Regular Constructions

Contents

1 Recall

Definition. The ordinary generating function (OGF) of a sequence (A_n) is the formal power series

$$A(z) = \sum_{n=0}^{\infty} A_n z^n.$$

We extend this to say that the ogf of a class $\mathcal{A}$ is the generating function of its counting sequence $A_n = \text{cardinality}(\mathcal{A}_n)$. Equivalently, we can write the OGF as

$$A(z) = \sum_{\alpha\in\mathcal{A}} z^{|\alpha|}.$$

In this context we say that the variable z marks the size of the underlying objects.

Two formal power series facts that we shall use: Multiplication rule:

$$\left(\sum A_n z^n\right)\left(\sum B_n z^n\right) = \sum_n \left(\sum_{k=0}^{n} A_k B_{n-k}\right) z^n$$

Quasi-inverse: If $A(z)$ is such that $A_0 = 0$, then

$$1 + A(z) + A(z)^2 + A(z)^3 + A(z)^4 + \cdots = \frac{1}{1-A(z)}.$$

2 A calculus for combinatorial classes

2.1 The basic idea: Union $\equiv$ sum

Our approach builds up a new class from existing classes, and translates counting information at the same time. A simple example of this principle is as follows. Imagine class $\mathcal{A}$ and class $\mathcal{B}$ have empty intersection, and that we know everything about them. What can we say about $\mathcal{C}$, if it is the union of these two classes:

$$\mathcal{C} = \mathcal{A}\cup\mathcal{B}?$$

In particular, $\mathcal{C}_n = \mathcal{A}_n \cup \mathcal{B}_n$, and hence $C_n = A_n + B_n$. This only works if the union is disjoint, i.e. $\mathcal{A}\cap\mathcal{B} = \emptyset$, but when this does hold, no further information is needed about $\mathcal{A}$ or $\mathcal{B}$. We translate the structural information about $\mathcal{C}$ into counting information about $\mathcal{C}$. Similarly, to describe a random generation scheme for $\mathcal{C}$, it is sufficient to have one for $\mathcal{A}$ and $\mathcal{B}$, and a way to choose which one to have an element from. But we are getting ahead of ourselves.

2.2 Admissible constructions

Now we build a toolbox of standard constructions. In each case we want the property that the counting information is a function of the construction and the counting sequences of the inputs, but nothing else. We can define a small set of constructions from which we can derive a remarkable number of objects of interest.

Our toolbox of combinatorial constructions will include the following kinds of operations:

- union;
- cartesion product;
- sequence;
- set and multiset;
- cycle;
- pointing and substitution.

These "admissable constructions" will translate directly to simple (and not so simple) operations on the generating functions.

First, a technical definition which formalizes the above intuition.

Definition. Let Φ be an m-ary construction that associates to any collection of classes $\mathcal{B}^{(1)}, ...\mathcal{B}^{(m)}$ a new class $\mathcal{A} = \Phi[\mathcal{B}^{(1)}, ...\mathcal{B}^{(m)}]$. The construction Φ is admissible iff the counting sequence (A_n) only depends on the counting sequences $(B_n^{(1)}, \dots, B_n^{(m)})$.

Given an admissible construction Φ, there exists a well-defined operator Ψ acting on the corresponding ordinary generating functions: $A(z) = \Psi[B^{(1)}(z), ..., B^{(m)}(z)]$.

Our constructions will be mechanical, but they will have a natural bijection with the desired class.

2.2.1 Bijective combinatorics

Enumeration is also important because it may help us recognize two classes that are essentially the same.

Definition. Two classes $\mathcal{A}, \mathcal{B}$ are said to be combinatorially isomorphic, written $\mathcal{A} \cong \mathcal{B}$ iff their counting sequences are the same. Note that $\mathcal{A} \cong \mathcal{B}$ iff there is a size-preserving bijection between $\mathcal{A}$ and $\mathcal{B}$.

There are many beautiful results in combinatorics resulting from the construction of bijections between seemingly disparate classes of objects. This "bijective" school of combinatorics is championed by the french.

Us, we will describe classes that will be in very natural bijections with the objects we are truly interested in.

2.3 Cartesian Product

The easiest of these is the construction based on the cartesian product

Definition. The cartesian product construction applied to two classes $\mathcal{B}, \mathcal{C}$ forms ordered pairs,

$$\mathcal{A} = \mathcal{B} \times \mathcal{C} \text{ iff}$$
$$\mathcal{A} = \{\alpha = (\beta, \gamma) | \beta \in \mathcal{B}, \gamma \in \mathcal{C}\}$$

where the size of a pair is additive, namely

$$|\alpha|_{\mathcal{A}} = |\beta|_{\mathcal{B}} + |\gamma|_{\mathcal{C}}.$$

By considering all possible pairs we see that the counting sequences satisfy

$$A_n = \sum_{k=0}^{n} B_k C_{n-k}$$

which is exactly the product of the generating functions

$$A(z) = B(z) \cdot C(z).$$

Since the counting sequence (A_n) depends only the sequences (B_n) and (C_n) (and not any other information) we see that the cartesian product is admissible and it translates as a product of OGFs.

Exercise. Verify that the set $\mathcal{A}$ defined as $\mathcal{A} = \mathcal{B} \times \mathcal{C}$ is a well-defined as a combinatorial class. Understand why cartesian product is an admissible operator.

Example. Consider the class of binary strings with a break, or bookmark somewhere in the string, possibly at the beginning or the end. Length is still the number of 1s plus the number of 0s. For example $100101^{\vee}100100$. We can model this by a cartesian product of binary strings: $100101^{\vee}100100 \equiv (100101, 100100)$. Here the length is additive. Thus, this class is equivalent to $\mathcal{V} \equiv \mathcal{W} \times \mathcal{W}$. We count the number of elements of length n in three different ways.

1. Combinatorial argument First, we can see that for any usual binary string of length n, there are $n+1$ ways to insert the bookmark, including the beginning and the end, hence we expect $2^n(n+1)$ elements of length n.

2. Recurrence on coefficients From the above formula, we have

$$\mathcal{V} \equiv \mathcal{W} \times \mathcal{W} \implies V_n = \sum_{k=0}^{n} W_k W_{n-k} = \sum_{k=0}^{n} 2^k 2^{n-k} = \sum_{k=0}^{n} 2^n = (n+1)2^n,$$

as predicted.

3. Generating functions By the formula, the fact that $W(z) = \frac{1}{1-2z}$ and the identity $[z^n]\frac{1}{(1-mz)^r} = m^r\binom{r+n-1}{n}$, we have that

$$V_n = [z^n]W(z)^2 = [z^n]\frac{1}{(1-2z)^2} = 2^n\binom{2+n-1}{n} = 2^n\frac{n+1}{n} = (n+1)2^n.$$

Exercise. Which one of these methods is easiest if we have 2 bookmarks (ordered)? 200 bookmarks? Which one is most automatic? What if we have 2 or 3.

2.4 Revisit Unions

Similarly we have seen that the *disjoint* union is nice

Definition. Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be combinatorial classes such that

$$A = B \cup C \qquad\qquad B \cap C = \varnothing$$

with size defined in a compatible way; if $\omega \in A$ then

$$|\omega|_{\mathcal{A}} = \begin{cases} |\omega|_{\mathcal{B}} & \omega \in \mathcal{B} \\ |\omega|_{\mathcal{C}} & \omega \in \mathcal{C} \end{cases}.$$

Then we clearly have

$$A_n = B_n + C_n$$
$$A(z) = B(z) + C(z)$$

Hence the union of disjoint sets is admissible and it translates as a sum of ogf.

If we see that it is a problem of counting elements of a union of disjoint sets or a cartesian product then we do not have to write down explicit recurrence relations as an intermediate stages, rather we can directly write down the relationship between the generating functions. This is the spirit of the symbolic method for combinatorial enumeration.

2.5 Elemental classes

So let us prove a theorem demonstrating some of the basic constructions we can use to build up more complicated objects.

Definition (Neutral + atomic classes)**.** Let $\mathcal{E}$ denote the neutral class that consists of a single object of size zero. This is typically denoted ϵ. It typically denotes (say) the single empty word, the single graph of zero vertices, or single graph of zero edges etc. etc.

Let $\mathcal{Z}$ denote the atomic class that consists of a single object of size 1 called an atom. This is typically something like a single generic vertex or a single generic edge or a single generic letter. Distinct copies of the atomic class should be denoted with a subscript. Eg for our binary words we could have $\mathcal{Z}_0 = \{0\}$, $\mathcal{Z}_1 = \{1\}$ etc.

Clearly $E(z) = 1$ and $Z(z) = z$.

A key way in which we use ϵ: elements of $\mathcal{E} \times \mathcal{B}$ are if the form (ϵ, β), and the size of this element is $|(\epsilon, \beta)| = |\epsilon| + |\beta| = 0 + |\beta| = |\beta|$. Thus, there is a natural combinatorial isomorphism between $\mathcal{E} \times \mathcal{B}$ and $\mathcal{B}$.

Cartesian product and disjoint union we have already seen. It is possible to to handle the "disjoint-edness" without explicitly knowing that the sets are disjoint. We use "$+$" instead of "$\cup$". The idea is to do something like

$$\mathcal{B} + \mathcal{C} = (\{\text{blue}\} \times \mathcal{B}) \cup (\{\text{red}\} \times \mathcal{C})$$

ie - colour all the elements of one set blue and the other set red — so that the size is unchanged and so that these new coloured sets must be disjoint regardless of whether or not the original sets are.

Note that if we really have to worry about whether or not the sets are disjoint we get into trouble because we need to compute the size of the intersection

$$\text{cardinality}(\mathcal{B}_n \cup \mathcal{C}_n) = \text{cardinality}(\mathcal{B}_n) + \text{cardinality}(\mathcal{C}_n) - \text{cardinality}(\mathcal{B}_n \cap \mathcal{C}_n)$$

and we cannot get the size of the intersection from the ogfs $B(z), C(z)$ without extra information — not an admissable construction.

Now, with just union and cartesian product we could define a few finite classes, but let us add one more to our toolbox before we move on to more advanced topics.

2.6 Sequence construction

Definition (Sequence)**.** If $\mathcal{B}$ is a class then the sequence class $\textsc{Seq}(\mathcal{B})$ is defined to be the infinite sum

$$\mathcal{A} = \textsc{Seq}(\mathcal{B}) = \mathcal{E} + \mathcal{B} + (\mathcal{B} \times \mathcal{B}) + (\mathcal{B} \times \mathcal{B} \times \mathcal{B}) + \dots$$

equivalently

$$\mathcal{A} = \{(\beta_1, \beta_2, \dots, \beta_l) \text{ s.t. } \beta_j \in \mathcal{B}, l \geq 0\}$$

Notice that this only works if $\mathcal{B}$ does not contain an element of size zero (a neutral element). Also

$$\alpha = (\beta_1, \beta_2, \dots, \beta_l) \qquad \Rightarrow |\alpha| = |\beta_1| + \dots + |\beta_l|$$

Notice that if we want a sequence that contains exactly k-objects or at least k objects then we might write

$$\textsc{Seq}_k(\mathcal{B}) = \mathcal{B}^k \qquad\qquad \textsc{Seq}_{\geq k}(\mathcal{B}) = \mathcal{B}^k \times \textsc{Seq}(\mathcal{B})$$

and modify the constructions accordingly

For example, the class of binary strings is combinatorially isomorphic to $\mathcal{W} = \textsc{Seq}(\mathcal{Z}_0 + \mathcal{Z}_1)$.

The generating functions follows similarly: If $\mathcal{A} = \textsc{Seq}_k(\mathcal{B}) = \overbrace{\mathcal{B} \times \mathcal{B} \times \cdots \times \mathcal{B}}^{k \text{ times}}$ then $A(z) = B(z)^k$. Hence, we can show that

$$A(z) = 1 + B(z) + B(z)^2 + B(z)^3 \cdots = \frac{1}{1 - A(z)}.$$

So, for example, in the case of binary words, $\mathcal{W} = \textsc{Seq}(\mathcal{Z}_0 + \mathcal{Z}_1)$ and thus the role of $\mathcal{A}$ is played by $\mathcal{Z}_0 + \mathcal{Z}_1$ which has generating function $2z$. Thus, $W(z) = \frac{1}{1-2z}$.

3 Families of words

Fix a finite alphabet $\mathcal{A}$ — we consider them to all have unit size. The set of all words on the alphabet is simply the sequences of letters made from this alphabet

$$\mathcal{W} = \textsc{Seq}(\mathcal{A})$$
$$W(z) = \frac{1}{1 - A(z)} = \frac{1}{1 - z \cdot \mathrm{card}(\mathcal{A})}$$

Let us return to binary words, but keep in mind how you might generalize all of this to a larger alphabet. The shorthand $\textsc{Seq}(\mathcal{A}) = \mathcal{A}^*$ is also very common. Furthermore, we often write the atoms $\mathcal{Z}_0$ and $\mathcal{Z}_1$ as simply 0 and 1.

Consider binary words on $0, 1$. As above, we can write

$$\mathcal{W} = \textsc{Seq}(\mathcal{Z}_0 + \mathcal{Z}_1) \equiv (0 + 1)^*.$$

But we could also decompose any such word by cutting it at each 1. This gives

$$\mathcal{W} = \textsc{Seq}(\mathcal{Z}_1) \times \textsc{Seq}(\mathcal{Z}_0 \times \textsc{Seq}(\mathcal{Z}_1)) = 0^*(10^*)^*$$

which mercifully gives

$$W(z) = \frac{1}{1-z}\frac{1}{1 - z\frac{1}{1-z}} = \frac{1}{1-2z}$$

To recover the binary string we remove the parentheses and concatenate:

$$((0,0),(1,0,0),(1,0),1,1,1,1,1) \to 001001011111.$$

This type of decomposition can be very helpful. The biggest challenge is ensuring that each word is generated in a unique way. For example, $(0011+00+1)^*$ generates the string 0011 in two different ways, and hence there is not a unique representation, and we cannot translate to the generating functions using our theorems.

Example. Let us now determine a specification for a combinatorial class $\mathcal{L}$ of binary words with the additional condition that each block of 0s is of even length. Thus, 0010000111001 and 111 are in the language but 00011 and 11110 are not. We write $\mathcal{L} = (00+1)^* = \textsc{Seq}((\mathcal{Z}_0 \times \mathcal{Z}_0) + \mathcal{Z}_1) = \textsc{Seq}(\textsc{Seq}_2(\mathcal{Z}_0) + \mathcal{Z}_1)$

Example. Wesley asked, what about the binary language where blocks of 0s are of odd length? Good question. Let's try an easier problem, and we will leave the more general problem for the homework. So, we assume all blocks are of even length to mean that there are no blocks of length 0.

First, let's list the first words:

$$\mathcal{W}_{odd} = \{0, 01, 10, 010, 000, 0001, 1000, 0101, 1010, \dots\}$$

We find the specification:

$$(1+\epsilon)0(00)^*(10(00)^*)^*(1+\epsilon).$$

This translates *directly* to the generating function:

$$\begin{array}{ccccccc} \mathcal{W}_{odd} = & (1+\epsilon) & 0(00)^* & (10(00)^*)^* & (1+\epsilon) & & \\ & \updownarrow & \updownarrow & \updownarrow & \updownarrow & & \\ W(z) = & (z+1) & z\frac{1}{1-z^2} & \frac{1}{1-\frac{z^2}{1-z^2}} & (z+1) & = \frac{z(z+1)^2}{2z^2-1} = z + 2z^2 + 3z^3 + 4z^4 + \dots \end{array}$$

Yay! This agrees with what we did above. Let us discuss this decomposition, a little more. The first term, $(1+\epsilon)$ says the first letter is either 1 or is what ever comes next. The term $0(00)^*$ is a sequence of 0's of odd length: a single 0 followed by a sequence of 0's of even length. The next term is the heart of the decomposition: every 1 is followed by an odd block of 0s, and we have decomposed the string by the ones. That is, we view a binary string as a sequence of terms of the form 1, followed by an odd block of 1s. We end with either a 0 or a 1. (The ϵ says that you end with a 0 in the decomposition.)

Exercise. Now, answer the problem that wesley intended: Blocks of 0's, if they appear, have odd length.

$$\mathcal{L} = \{\epsilon, 0, 1, 11, 01, 10, 111, 110, 101, 011, 000, \dots\}$$