12*m21 - m01*m10*m22
   + m00*m11*m22,
-m11*m20 + m10*m21,
m01*m20 - m00*m21,
-m12*m20 + m10*m22,
m02*m20 - m00*m22,
m20,
```

```
-m12*m21 + m11*m22,
m02*m21 - m01*m22,
m21,
m22]
sage: mat_to_equations(M, [1,1,1])
[-m02*m11*m20 + m01*m12*m20 + m02*m10*m21 - m00*m12*m21 - m01*m10*m22
    + m00*m11*m22,
-m01*m10 + m00*m11,
-m01*m20 + m00*m21,
-m11*m20 + m10*m21,
m00,
m10,
m20]
sage: mat_to_equations(M, [1])
[-m02*m11*m20 + m01*m12*m20 + m02*m10*m21 - m00*m12*m21 - m01*m10*m22
    + m00*m11*m22]
sage: M,R=generic_matrix([2,1])
sage: M
[m00   1   0   0   0   0]
[m10   0 m12   1   0   0]
[m20   0 m22   0 m24   1]
sage: mat_to_equations(M, [1])
[m12*m20 - m10*m22]
sage: J=R.ideal(mat_to_equations(M, [1]))
sage: J.dimension()
5
sage: R
Multivariate Polynomial Ring in m00, m10, m12, m20, m22, m24 over
    Rational Field
sage: M,R=generic_matrix([2,2])
sage: J=R.ideal(mat_to_equations(M, [2,2]))
sage: J.dimension()
-1
sage: M,R=generic_matrix([2,1])
sage: J=R.ideal(mat_to_equations(M, [3,2,1]))
```

```
sage: J.dimension()
0
sage: J.groebner_basis()
[m00, m10, m12, m20, m22, m24]
sage: J.variety(QQ)
[{m24: 0, m22: 0, m20: 0, m12: 0, m10: 0, m00: 0}]
sage: J.variety(QQbar)
[{m00: 0, m20: 0, m10: 0, m22: 0, m12: 0, m24: 0}]
sage: J.variety(CC)
[{m24: 0.000000000000000, m22: 0.000000000000000, m20:
   0.000000000000000, m12: 0.000000000000000, m10: 0.000000000000000,
   m00: 0.000000000000000}]
sage: M,R=generic_matrix([1])
sage: J=R.ideal(mat_to_equations(M, [1]))
sage: J.groebner_basis()
[m11*m20 - m10*m21]
sage: R
Multivariate Polynomial Ring in m00, m01, m10, m11, m13, m20, m21, m23
    over Rational Field
sage: M,R=generic_matrix([2])
sage: J=R.ideal(mat_to_equations(M, [3,1,1]))
sage: J.groebner_basis()
[m00, m10, m20, m22, m23]
sage: R
Multivariate Polynomial Ring in m00, m10, m12, m13, m20, m22, m23 over
    Rational Field
```

25. Pieri rules and the cohomology ring of the Grassmannian (Lecture 14)

We compute triple intersections $c_\lambda \cdot c_\mu \cdot c_{(r)}$ and match it with Pieri rules, thus proving that the map sending s_λ to c_λ is a ring homomorphism.

26. Geometric interpret (Lecture 15)