Bachelor's Thesis Computing Science

Generating Functions for Computing Science

Sophie Groenendaal
s1084023

March 19, 2026

First supervisor/assessor:
dr. Perry Groot

Second assessor:
dr. Engelbert Hubbers

Radboud University

Abstract

This thesis develops an accessible yet rigorous introduction to generating functions for undergraduate computer science students. Starting from sequences and recurrence relations, it motivates why classical techniques, such as iteration, guessing, and characteristic equations, can become cumbersome, and then shows how generating functions provide a systematic algebraic alternative. The core of the thesis focuses on ordinary generating functions, treating them as formal power series that encode sequences and can be manipulated through a toolbox of operations, including adding, multiplying, shifting, and differentiating. These properties are developed step by step and applied in worked examples.

Building on the same algebraic framework, the thesis then introduces probability generating functions as a probabilistic specialisation of ordinary generating functions, where coefficients represent probabilities of a discrete random variable. It demonstrates how probability generating functions simplify distributions of sums of independent variables via products, and how the expected value and variance can be derived using differentiation.

Contents

Chapter 1

Introduction

Generating functions are a core technique in discrete mathematics. For counting problems and recurrence relations, the amount of casework or algebra can grow quickly as the objects or parameters become more complex, and generating functions provide a systematic alternative. Although they are often introduced as a tool for solving recurrences, their applicability is much broader. They appear throughout combinatorics, as well as in probability (as probability generating functions), in the analysis of algorithms, and in automata theory, where they can be used to count how many words of each length are accepted by a regular language [1]. More generally, generating functions translate discrete problems into algebra so that similar steps can be reused across different kinds of problems.

Many books and papers discuss generating functions and provide examples. The material, however, is often covered briefly and quickly moves to more advanced applications. Additionally, it is not taught much to undergraduate students. Because the method lies at the intersection of discrete reasoning and more advanced series/analysis ideas, it is often postponed until students have more mathematical maturity [2]. Moreover, instructors note that, although generating functions are powerful, determining the desired coefficients from them can involve lengthy and technical calculations, which may explain why they appear less often in standard undergraduate curricula [3]. The goal of this thesis, however, is to provide a more accessible introduction to generating functions by building a clearer and stronger foundation for readers who are new to the topic and explaining why the topic may already be valuable at the undergraduate level. In particular, the thesis is motivated by the question of whether generating functions deserve a more visible place in the education of computer science students at Radboud University, given their relevance to counting, recurrence relations, probability, and algorithmic analysis.

This thesis addresses the question of how ordinary and probability generating functions can be introduced in a way that is both rigorous and acces-

sible while still enabling readers to solve non-trivial counting and recurrence problems. The main contribution is a guided development from sequences and recurrence relations to ordinary generating functions (OGFs) and their algebraic properties, supported by step-by-step examples. In addition, we study probability generating functions (PGFs), which closely resemble OGFs and reuse much of the same algebraic toolbox but encode a probability distribution. This makes it possible to derive quantities such as expected value and variance directly from the generating function. Compared to many standard introductions, the emphasis here is not on quickly reaching advanced applications but on building intuition and a reusable toolbox.

The structure of this thesis is informed by a comparison of several textbooks, lecture notes, and papers on generating functions, with attention to how they motivate the topic and develop the basic techniques. The presentation, therefore, builds the necessary toolbox step by step, first for OGFs and then, using the same ideas, for PGFs and their links to expectation and variance.

Before introducing generating functions, we briefly discuss sequences, recurrence relations, and some limitations of classical methods in Section 2. This provides the basic language needed for generating functions. Next, ordinary generating functions are introduced in Section 3.1, along with the key operations and properties in Section 3.2 (such as addition and multiplication) that make them useful in practice. After these properties are established, they are applied in more advanced examples in Section 3.3. In particular, we show how generating functions can be used to solve recurrence relations in Section 3.4. Finally, probability generating functions are introduced in Section 3.5. We compare PGF techniques with more classical probability calculations and translate probabilistic questions into the algebraic properties of PGFs, including connections to expected value and variance. The thesis concludes with a brief discussion of related work and a broader perspective on other types of generating functions in Section 4, followed by the conclusion in Section 5.

Chapter 2

Preliminaries

Before generating functions can be introduced, it is important to discuss some key concepts that will be relevant to understand in advance and to explain why generating functions are valuable to study.

2.1 Sequences

We begin with the concept of a **sequence**, as it forms the basis for understanding recurrence relations and generating functions. Sequences are central to many areas of mathematics and computer science, since they describe ordered collections of values indexed by the natural numbers. Formally, a sequence is an ordered list of elements, which may be finite or infinite [4]. We denote the n-th term of a sequence by a_n. An infinite sequence is typically written as $(a_1, a_2, a_3, \dots)$, for example $(1, 1, 1, \dots)$. A finite sequence of length n is written as $(a_1, a_2, \dots, a_n)$, for example $(1, 1, \dots, 1)$.

For instance, in computer science, a_n can represent the number of operations an algorithm performs when the input size is n. If an algorithm uses two nested loops that each run n times, then the number of operations increases as n^2, and we can describe this sequence by $a_n = n^2$.

Another example comes from data structures or combinatorics. Here, a_n might represent the number of possible configurations of a structure of size n, such as the number of binary strings of length n.

Sequences can be defined in two main ways. They can be described explicitly by giving a formula for a_n, or recursively, where each term is defined using one or more previous terms [4]. Recursive definitions naturally lead to recurrence relations, which are discussed in the next section.

To work with sequences algebraically, we can represent them using **power series**. In a power series, each term of a sequence corresponds to the coefficient of a power of x:

$$F(x) = \sum_{n=0}^{\infty} a_n x^n = a_0 + a_1 x + a_2 x^2 + \dots$$

This representation will be used in the next chapter, where it is essential for defining and working with generating functions.

2.2 Recurrence relations

Many problems in mathematics and computer science can be described by quantities that evolve step by step. When each term in a sequence depends on one or more of its preceding terms, such a relationship can be expressed as a **recurrence relation**. Recurrence relations are widely used to model and analyse iterative or recursive processes—for example, counting the number of possible configurations, analysing the runtime of recursive algorithms, or describing population growth.

A **recurrence relation** is an equation that defines a sequence in which each term is expressed as a function of one or more of its predecessors. A well-known recurrence relation is the Fibonacci sequence [4]. It begins with $f_0 = 0$ and $f_1 = 1$, and each subsequent term is obtained by summing the two preceding ones, giving the recurrence relation

$$f_n = f_{n-1} + f_{n-2}.$$

The Fibonacci sequence was originally used to model the growth of a rabbit population, but similar relations arise naturally when a problem can be decomposed into smaller subproblems, which is common in computer science and combinatorics.

Recurrence relations appear in many forms, and different techniques exist for solving them, such as iteration, characteristic equations, and generating functions. In this work, we focus on the **linear homogeneous** and **non-homogeneous** types, since these provide a clear foundation for illustrating the connection between recurrence relations and generating functions.

2.2.1 Homogeneous recurrence relations

Definition 2.2.1 (Linear homogeneous recurrence relation [4])**.** A linear homogeneous recurrence relation of degree k with constant coefficients is a recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

where $c_1, \ldots, c_k$ are real numbers, and $c_k \neq 0$.

As an illustration, the Fibonacci sequence is a linear homogeneous recurrence relation of order two. The formula is **linear** because the right-hand side is a sum of previous terms of the sequence, each multiplied by a function of n, and it is **homogeneous** because every term is expressed solely in terms of earlier terms of the sequence. Lastly, the coefficients multiplying the previous terms are **constants**, meaning they do not depend on n [4].

2.2.2 Non-homogeneous recurrence relations

Definition 2.2.2 (Linear non-homogeneous recurrence relation [4])**.** A recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + f(n),$$

where $c_1, \ldots, c_k$ are constants and $f(n)$ is a non-zero function of n, is called a linear non-homogeneous recurrence relation.

Example 2.2.3. Consider the recurrence

$$a_n = 2a_{n-1} + 3^n.$$

This recurrence is *linear* because the terms a_j occur only to the first power (no products like $a_i a_j$), and it is *non-homogeneous* because of the extra term 3^n, which is not expressed in terms of previous terms.

2.3 Limitations of classical methods

Before we introduce generating functions, it is useful to recall several classical techniques for solving problems in discrete mathematics. They are often solved using iteration, substitution, or characteristic equation methods. These classical techniques work for linear recurrences with constant coefficients.

An approach used for many problem solving tasks is iteration, which involves repeatedly expanding the recurrence until a pattern emerges.

Example 2.3.1. Consider the recurrence $a_n = 2a_{n-1}$ with $a_0 = 1$. Expanding a few steps gives

$$a_n = 2a_{n-1} = 2^2 a_{n-2} = \cdots = 2^n a_0 = 2^n,$$

a standard "iteration/unrolling" approach for recurrences [4].

This direct iteration works well when each expansion remains simple and a clear pattern emerges. However, the method can become difficult to manage when the recurrence involves multiple previous terms (e.g., $a_n = a_{n-1} + a_{n-2}$), includes additional additive parts (e.g., $a_n = 2a_{n-1} + n$), or leads to nested sums since the resulting expressions tend to expand quickly and require increasingly careful bookkeeping.

Another strategy that is often used is to guess the general shape of the solution and then verify it. For instance, for $a_n = 3a_{n-1} + 5$, one might guess a form "exponential + constant", plug it in, and solve for the unknown parameters. This can be fast when one has experience with typical forms, but there is no systematic approach to it. Especially when problems become more difficult, the guessing will also become more challenging [4].

For linear homogeneous recurrences with constant coefficients, a systematic method exists. If

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k},$$

one can try a solution of the form $a_n = r^n$, which leads to the characteristic polynomial

$$r^k - c_1 r^{k-1} - \cdots - c_k = 0.$$

When applicable, this method yields a closed form after solving the polynomial and fitting constants using initial values. Its limitation is built into its assumptions, which state that it does not apply to non-homogeneous terms (unless additional machinery is introduced), nor to variable coefficients, nor to non-linear dependence [4].

More broadly, recurrences that are non-linear, have variable coefficients, involve convolution sums, or form systems of mutually dependent sequences fall outside the scope of these classical methods and require different analytical techniques [5].

These limitations motivate the introduction of more general techniques in the next chapter, where we develop generating functions as a systematic way to encode sequences and manipulate recurrences through algebraic operations.

Chapter 3

Generating Functions

3.1 Introduction to generating functions

In this section and the ones that will follow, we will discuss generating functions. What are they, how are they built and applied, and why are they important for the computing science field of study.

Generating functions provide a bridge between discrete mathematics and algebraic methods. As the mathematician Herbert S. Wilf puts it, "A generating function is a clothesline on which we hang a sequence of numbers for display" [6]. They allow sequences, often defined recursively or combinatorially, to be manipulated using algebraic operations such as addition, multiplication, or differentiation. In this way, counting problems and recurrence relations can often be solved by those standard manipulations of generating functions, rather than by iteration or induction.

Although generating functions look like ordinary functions, in combinatorics, they are treated as formal power series, where the coefficients, not the function values, carry the essential information.

Example 3.1.1. For example:

$$F(x) = 1 + 2x + 3x^2 + 4x^3 + \dots$$

represents the **sequence** (1,2,3,4,...) and the **generating function**

$$F(x) = \sum_{n=0}^{\infty} a_n x^n$$

encodes that sequence via its coefficients a_n.

In many cases, the generating function itself can be written in closed form, which is a finite expression. For example, as a rational function like $\frac{1}{1-x}$, which makes algebraic manipulation and coefficient extraction systematic [6].

There are two main types of generating functions: the ordinary generating function and the exponential generating function [6]. In this thesis, we focus primarily on ordinary generating functions, as they already provide sufficient expressive power for the problems under consideration.

Although exponential generating functions play an important role in combinatorics and probability theory, they introduce additional technical complexity and are therefore not considered in detail here. Additional information and related perspectives are included in Section 4. Additionally, the setting of this thesis largely focuses on unlabelled objects. Labelled objects have distinct labels on their components, so two objects that differ only by renaming labels are considered different, whereas unlabelled objects are considered the same after renaming [7]. Exponential generating functions are designed for labelled constructions. Restricting attention to ordinary generating functions allows for a clearer presentation of the fundamental concepts and techniques. These will be discussed in the next subsections.

3.1.1 Ordinary generating function

The ordinary generating function forms the foundation of this thesis. This section introduces the concept and establishes the basic intuition, after which its properties and applications are developed further in the subsequent section. We begin with the formal definition of ordinary generating functions.

Definition 3.1.2. Ordinary generating function (OGF). Let $(a_0, a_1, \dots)$, be a sequence of real numbers. The **ordinary generating function** (OGF) associated with this sequence is the function whose value at x is

$$F(x) = \sum_{n=0}^{\infty} a_n x^n.$$

The numbers $a_0, a_1, \dots$ are the coefficients of the generating function [8].

The most basic generating function is $F(x) = \frac{1}{1-x}$, it generates the sequence (1,1,1,1,...). From this function, many elementary generating functions can be derived by applying algebraic operations. If we consider this constant sequence ($f_n = 1$ for all $n \geq 0$), the corresponding generating function is

$$F(x) = 1 + x + x^2 + x^3 + \cdots = \sum_{n=0}^{\infty} f_n x^n = \sum_{n=0}^{\infty} x^n = \frac{1}{1-x}.$$

The equality on the right follows from the formula for the geometric series. This equality can also be proven.

Proof. Let $x \in \mathbb{R}$ with $x \neq 1$ and the partial sum

$$S_k = 1 + x + x^2 + \cdots + x^k.$$

This partial sum translates to the sequence (1,1,1,...,1) of length $k+1$. Multiplying by x shifts every term one power up:

$$xS_k = x + x^2 + \cdots + x^{k+1}.$$

Subtracting the two expressions causes the middle terms to cancel:

$$S_k - xS_k = (1 + x + x^2 + \cdots + x^k) - (x + x^2 + \cdots + x^{k+1}) = 1 - x^{k+1}.$$

This will result in the following equation:

$$(1-x)S_k = 1 - x^{k+1} \quad \text{which implies that} \quad S_k = \frac{1 - x^{k+1}}{1-x}.$$

If we assume that $|x| < 1$, then $x^{k+1} \to 0$ as $k \to \infty$, and therefore

$$\sum_{n=0}^{\infty} x^n = \lim_{k\to\infty} S_k = \lim_{k\to\infty} \frac{1 - x^{k+1}}{1-x} = \frac{1}{1-x}.$$

Thus, for $|x| < 1$, the generating function of $(1, 1, 1, \dots)$ is

$$F(x) = 1 + x + x^2 + \cdots = \frac{1}{1-x}$$

□

Using this generating function as a building block, we can easily construct new generating functions. For example, if the generating function is finite (e.g., $1 + x + x^2 + x^3$), then we can use the function given above ($\frac{1-x^{m+1}}{1-x}$) and the sequence would become $(1, 1, 1, 1)$.

If, instead, the generating function is multiplied by x, we obtain $\frac{x}{1-x}$, which generates the sequence (0,1,1,1,...). This operation corresponds to a shift of the original sequence by one position. Likewise, differentiating $\frac{1}{1-x}$ yields $\frac{1}{(1-x)^2}$, which generates (1,2,3,4,...) Section 3.2 will have a more elaborate explanation on shifting, differentiating, and related operations on generating functions. Together, these examples illustrate how simple algebraic manipulations of generating functions translate directly into transformations of the underlying sequences.

Table 3.1 lists several common generating functions and identities that can be used as building blocks. In the sections that follow, these will be applied to solve problems, illustrating how they can be used in practice.

$(1+x)^n = \sum_{r=0}^{n} \binom{n}{r} x^r$
$\frac{1-x^{m+1}}{1-x} = \sum_{n=0}^{m} x^n = 1 + x + x^2 + \cdots + x^m$
$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n = 1 + x + x^2 + \ldots$
$\frac{1}{1-ax} = \sum_{n=0}^{\infty} a^n x^n = 1 + ax + a^2x^2 + \ldots$
$\frac{1}{1-x^r} = \sum_{n=0}^{\infty} x^{rn} = 1 + x^r + x^{2r} + \ldots$
$\frac{1}{(1-x)^2} = \sum_{n=0}^{\infty} (n+1)x^n = 1 + 2x + 3x^2 + \ldots$
$(\frac{1}{1-x})^m = \sum_{n=0}^{\infty} \binom{m+n-1}{n} x^n = (1 + x + x^2 + x^3 + \ldots)^m$

Table 3.1: Common generating-function identities. Here, $m, n, r \in \mathbb{N}$ with $r \geq 1$, and $a \in \mathbb{R}$ [9] [4].

3.2 Properties of generating functions

In this section, the fundamental algebraic properties of generating functions are introduced to show how they behave under different operations. We will focus on addition, multiplication, shifts, and differentiation, as these four operations are essential tools for analysing sequences. In Section 3.3 we will solve concrete counting problems, using these tools, but first we build the toolbox.

3.2.1 Addition

The addition of generating functions corresponds to the term wise additions of the coefficient sequences. In counting problems, this corresponds to situations where there are two separate possibilities that do not overlap. We can choose one option or the other, but not at the same time.

Example 3.2.1. Let a_n be the number of subsets of $\{1, 2, \ldots, n\}$ that are either empty or have exactly one element. These two cases are disjoint.

There is 1 empty subset and there are n subsets of size 1, so

$$a_n = 1 + n.$$

The ordinary generating function is therefore

$$F(x) = \sum_{n=0}^{\infty} a_n x^n = \sum_{n=0}^{\infty} (1+n)x^n = \frac{1}{1-x} + \frac{x}{(1-x)^2}.$$

More generally, when two disjoint cases have generating functions $F(x)$ and $G(x)$, the generating function for their union is $F(x) + G(x)$.

Theorem 3.2.2. *Let* $F(x) = \sum_{n=0}^{\infty} a_n x^n$ *and* $G(x) = \sum_{n=0}^{\infty} b_n x^n$. *Then the sum satisfies [4]*

$$F(x) + G(x) = \sum_{n=0}^{\infty} (a_n + b_n) x^n.$$

In the theorem given above, it is clearly visible that for each n, the corresponding coefficients of $F(x)$ and $G(x)$ are added together. This process continues for all n. If one generating function, for instance $G(x)$, has a longer sequence than the other, then the missing coefficients in $F(x)$ are considered to be zero. This is illustrated in the example below.

Example 3.2.3. Let $F(x) = 1+2x+3x^2$ and $G(x) = 5+4x+2x^2+x^3+x^4$. Then

$$\begin{aligned} F(x) + G(x) &= (1+5) + (2+4)x + (3+2)x^2 + (0+1)x^3 + (0+1)x^4 \\ &= 6 + 6x + 5x^2 + x^3 + x^4. \end{aligned}$$

This gives the resulting sequence $(6, 6, 5, 1, 1)$.

3.2.2 Multiplication

The multiplication of generating functions corresponds to the combination of sequences in such a way that all possible pairs of their terms are taken into account. In combinatorics, this operation represents situations where two choices can occur together rather than separately. While addition corresponds to "either/or" cases, multiplication corresponds to "both/and" situations.

For example, if one generating function counts the number of ways to choose one type of object, and another generating function counts the number of ways to choose a second type of object, their product represents the number of ways to choose one of each type simultaneously [4].

Theorem 3.2.4. *Let* $F(x) = \sum_{n=0}^{\infty} a_n x^n$ *and* $G(x) = \sum_{n=0}^{\infty} b_n x^n$. *Then the product of*

$$F(x)G(x) = \sum_{n=0}^{\infty} \left(\sum_{m=0}^{n} a_m b_{n-m} \right) x^n.$$

The coefficient of x^n in the product $F(x)G(x)$ is therefore obtained by summing all products $a_m b_{n-m}$ for which the indices add up to n. This operation is known as the **Cauchy (or Convolution) product** of two series [8].

Example 3.2.5. Let $F(x) = 1 + 2x + 3x^2$ and $G(x) = 1 + x + x^2$. Then

$$\begin{aligned} F(x)G(x) &= (1 + 2x + 3x^2)(1 + x + x^2) \\ &= 1 + 3x + 6x^2 + 5x^3 + 3x^4. \end{aligned}$$

The resulting coefficients $(1, 3, 6, 5, 3)$ can also be obtained by applying the Cauchy product formula. If we write

$$F(x)G(x) = \sum_{n=0}^{\infty} c_n x^n,$$

then c_n denotes the coefficient of x^n in the product $F(x)G(x)$, and it is given by

$$c_n = \sum_{m=0}^{n} a_m b_{n-m},$$

where a_m and b_m denote the coefficients of $F(x)$ and $G(x)$, respectively. For instance, the coefficient of x^2 is

$$c_2 = a_0 b_2 + a_1 b_1 + a_2 b_0 = 1 \cdot 1 + 2 \cdot 1 + 3 \cdot 1 = 6,$$

which matches the coefficient of x^2 in the product above.

To illustrate how multiplication models the combination of independent components, consider the following example, where the product of generating functions naturally encodes sums or concatenations of structures.

Example 3.2.6. Consider two fair six-sided dice. We are looking for the generating function $F(x) = \sum_{n=0}^{\infty} a_n x^n$, where a_n denotes the number of ordered pairs (i, j) with $1 \leq i, j \leq 6$ such that $i + j = n$. Each die can show 1 through 6, so the generating function of one die is

$$F(x) = x + x^2 + x^3 + x^4 + x^5 + x^6.$$

Since the two dice are independent, the generating function for their sum is the product

$$F(x) = F(x)^2 = (x + x^2 + x^3 + x^4 + x^5 + x^6)^2.$$

Expanding this gives

$$F(x) = x^2 + 2x^3 + 3x^4 + 4x^5 + 5x^6 + 6x^7 + 5x^8 + 4x^9 + 3x^{10} + 2x^{11} + x^{12}.$$

Multiplications of generating functions model situations in which an object is constructed by independently choosing components from two or more sets, with the coefficient of each term counting the number of ways these choices combine to produce a given outcome. This idea will reappear when we use generating functions to solve recurrence relations and count composite combinatorial structures.

3.2.3 Shifts

Besides addition and multiplication, a generating function can also be shifted forward or backward. Shifting is particularly useful when solving recurrence relations or manipulating sequences. We first consider the case of a **forward shift** [3].

Theorem 3.2.7. *Let $F(x) = \sum_{n=0}^{\infty} a_n x^n$. Then multiplying by x^k shifts the sequence forward by k positions:*

$$x^k F(x) = \sum_{n=0}^{\infty} a_n x^{n+k} = \sum_{n=k}^{\infty} a_{n-k} x^n.$$

Each term in the sequence is thus moved forward by k places. For example, multiplying a generating function by x shifts the sequence from $(a_0, a_1, a_2, \dots)$ to $(0, a_0, a_1, a_2, \dots)$. In general, multiplying by x^k results in the sequence $(0, 0, \dots, 0, a_0, a_1, a_2, \dots)$, where the first k elements are zeros.

It is also possible to shift the sequence backward, this is called **backward shifting**. This corresponds to dividing the generating function by x^k, effectively removing the first k elements of the sequence [6].

Theorem 3.2.8. *Let $F(x) = \sum_{n=0}^{\infty} a_n x^n$. Then dividing by x (and subtracting the constant term) shifts the sequence one step backward [6]:*

$$\frac{F(x) - a_0}{x} = \sum_{n=0}^{\infty} a_{n+1} x^n.$$

More generally,

$$\frac{F(x) - a_0 - a_1 x - \cdots - a_{k-1} x^{k-1}}{x^k} = \sum_{n=0}^{\infty} a_{n+k} x^n.$$

Backward shifting is useful when we want to express a sequence starting from a later index or isolate the relationship between consecutive terms in a recurrence. Together, forward and backward shifting form an essential part of translating recurrences into algebraic equations when working with generating functions. In the following example, a small demonstration can be seen of how it works.

Example 3.2.9. Let $F(x)$ be the generating function of the sequence $(1, 2, 3, 4, \dots)$:

$$F(x) = 1 + 2x + 3x^2 + 4x^3 + \dots.$$

Forward shift. Multiplying by x shifts all coefficients one position to the right:

$$xF(x) = 1x + 2x^2 + 3x^3 + \dots,$$

which corresponds to the sequence

$$(0, 1, 2, 3, \dots).$$

Backward shift. Subtracting the constant term removes the first coefficient:

$$F(x) - 1 = 2x + 3x^2 + 4x^3 + \dots.$$

Dividing by x shifts the remaining coefficients one position to the left:

$$\frac{F(x) - 1}{x} = 2 + 3x + 4x^2 + \dots,$$

corresponding to the sequence $(2, 3, 4, \dots)$.

3.2.4 Differentiation

Differentiation might seem unexpected at first, but it is also a valid and useful way to manipulate the sequence represented by a generating function. When a generating function is differentiated term by term, each coefficient is multiplied by its index, and the resulting sequence is shifted one position to the left.

Theorem 3.2.10. *If the sequence $(a_0, a_1, a_2, a_3, \dots)$ corresponds to the generating function $F(x)$, then $F'(x)$ has coefficients $(a_1, 2a_2, 3a_3, \dots)$ (shifted by one power) [5].*

To illustrate this, consider the derivative of $\frac{1}{1-x}$:

$$1 + x + x^2 + x^3 + x^4 + \cdots = \frac{1}{1 - x}.$$

Differentiating both sides gives

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

Hence, the sequence $(1, 2, 3, 4, \dots)$ corresponds to the generating function $\frac{1}{(1-x)^2}$. Formally,

$$\frac{d}{dx}\left(\sum_{n=0}^{\infty} x^n\right) = \sum_{n=1}^{\infty} nx^{n-1} = \sum_{n=0}^{\infty}(n+1)x^n, \quad \text{and} \quad \frac{d}{dx}\left(\frac{1}{1-x}\right) = \frac{1}{(1-x)^2}.$$

By multiplying both sides by x, we align the powers of x:

$$\sum_{n=0}^{\infty} nx^n = \frac{x}{(1 - x)^2}.$$

Therefore, differentiation provides a systematic way to make the factor n explicit in the coefficients. This is particularly useful when dealing with sequences that involve n as a factor [5].

Example 3.2.11. Consider the sequence $a_n = 2^n$ for $n \geq 0$. Its generating function is

$$F(x) = \sum_{n=0}^{\infty} 2^n x^n = \frac{1}{1-2x}.$$

By applying the identity $\frac{1}{1-ax} = \sum_{n=0}^{\infty} a^n x^n$ from Table 3.1, we obtain the generating function. We use this generating function to illustrate the differentiation rule and to determine the generating function of the sequence $n \cdot 2^n$. Differentiating $F(x)$ term by term gives

$$F'(x) = \sum_{n=1}^{\infty} n\, 2^n x^{\,n-1}.$$

Multiplying by x aligns the powers of x with the index of the sequence:

$$xF'(x) = \sum_{n=1}^{\infty} n\, 2^n x^n.$$

Thus, $xF'(x)$ is the generating function for the sequence $n \cdot 2^n$ (for $n \geq 0$). Using the closed form of the generating function $F(x)$, we obtain

$$G(x) = xF'(x) = \frac{2x}{(1-2x)^2}.$$

This example demonstrates how the differentiation of a generating function can be used to introduce a factor of n into the coefficients. Such factors arise naturally in non-homogeneous recurrence relations and in combinatorial counting problems where one element is distinguished or marked. Differentiation, therefore, provides a systematic way to obtain the generating function of the sequence $n \cdot a_n$ with $n \geq 0$ from the generating function of a_n with $n \geq 0$.

With these properties of generating functions established, the construction and manipulation of generating functions become more systematic. In the following sections, several examples illustrate how these properties can be applied and demonstrate the usefulness of generating functions in concrete problems.

3.3 Examples of generating functions

Now that ordinary generating functions have been introduced, along with the tools for manipulating them, this section presents several illustrative examples demonstrating their application. The first example shows a setting in which generating functions handle multiple independent constraints in a uniform way. Each constraint contributes a simple factor, and combining them reduces to multiplying these factors. In this case, the resulting

product simplifies to a closed form from which the answer can be read off immediately. By contrast, it is less clear how to obtain the same results as when directly using other standard counting techniques.

Example 3.3.1. In how many ways can we buy n apples, bananas, oranges, and pears, such that there are an even number of apples, 0 or 1 bananas, at most 4 oranges, and a multiple of 5 pears.

1. Translate the conditions to geometric series and then to closed form functions using Table 3.1:
   - Apples: $A(x) = (1 + x^2 + x^4 + x^6 + \dots) = \frac{1}{1-x^2}$
   - Bananas: $B(x) = (1 + x)$
   - Oranges: $O(x) = (1 + x + x^2 + x^3 + x^4) = \frac{1-x^5}{1-x}$
   - Pears: $P(x) = (1 + x^5 + x^{10} + x^{15} + \dots) = \frac{1}{1-x^5}$
2. Multiply all closed forms with each other and simplify:

$$\begin{aligned} F(x) &= A(x) \cdot B(x) \cdot O(x) \cdot P(x) \\ &= \left(\frac{1}{1-x^2}\right)(1+x)\left(\frac{1-x^5}{1-x}\right)\left(\frac{1}{1-x^5}\right) \\ &= \frac{(1+x)(1-x^5)}{(1-x^2)(1-x)(1-x^5)} \\ &= \frac{(1+x)(1-x^5)}{(1-x)(1+x)(1-x)(1-x^5)} \\ &= \frac{1}{(1-x)^2}. \end{aligned}$$

3. To read off the coefficient, we use the standard identity

$$\frac{1}{(1-x)^2} = \sum_{n=0}^{\infty}(n+1)x^n,$$

   which follows by differentiating $\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$.

In conclusion, the coefficient of x^n is $n+1$, meaning there are $n+1$ ways to buy the fruit.

The previous example illustrates how generating functions can encode multiple independent constraints, such as parity, bounded choices, and modular conditions, and how a closed form can then be used to extract an explicit counting formula.

A direct counting approach would require splitting into cases to handle the parity condition on apples together with the "multiple of 5" condition

on pears (and the bounded choices for bananas and oranges). Generating functions encode these constraints uniformly and avoid this casework.

The following example demonstrates the same product principle in a different setting, where generating functions are used to model concatenated structures and derive an explicit formula for the number of valid configurations.

Example 3.3.2. Suppose we want to form a password that consists of two parts:

- a sequence of lowercase letters (each of the 26 letters may be used any number of times),
- followed by a sequence of digits (each of the 10 digits may be used any number of times).

Let p_n denote the number of such passwords of total length n, where either part may be empty. Determine p_n.

1. The generating function for the letter part is
$$L(x) = 1 + 26x + 26^2x^2 + \cdots = \frac{1}{1-26x},$$
and for the digit part,
$$D(x) = 1 + 10x + 10^2x^2 + \cdots = \frac{1}{1-10x}.$$

2. Because the two parts occur **together** (letters followed by digits), the generating function for the entire password is the product:
$$P(x) = L(x)D(x) = \frac{1}{(1-26x)(1-10x)}.$$
The coefficient of x^n in $P(x)$ gives the number of possible passwords of total length n. By partial fractions,
$$P(x) = \frac{1}{16}\left(\frac{26}{1-26x} - \frac{10}{1-10x}\right),$$

3. Using $\frac{1}{1-ax} = \sum_{k=0}^{\infty} a^k x^k$ from Table 3.1, we obtain
$$\begin{aligned} P(x) &= \frac{1}{16}\left(26\sum_{k=0}^{\infty}(26x)^k - 10\sum_{k=0}^{\infty}(10x)^k\right) \\ &= \frac{1}{16}\sum_{k=0}^{\infty}\left(26^{k+1} - 10^{k+1}\right)x^k. \end{aligned}$$
Therefore,
$$p_n = \frac{26^{n+1} - 10^{n+1}}{16}.$$

In conclusion, the number of possible passwords of total length n is

$$\frac{26^{n+1} - 10^{n+1}}{16}.$$

These examples provide a foundation for the computation and application of ordinary generating functions. Together with the preceding sections, they demonstrate how ordinary generating functions can be used to encode combinatorial structures and derive explicit counting formulas. In the next section, we apply generating functions to recurrence relations. By encoding a sequence in a power series, a recurrence can often be transformed into an algebraic equation for the generating function. Solving this equation and extracting coefficients then yields an explicit formula for the sequence. The examples in that section illustrate this technique step by step.

3.4 Solving recurrences with generating functions

As mentioned before, generating functions can be very useful for solving certain recurrence relations. By rewriting a recurrence in terms of a generating function, one can often obtain an explicit expression for the generating function $F(x)$, after which the coefficients a_n can be extracted. In this section, several examples will be discussed in which recurrence relations are transformed into generating functions.

The central idea behind this method is that a recurrence relation connects different shifts of the same sequence (e.g., $a_n = a_{n-1} + a_{n-2}$). A generating function converts these shifts into algebraic operations such as multiplication by x or subtraction of initial terms. As a result, the recurrence, which is originally recursive in nature, becomes an algebraic equation in a generating function.

Solving this algebraic equation for $F(x)$ is often a convenient alternative to solving the recurrence directly. For instance, via a characteristic equation in the linear homogeneous case. Moreover, having an explicit form for $F(x)$ provides a systematic route to the coefficients a_n, although extracting these coefficients may still require additional algebraic steps.

Not every recurrence relation is equally well suited to solution by generating functions. The method is particularly effective in introductory settings for linear recurrences with constant coefficients, since shifts in the recurrence translate directly into powers of x and often lead to a manageable expression for $F(x)$. Certain non-homogeneous terms can also be incorporated effectively, especially when their generating functions have a simple closed form. In contrast, non-linear recurrences or recurrences with variable coefficients may require additional techniques or may not lead to a simple expression for $F(x)$ [6, 4].

Translating a recurrence relation into a generating function can be accomplished using a few standard steps:

1. Multiply the recurrence by x^n.
2. Sum the resulting equation over all n for which the recurrence is defined.
3. Rewrite the sums in terms of the generating function $F(x)$ using Table 3.1 and the shift rules.
4. Incorporate the initial conditions to simplify the expression.
5. Solve the resulting algebraic equation for $F(x)$.

These steps form the basis for the examples discussed in the remainder of this section.

Example 3.4.1. Solve the recurrence relation $a_n = 3a_{n-1}$ for $n = 1, 2, 3, \ldots$ with initial condition $a_0 = 2$ [4]. Let $F(x) = \sum_{n=0}^{\infty} a_n x^n$.

1. Multiply by x^n. For $n \geq 1$,
$$a_n x^n = 3a_{n-1}x^n.$$
2. Sum over all n where the recurrence holds. Summing from $n = 1$ to ∞ gives
$$\sum_{n=1}^{\infty} a_n x^n = 3\sum_{n=1}^{\infty} a_{n-1}x^n.$$
Notice that the summations start at $n = 1$, because the recurrence is only defined for $n \geq 1$. If we started at $n = 0$, the term a_{n-1} would become a_{-1}, which is not defined.[1]
3. Rewrite the sums in terms of $F(x)$. The left-hand side is
$$\sum_{n=1}^{\infty} a_n x^n = F(x) - a_0.$$
For the right-hand side, factor out x and shift the index:
$$\sum_{n=1}^{\infty} a_{n-1}x^n = x\sum_{n=1}^{\infty} a_{n-1}x^{n-1} = x\sum_{m=0}^{\infty} a_m x^m = xF(x).$$
Substituting these into the summed recurrence yields the single equation
$$F(x) - a_0 = 3xF(x).$$

[1] Alternatively, one can extend the sequence by defining $a_{-1} = a_0/3$, so that the recurrence also holds for $n = 0$ and the sums may start at $n = 0$.

4. Incorporate the initial condition. Using $a_0 = 2$:

$$F(x) - 2 = 3xF(x).$$

5. Solve for $F(x)$ and extract coefficients.

$$F(x) = \frac{2}{1-3x}.$$

Using the geometric series identity $\frac{1}{1-3x} = \sum_{n=0}^{\infty} 3^n x^n$ from Table 3.1, it follows that

$$F(x) = 2\sum_{n=0}^{\infty} 3^n x^n = \sum_{n=0}^{\infty} (2 \cdot 3^n) x^n, \qquad \text{so } a_n = 2 \cdot 3^n.$$

This first example illustrates the basic mechanism of the generating function method in its simplest form. A first-order linear recurrence translates directly into a linear equation in the generating function, which can be solved by elementary algebra. The resulting generating function has a simple rational form, making it straightforward to recover an explicit formula for the sequence.

The next example extends this approach to a higher-order recurrence relation. In this case, the recurrence involves multiple shifts of the sequence, which leads to additional terms in the generating function equation. This demonstrates how generating functions naturally handle recurrences of higher degree by systematically incorporating multiple shifts and initial conditions.

Example 3.4.2. Solve the recurrence relation $a_n = 3a_{n-1} + 2a_{n-2}$ for $n \geq 2$, with $a_0 = 1$ and $a_1 = 4$, by transforming it into its corresponding generating function.

Let $F(x) = \sum_{n=0}^{\infty} a_n x^n$.

1. Multiply the recurrence by x^n. For $n \geq 2$,

$$a_n x^n = 3a_{n-1}x^n + 2a_{n-2}x^n.$$

2. Sum over all n where the recurrence is defined. Summing from $n = 2$ to ∞ gives

$$\sum_{n=2}^{\infty} a_n x^n = 3\sum_{n=2}^{\infty} a_{n-1}x^n + 2\sum_{n=2}^{\infty} a_{n-2}x^n.$$

3. Rewrite the sums using $F(x)$ and shift rules.

$$\sum_{n=2}^{\infty} a_n x^n = F(x) - a_0 - a_1 x,$$

$$\sum_{n=2}^{\infty} a_{n-1}x^n = x\sum_{n=2}^{\infty} a_{n-1}x^{n-1} = x\sum_{m=1}^{\infty} a_m x^m = x(F(x) - a_0),$$

$$\sum_{n=2}^{\infty} a_{n-2}x^n = x^2\sum_{n=2}^{\infty} a_{n-2}x^{n-2} = x^2\sum_{m=0}^{\infty} a_m x^m = x^2F(x).$$

Substituting into the summed recurrence yields

$$F(x) - a_0 - a_1x = 3x(F(x) - a_0) + 2x^2F(x).$$

4. Incorporate the initial conditions. With $a_0 = 1$ and $a_1 = 4$,

$$F(x) - 1 - 4x = 3x(F(x) - 1) + 2x^2F(x).$$

5. Solve for $F(x)$. Rearranging gives

$$(1 - 3x - 2x^2)F(x) = 1 + x, \qquad \text{so } F(x) = \frac{1+x}{1-3x-2x^2}.$$

This example shows that even when the recurrence relation is more complex, the procedure remains largely mechanical. Once the generating function has been expressed as a rational function, algebraic techniques such as partial fraction decomposition can be used to obtain an explicit expression for the sequence. This highlights one of the main strengths of the generating function method: it reduces recursive problems to algebraic ones.

While the previous examples dealt with homogeneous recurrences, generating functions are equally effective for certain non-homogeneous recurrences. In particular, non-homogeneous terms that have well-known generating functions can be incorporated naturally into the algebraic framework. The following examples illustrate this combinatorial setting.

Example 3.4.3. Suppose we consider codewords of length n in decimal notation. We call a codeword *valid* if it contains an even number of zeros. Let a_n denote the number of valid codewords of length n. Use generating functions to find an explicit formula for a_n, where $a_0 = 1$ and $a_1 = 9$ [4]:

$$a_n = 8a_{n-1} + 10^{n-1} \qquad (n \geq 1).$$

Let $F(x) = \sum_{n=0}^{\infty} a_n x^n$.

1. Multiply the recurrence by x^n. For $n \geq 1$,

$$a_n x^n = 8a_{n-1}x^n + 10^{n-1}x^n.$$

2. Sum over all n where the recurrence is defined. Summing from $n = 1$ to ∞ gives

$$\sum_{n=1}^{\infty} a_n x^n = 8\sum_{n=1}^{\infty} a_{n-1}x^n + \sum_{n=1}^{\infty} 10^{n-1}x^n.$$

3. Rewrite the sums using $F(x)$ and standard identities.

$$\sum_{n=1}^{\infty} a_n x^n = F(x) - a_0, \qquad \sum_{n=1}^{\infty} a_{n-1} x^n = x \sum_{m=0}^{\infty} a_m x^m = xF(x),$$

and

$$\sum_{n=1}^{\infty} 10^{n-1} x^n = x \sum_{n=1}^{\infty} (10x)^{n-1} = \frac{x}{1-10x}.$$

Substituting gives the single equation

$$F(x) - a_0 = 8xF(x) + \frac{x}{1-10x}.$$

4. Incorporate the initial condition. With $a_0 = 1$,

$$F(x) - 1 = 8xF(x) + \frac{x}{1-10x}.$$

5. Solve for $F(x)$.

$$F(x)(1-8x) = 1 + \frac{x}{1-10x} \quad \Longrightarrow \quad F(x) = \frac{1}{1-8x} + \frac{x}{(1-10x)(1-8x)}.$$

To extract a_n, use partial fractions:

$$\frac{x}{(1-10x)(1-8x)} = \frac{1}{2}\left(\frac{1}{1-10x} - \frac{1}{1-8x}\right),$$

so

$$F(x) = \frac{1}{2}\left(\frac{1}{1-10x} + \frac{1}{1-8x}\right) = \sum_{n=0}^{\infty} \frac{1}{2}(10^n + 8^n)x^n.$$

Hence,

$$a_n = \frac{1}{2}(10^n + 8^n).$$

These examples demonstrate how generating functions provide a systematic and flexible framework for solving a wide range of recurrence relations. By translating recursive definitions into algebraic equations, generating functions allow both homogeneous and certain non-homogeneous recurrences to be handled in a unified manner. Beyond their theoretical elegance, this approach is particularly valuable in computer science, where generating functions are used to analyse algorithmic growth, enumerate data structures, and study discrete probability distributions. As such, generating functions form a powerful bridge between discrete mathematics and practical applications in theoretical computer science. In the next section, a different type of generating function is introduced: the probability generating function. This class of generating functions is closely related to ordinary generating functions but is specifically designed to model discrete probability distributions and their properties.

3.5 Probability generating functions

This section introduces a special type of ordinary generating function, called a **probability generating function (PGF)** [6]. Additionally, from here on, s will be used as the PGFs formal variable. In a PGF, the coefficients are not arbitrary numbers but the probabilities associated with the outcomes of a discrete random variable. As a result, a PGF encodes an entire probability distribution into a single power series.

Example 3.5.1. The probability generating function for a single six-sided die is

$$F(s) = \frac{1}{6}s + \frac{1}{6}s^2 + \frac{1}{6}s^3 + \frac{1}{6}s^4 + \frac{1}{6}s^5 + \frac{1}{6}s^6.$$

Here, the coefficient of each term represents the probability of rolling a particular face, while the exponent corresponds to the numerical value of that outcome [10].

In probability theory, we often know the set of possible outcomes of a random variable but not which outcome will occur in a given trial. A probability generating function captures this uncertainty by encoding the full probability distribution into a power series, where the exponent represents a possible outcome and the corresponding coefficient gives its probability. This makes PGFs a compact and algebraically convenient way to represent discrete probability distributions [11].

From a formal perspective, probability generating functions are ordinary generating functions with a probabilistic interpretation: the coefficients form a probability distribution and therefore sum to one. Consequently, many algebraic properties of ordinary generating functions, such as shift rules and product identities, carry over directly to PGFs. This close connection allows techniques developed for ordinary generating functions to be applied naturally in a probabilistic setting.

In the following definition, the random variable X takes numerical values. Outcomes that are naturally described by symbols, such as Heads and Tails in a coin toss, are therefore encoded numerically by defining an appropriate random variable, for example, by assigning the values 0 and 1. Any distinct non-negative integers can be used for such an encoding, although using 0 and 1 is often convenient because it keeps the exponents small and simplifies interpretation. This numerical encoding allows probability generating functions to be applied without changing the underlying probability model [12].

Definition 3.5.2 (Probability Generating Function (PGF))**.** Let X be a discrete random variable taking values in $\{0, 1, 2, \dots\}$. The **probability generating function** of X is

$$F_X(s) = \mathbb{E}(s^X) = \sum_{x=0}^{\infty} s^x \, \mathbb{P}(X = x).$$

Here, $\mathbb{E}[\cdot]$ denotes the expected value (average). For each fixed s, the quantity s^X is a random variable, and $F_X(s)$ is its expected value.

The coefficient of s^x equals $\mathbb{P}(X = x)$, while the exponent x represents the value taken by the random variable. The PGF provides a compact representation of the distribution of X and allows properties of the distribution to be obtained through algebraic operations and differentiation [13].

One of the main advantages of probability generating functions is that important characteristics of a random variable can be derived directly from the generating function. In particular, moments such as the expectation and variance, as well as the distributions of sums of independent random variables, can be obtained through algebraic operations on PGFs. These properties will be discussed in detail later in this section.

Classical probability calculations often rely on the explicit enumeration of outcomes or repeated conditioning arguments, which can become cumbersome as the complexity of a problem increases. Probability generating functions offer an alternative approach by transforming probabilistic questions into algebraic ones, thereby simplifying calculations that would otherwise require extensive case analysis.

In the remainder of this section, we explore these properties in more detail and compare the generating function approach with classical probability techniques, illustrating how PGFs provide an efficient and structured framework for probabilistic analysis.

3.5.1 Comparison with classical probability calculations

In the following examples, the probability distribution of tossing two coins is calculated in two ways. The first method is the classical one, and the second uses probability generating functions. The goal is to illustrate how probability generating functions work and why they are particularly effective in probabilistic analysis.

We will start with the example of the classical method and then continue to apply the principles of the probability generating function. In both experiments, we use fair coins, meaning each coin lands Heads or Tails with a probability of $\frac{1}{2}$.

Example 3.5.3. Let X_1 and X_2 be the outcomes of coin 1 and coin 2, where $X_i = 1$ denotes Heads and $X_i = 0$ denotes Tails. Then $\mathbb{P}(X_i = 0) = \mathbb{P}(X_i = 1) = \frac{1}{2}$ for $i = 1, 2$, and we assume X_1 and X_2 are independent. Define the random variable

$$S := X_1 + X_2,$$

which counts the total number of heads. We now determine the distribution of S in two ways.

Classical approach. Using the definition above, we list all possible outcomes of (X_1, X_2) and the corresponding value of $S = X_1 + X_2$:

Coin 1	Coin 2	Number of heads
0	0	0
0	1	1
1	0	1
1	1	2

Hence,

$$\mathbb{P}(S=0)=\frac{1}{4}, \qquad \mathbb{P}(S=1)=\frac{1}{2}, \qquad \mathbb{P}(S=2)=\frac{1}{4}.$$

PGF approach. Let X be the outcome of a single fair coin toss encoded as $X \in \{0,1\}$. Its PGF is

$$F(s)=\frac{1}{2}+\frac{1}{2}s.$$

Since X_1 and X_2 are independent, the PGF of $S=X_1+X_2$ is

$$F_S(s)=F_{X_1+X_2}(s)=F_{X_1}(s)\,F_{X_2}(s)=F(s)^2.$$

Therefore,

$$F_S(s)=\left(\frac{1}{2}+\frac{1}{2}s\right)^2=\frac{1}{4}+\frac{1}{2}s+\frac{1}{4}s^2.$$

Reading off the coefficients gives

$$\mathbb{P}(S=0)=\frac{1}{4}, \qquad \mathbb{P}(S=1)=\frac{1}{2}, \qquad \mathbb{P}(S=2)=\frac{1}{4}.$$

As can be seen, the coefficients are the same in both the classical and the PGF approach. For two coins, this can be worked out directly, but as the number of coins increases, the amount of bookkeeping grows. One would have to enumerate many more outcomes or apply a general counting argument. Using probability generating functions does not remove all work, but it provides a systematic procedure that remains the same for any number of independent coins, whether they are fair or unfair. In the unfair case, a coin with probability p of Heads has PGF $(1-p)+ps$, and the PGF of the total number of heads is again obtained by multiplying the individual PGFs.

The coin example shows how PGFs reproduce a distribution without listing outcomes. Next, we illustrate that the same idea applies even when the random variables are not identical: we add a Bernoulli variable [14] to a uniform die roll.

Example 3.5.4. Consider two independent discrete random variables. Let X be a Bernoulli random variable with $\mathbb{P}(X = 0) = \frac{1}{2}$ and $\mathbb{P}(X = 1) = \frac{1}{2}$, and let Y be the outcome of a fair four-sided die, so $\mathbb{P}(Y = n) = \frac{1}{4}$ for $n = 1, 2, 3, 4$. We determine the probability distribution of their sum $S = X + Y$ using probability generating functions.

The PGF of X is

$$F_X(s) = \frac{1}{2} + \frac{1}{2}s,$$

and the PGF of Y is

$$F_Y(s) = \frac{1}{4}s + \frac{1}{4}s^2 + \frac{1}{4}s^3 + \frac{1}{4}s^4.$$

Since X and Y are independent, we may use the fact that the PGF of a sum is the product of the PGFs of the individual variables. Theorem 3.5.5, which will be explained later, shows that the PGF of $S = X + Y$ is the product

$$F_S(s) = F_X(s)F_Y(s).$$

Expanding this product yields

$$F_S(s) = \frac{1}{8}s + \frac{1}{4}s^2 + \frac{1}{4}s^3 + \frac{1}{4}s^4 + \frac{1}{8}s^5.$$

The coefficients of these generating functions give the full probability distribution of S:

$$\mathbb{P}(S = 1) = \frac{1}{8}, \quad \mathbb{P}(S = 2) = \frac{1}{4}, \quad \mathbb{P}(S = 3) = \frac{1}{4},$$

$$\mathbb{P}(S = 4) = \frac{1}{4}, \quad \mathbb{P}(S = 5) = \frac{1}{8}.$$

Instead of enumerating all possible pairs (X, Y), a single algebraic multiplication provides the distribution of the sum.

The main takeaway is that independence turns the distribution of a sum into a product of generating functions. The next section will go into this and other useful algebraic properties and show how they can be used.

3.5.2 Algebraic properties of PGFs

The algebraic operations from Section 3.2, such as multiplication and differentiation, can also be applied to probability generating functions. Since a PGF is an infinite power series, these manipulations are justified for the values of s for which the series converges.

Convergence means that the partial sums of a series approach a finite limit as more terms are added [15]. For example, the geometric series $1 +$

$\frac{1}{2} + \frac{1}{4} + \ldots$ converges to 2, whereas $1 + 1 + 1 + \ldots$ diverges. In addition, converging absolutely means that the series still converges when each term is replaced by its absolute value.

A general theorem for power series states that there exists a radius of convergence $R \in [0, \infty]$ such that the series converges absolutely for $|s| < R$ and diverges for $|s| > R$ (the boundary case $|s| = R$ must be considered separately). For PGFs, the coefficients satisfy $p_k \geq 0$ and $\sum_{k \geq 0} p_k = 1$, which imply absolute convergence for all $|s| \leq 1$. In particular, every PGF satisfies $F_X(1) = 1$. Moreover, the series can be differentiated term-by-term for $|s| < 1$, and evaluating derivatives at $s = 1$ is valid whenever the corresponding expectation is finite [15].

In contrast, in the beginning of this thesis, we treated ordinary generating functions primarily as formal power series, so analytic convergence was not required for the algebraic manipulations performed there [8].

We start with the most important algebraic property: the sums of independent random variables correspond to the products of their PGFs.

3.5.2.1 Product rule for PGFs

One of the main advantages of PGFs is that operations on random variables translate into algebraic operations on their PGFs. The most important property is the sum of independent random variables [15].

Theorem 3.5.5. *If X and Y are independent discrete random variables, then their probability generating functions satisfy*

$$F_{X+Y}(s) = F_X(s) F_Y(s).$$

Proof. Using the definition of a PGF,

$$F_{X+Y}(s) = \mathbb{E}[s^{X+Y}] = \mathbb{E}[s^X s^Y].$$

If X and Y are independent, then s^X and s^Y are independent, hence

$$\mathbb{E}[s^X s^Y] = \mathbb{E}[s^X]\mathbb{E}[s^Y] = F_X(s) F_Y(s).$$

□

Intuitively, when we multiply the series

$$F_X(s) = \sum_x p_x s^x$$

and

$$F_Y(s) = \sum_y q_y s^y,$$

each product term $p_x q_y s^{x+y}$ corresponds to the joint outcome $(X = x, Y = y)$. Independence gives $\mathbb{P}(X = x, Y = y) = p_x q_y$, and collecting equal powers of s adds up all probabilities of pairs with the same total. This is exactly the convolution of the two distributions.

Example 3.5.6. Let X and Y be independent Bernoulli random variables with

$$\mathbb{P}(X=0)=1-p, \mathbb{P}(X=1)=p, \qquad \mathbb{P}(Y=0)=1-q, \mathbb{P}(Y=1)=q.$$

Their PGFs are

$$F_X(s)=(1-p)+ps, \qquad F_Y(s)=(1-q)+qs.$$

By the product rule for independent sums,

$$F_{X+Y}(s)=F_X(s)F_Y(s)=((1-p)+ps)((1-q)+qs).$$

Expanding gives

$$F_{X+Y}(s)=(1-p)(1-q)+(p(1-q)+(1-p)q)s+pqs^2.$$

So the coefficients yield the distribution:

$$\mathbb{P}(X+Y=0)=(1-p)(1-q), \qquad \mathbb{P}(X+Y=1)=p(1-q)+q(1-p),$$

$$\mathbb{P}(X+Y=2)=pq.$$

If you want the simplest special case, set $p=q=\frac{1}{2}$, then

$$F_X(s)=F_Y(s)=\frac{1}{2}+\frac{1}{2}s, \qquad F_{X+Y}(s)=\frac{1}{4}+\frac{2}{4}s+\frac{1}{4}s^2,$$

so $\left(\frac{1}{4},\frac{1}{2},\frac{1}{4}\right)$ for $0,1,2$ heads.

This example illustrates how the distribution of a sum can be read directly from the coefficients of the product $F_X(s)F_Y(s)$ without enumerating all joint outcomes.

Besides combining distributions via multiplication, PGFs also allow us to extract the numerical characteristics of a distribution through differentiation. The next subsection shows how differentiation can be applied to a PGF and how the derivatives of $F_X(s)$ are useful.

3.5.2.2 Differentiation of PGFs

A second valuable property of PGFs is that they can be differentiated. By differentiating once or multiple times, factorial moments can be obtained, which contain useful information such as the expected value and the variance. Assuming $\mathbb{E}[(X)_r] < \infty$, the r-th derivative $F_X^{(r)}(s)$, evaluated at $s=1$, equals the r-th factorial moment:

$$F_X^{(r)}(1)=\mathbb{E}[(X)_r].$$

For moments we use r to denote the derivative order, while for coefficient extraction we will use k to match the probability p_k.

Additionally, the probabilities $p_k = \mathbb{P}(X = k)$ can be extracted from the PGF by differentiating k times and evaluating at $s = 0$ [13]:

$$p_k = \frac{F_X^{(k)}(0)}{k!}.$$

Since

$$F_X(s) = \sum_{x=0}^{\infty} p_x s^x,$$

differentiating k times yields

$$F_X^{(k)}(s) = \sum_{x=k}^{\infty} x(x-1)\cdots(x-k+1)\, p_x\, s^{x-k},$$

and evaluating at $s = 0$ removes all terms with $x > k$ (because they still contain a positive power of s), leaving only the $x = k$ term:

$$F_X^{(k)}(0) = k!\, p_k.$$

Theorem 3.5.7. *Differentiating [13]*

$$F_X(s) = \sum_{x=0}^{\infty} p_x s^x$$

gives

$$F_X'(s) = \sum_{x=1}^{\infty} x p_x s^{x-1} = p_1 + 2p_2 s + 3p_3 s^2 + 4p_4 s^3 + \dots.$$

Below is an example of how the probabilities can be recovered via derivatives.

Example 3.5.8. Consider the PGF

$$F_X(s) = \frac{1}{2} + \frac{1}{3}s + \frac{1}{6}s^2.$$

Then,

$$p_0 = F_X(0) = \frac{1}{2}$$

Differentiating once gives

$$F_X'(s) = \frac{1}{3} + \frac{2}{6}s = \frac{1}{3} + \frac{1}{3}s.$$

So,

$$p_1 = F_X'(0) = \frac{1}{3}.$$

Differentiating twice gives

$$F_X''(s) = \frac{1}{3}.$$

Hence,

$$p_2 = \frac{F_X''(0)}{2!} = \frac{\frac{1}{3}}{2} = \frac{1}{6}.$$

Therefore, the probabilities (p_0, p_1, p_2) can be recovered from the derivatives of the PGF evaluated at $s = 0$ [13].

In the next section, we show how differentiating a PGF and evaluating the result at $s = 1$ yields the expected value and provides a direct route to the variance.

3.5.3 Expected value and variance via PGFs

Two important concepts within probability are the expected value and the variance. The formulas for these concepts can also be expressed using probability generating functions. Before doing so, we briefly recall their standard definition.

Here we return to $\mathbb{E}[X]$, the expected value of X itself (rather than of s^X, as in the PGF definition).

The **expected value** (or mean) of a random variable X represents the long-run average outcome of an experiment after many repetitions. It is commonly denoted by μ. Formally, it is defined as [12]

$$\mathbb{E}[X] = \sum_x x\,\mathbb{P}(X = x).$$

For a non-negative integer-valued X with $p_x = \mathbb{P}(X = x)$, this becomes

$$\mathbb{E}[X] = \sum_{x=0}^{\infty} x p_x.$$

The **variance** of X measures how far the possible values of the random variable typically deviate from the mean. It is defined by [12]

$$\mathrm{Var}(X) = \mathbb{E}[(X - \mu)^2].$$

Expanding the square shows the equivalent and often-used identity.

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

These classical definitions will form the basis for expressing the mean and variance in terms of probability generating functions. They can be rewritten using the probability generating function of a discrete random variable. Since the PGF $F_X(s)$ is a power series whose coefficients are the

probabilities $p(x)$, differentiating it allows us to extract information about the factorial moments of X.

Let X be a non-negative integer-valued random variable with

$$p_x = \mathbb{P}(X = x).$$

Its PGF is

$$F_X(s) = \mathbb{E}[s^X] = \sum_{x=0}^{\infty} p_x s^x.$$

For $|s| < 1$, the power series defining $F_X(s)$ converges absolutely, so it may be differentiated term-by-term. If in addition $\mathbb{E}[X] < \infty$, then the resulting series also converges at $s = 1$ and we can evaluate the derivative there. Differentiating term-by-term gives

$$F'_X(s) = \sum_{x=1}^{\infty} x p_x s^{x-1}.$$

Evaluating at $s = 1$ yields

$$F'_X(1) = \sum_{x=1}^{\infty} x p_x = \mathbb{E}[X],$$

so $\mathbb{E}[X] = F'_X(1)$ [16].

Assuming additionally that $\mathbb{E}[X^2] < \infty$, a second differentiation provides access to the second factorial moment, which leads to the following expression for the variance:

$$F''_X(s) = \sum_{x=2}^{\infty} x(x-1) p_x s^{x-2}. \qquad \text{Hence,} \qquad F''_X(1) = \mathbb{E}[X(X-1)].$$

Using the identity $X^2 = X(X-1) + X$, we have

$$\mathbb{E}[X^2] = \mathbb{E}[X(X-1)] + \mathbb{E}[X] = F''_X(1) + F'_X(1).$$

Therefore,

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = F''_X(1) + F'_X(1) - \big(F'_X(1)\big)^2.$$

These formulas illustrate the strength of the PGF approach: moments of a random variable can be computed through differentiation rather than direct summation [16]. The following two examples show how the expected value and variance can be obtained from a PGF.

Example 3.5.9 (Expected value via a PGF)**.** Let X be the outcome of a fair six-sided die, so $X \in \{1, 2, 3, 4, 5, 6\}$ and

$$F_X(s) = \frac{1}{6}(s + s^2 + s^3 + s^4 + s^5 + s^6).$$

Differentiating gives

$$F'_X(s) = \frac{1}{6}\big(1 + 2s + 3s^2 + 4s^3 + 5s^4 + 6s^5\big).$$

Using $\mathbb{E}[X] = F'_X(1)$, we obtain

$$\mathbb{E}[X] = F'_X(1) = \frac{1}{6}(1 + 2 + 3 + 4 + 5 + 6) = \frac{21}{6} = \frac{7}{2}.$$

Thus, it can be concluded that the average outcome of throwing a die many times is 3.5.

Example 3.5.10 (Variance via a PGF)**.** Encode a fair coin toss as a random variable $X \in \{0, 1\}$, where $X = 1$ corresponds to Heads and $X = 0$ to Tails. Then

$$F_X(s) = \mathbb{E}[s^X] = \tfrac{1}{2} + \tfrac{1}{2}s.$$

Differentiate:

$$F'_X(s) = \tfrac{1}{2}, \qquad F''_X(s) = 0.$$

Evaluating at $s = 1$ gives $\mathbb{E}[X] = F'_X(1) = \frac{1}{2}$. Moreover,

$$\mathbb{E}[X(X - 1)] = F''_X(1) = 0,$$

so $\mathbb{E}[X^2] = \mathbb{E}[X(X - 1)] + \mathbb{E}[X] = 0 + \frac{1}{2} = \frac{1}{2}$. Therefore,

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - \big(\mathbb{E}[X]\big)^2 = \tfrac{1}{2} - \big(\tfrac{1}{2}\big)^2 = \tfrac{1}{4}.$$

These examples illustrate that relatively simple derivative computations on a PGF can yield the expected value and variance of a discrete random variable without evaluating the defining sums term-by-term. This reinforces the usefulness of PGFs as an algebraic interface to probabilistic quantities, especially when direct summation becomes cumbersome.

Chapter 4

Related Work

This thesis focuses on ordinary generating functions and probability generating functions and uses them to illustrate a general workflow for solving counting problems and recurrence relations. In doing so, it necessarily leaves out a large part of the generating function literature. The literature on generating functions is broad, and several closely related frameworks are commonly used depending on the type of objects being counted, the presence of labels, and whether one is interested in probabilistic or asymptotic information. In this section, we will provide some other types and extensions of generating functions.

4.1 Different types of generating functions

A first major extension beyond OGFs is the exponential generating function.

Definition 4.1.1. Exponential generating function (EGF). If $(a_0, a_1, a_2, \dots)$ is an infinite sequence of integers, the exponential generating function [3] of $(a_n)_{n\in\mathbb{N}}$ is the function

$$F(x) = \sum_{n=0}^{\infty} \frac{a_n}{n!} x^n.$$

While OGFs are particularly natural for many unlabelled counting problems, EGFs are standard when counting labelled combinatorial structures, such as permutations and labelled graphs. In that setting, EGFs often lead to cleaner algebraic rules that match the combinatorial operations on labelled objects. This distinction between ordinary and exponential generating functions is treated in applied combinatorics texts and generating function references, and it motivates why many introductions to combinatorics present OGFs and EGFs as two foundational forms of generating functions [16, 6].

Additionally, there are several other special generating functions that are used in particular areas. For example, Dirichlet series generating functions

are discussed along with broader formal theories of generating functions [6].

Definition 4.1.2. Given a sequence $\{a_n\}_1^\infty$; we say that a formal series

$$\begin{aligned} F(s) &= \sum_{n=1}^{\infty} \frac{a_n}{n^s} \\ &= a_1 + \frac{a_2}{2^s} + \frac{a_3}{3^s} + \frac{a_4}{4^s} + \dots \end{aligned}$$

is the Dirichlet series generating function (DSGF) of the sequence.

A Dirichlet series generating function encodes $\{a_n\}$ using weights n_s instead of x^n. This form is common in number theory because it interacts naturally with divisibility and prime factorisation, and for suitable s, it converges and can be studied as an analytic function. Along with other variants, this shows that the choice of generating function depends on the structure of the problem, whether that structure is combinatorial (labelled versus unlabelled), analytic (asymptotic and growth rates), or probabilistic (distributions and moments).

4.2 Multivariate and compositional frameworks

Within a type of generating function, many variations are possible. Bóna's Introduction to Enumerative and Analytic Combinatorics places generating functions in a broader toolkit for counting and extracting further information from sequences [17]. Beyond the basic one-variable generating functions, he continues to multivariate generating functions, where additional variables can mark extra parameters (for instance, counting objects of size n while also tracking a statistic k) [17]. This perspective shows that generating functions are not only a method for obtaining closed forms or solving recurrences, but also a way to encode more detailed combinatorial information in a single object.

More generally, the literature emphasises the standard translation between combinatorial constructions and algebraic operations on generating functions. For example, multiplying generating functions corresponds to combining objects from two classes into a single object whose size is the sum of the sizes, while composition corresponds to building objects by replacing each component of a structure with an object from another class [17, 6].

4.3 Generating functions in probability

On the side of probability, PGFs are an important tool for non-negative integer-valued random variables. However, they are part of a larger group

of generating function methods used in probability theory and statistics. Moment generating functions (MGFs) are closely related to them, they represent a distribution via $M(t) = \mathbb{E}[e^{tX}]$ and allow moments to be obtained by differentiation at $t = 0$ [12, 18]. In many probability texts, MGFs are introduced alongside other function representations because they make certain computations systematic, especially for sums of independent random variables and for extracting summary statistics [12]. Compared to PGFs, MGFs are not restricted to integer-valued variables, which explains why they are widely used in statistics, while PGFs are particularly natural when the random variable represents a count.

4.4 Concluding remarks

Overall, the related literature supports the main design choice of this thesis, which focuses on OGFs as a broadly applicable starting point in discrete mathematics and then introduces PGFs as a natural specialisation where the same algebraic toolbox has a probabilistic interpretation. The different types of generating functions given above extend the scope beyond the generating functions discussed in this research. It allows for a more advanced step in generating functions, especially for more specific applications of the material [17, 6].

Chapter 5

Conclusions

The goal of this thesis is to provide an accessible introduction to generating functions by building a clearer and stronger foundation for readers who are new to the topic and to explain why it would be valuable for undergraduate students at Radboud University. After the research that was conducted, we can draw some conclusions.

This thesis demonstrates that generating functions are a powerful tool for making computations easier. A generating function can be used to derive an explicit formula. We looked at ordinary generating functions as the foundation of this topic since they provide the basic tools and building blocks that can be used for other generating functions. In addition to these building blocks, there are algebraic properties, including multiplication, addition, differentiation, and shifts, which can help build and manipulate generating functions to achieve an outcome.

Moreover, there is a standard framework to which generating functions can be applied. For example, a recurrence relation can be translated into a closed formula by applying a set of steps. One of the advantages of this approach is that recurrence relations become easier to analyse. They are best suited for linear recurrences with constant coefficients.

Additionally, a probability generating function is a special type of ordinary generating function that encodes an entire probability distribution in a single power series. Its main advantage is that key properties of a discrete random variable can be derived directly from the generating function. For example, the expected value and variance follow from simple algebraic operations on the PGF, such as differentiation and evaluation at $s = 1$.

Furthermore, probability generating functions obey the same core algebraic rules as ordinary generating functions, meaning the techniques developed for ordinary generating functions, such as manipulating series through addition, multiplication, and differentiation, carry over naturally to the probabilistic setting.

Generating functions can be useful in more contexts than recurrence

relations. They are often used in counting problems in combinatorics, where the coefficients represent the number of ways to construct an object of a certain size, such as strings, paths, or partitions. They can also be used to derive identities or closed forms for sums and to study the behaviour of sequences, for example, growth rates. In probability, probability generating functions make it easier to work with discrete distributions and sums of independent random variables.

On top of that, there are many more types of generating functions. They differ in form and application, allowing the user to choose a form that fits their problem. It is nevertheless understandable that generating functions are not always emphasised in undergraduate teaching, since the method requires students to move comfortably between discrete reasoning and formal power series manipulations. However, this thesis also indicates that generating functions could form a valuable addition to the mathematical toolbox of computer science students at Radboud University. Their relevance extends beyond recurrence relations, since they also provide useful methods for counting problems, discrete probability, and the analysis of sequences. As such, they may help students develop a broader and more connected understanding of topics within discrete mathematics and theoretical computer science.

Bibliography

[1] Philippe Flajolet. "Analytic Analysis of Algorithms". In: *Automata, Languages and Programming*. Ed. by Werner Kuich. Vol. 623. Lecture Notes in Computer Science. Springer, 1992.

[2] Christopher A. Healy. "Generating Functions Without Toil". In: *Journal of Computing Sciences in Colleges* 39.4 (2023). Papers of the 30th Annual CCSC Midwestern Conference, pp. 64–73. DOI: 10.5555/3637036.3637043. URL: https://www.ccsc.org/publications/journals/MW2023.pdf (visited on 02/23/2026).

[3] Michel X. Goemans. *Generating Functions*. Lecture notes for 18.310: Principles of Applied Mathematics, Massachusetts Institute of Technology. 2015. URL: https://math.mit.edu/~goemans/18310S15/generating-function-notes.pdf (visited on 11/08/2025).

[4] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. 8th. McGraw-Hill Education, 2019. ISBN: 9781259676512.

[5] Eric Lehman, F. Thomson Leighton, and Albert R. Meyer. *Mathematics for Computer Science*. Massachusetts Institute of Technology, 2010. URL: https://people.csail.mit.edu/meyer/mcs.pdf (visited on 12/12/2025).

[6] Herbert S. Wilf. *Generatingfunctionology*. Academic Press, Inc., 1990. ISBN: 9780127519555.

[7] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. ISBN: 9780521898065.

[8] Edward A. Bender and S. Gill Williamson. *Foundations of Combinatorics with Applications*. Dover Publications, 2006. ISBN: 9780486446035.

[9] George E. Martin. *Counting: The Art of Enumerative Combinatorics*. Springer, 2001. ISBN: 9781441929150.

[10] University of Cambridge Computer Laboratory. *Course Material 2007–08: Probability*. Course webpage. 2008. URL: https://www.cl.cam.ac.uk/teaching/0708/Probabilty/prob06.pdf (visited on 12/10/2025).

[11] Robert A. Beeler. *How to Count. An Introduction to Combinatorics and Its Application*. Springer, 2015. ISBN: 9783319138435.

[12] Sheldon M. Ross. *A First Course in Probability*. 8th. Pearson, 2010. ISBN: 9780136033134.

[13] Rachel Fewster. *Course Notes: STATS 325 Stochastic Processes*. Lecture notes (PDF), Department of Statistics, The University of Auckland. 2014. URL: https://www.stat.auckland.ac.nz/~fewster/325/notes/2012S2/325book.pdf (visited on 03/23/2026).

[14] Will Monroe. *Bernoulli and Binomial Random Variables*. Lecture notes. 2017. URL: https://web.stanford.edu/class/archive/cs/cs109/cs109.1208/lectureNotes/LN07_bernoulli_binomial.pdf (visited on 01/11/2025).

[15] Geoffrey R. Grimmett and David R. Stirzaker. *Probability and Random Processes*. 3rd. Oxford University Press, 2001. ISBN: 9780198572220.

[16] Fred S. Roberts and Barry Tesman. *Applied Combinatorics*. 3rd. Chapman and Hall / CRC Press, 2024. ISBN: 9781032816524.

[17] Miklós Bóna. *Introduction to Enumerative and Analytic Combinatorics*. 3rd. Chapman and Hall / CRC Press, 2025. ISBN: 9781032302706.

[18] Christoffer Rydén. *Generating Functions: Powerful Tools for Recurrence Relations. Hermite Polynomials Generating Function*. Online notes / article. 2023. URL: https://www.diva-portal.org/smash/get/diva2:1766820/FULLTEXT01.pdf (visited on 01/17/2026).