Algorithmic Manipulations and Transformations of Univariate Holonomic Functions and Sequences

Diplomarbeit

zur Erlangung des akademischen Grades
"Diplom-Ingenieur"
in der Studienrichtung
Technische Mathematik

Verfaßt von
Christian Mallinger

Angefertigt am Institut für Mathematik
der Technisch-Naturwissenschaftlichen Fakultät
an der Johannes Kepler Universität Linz

Eingereicht bei
o.Univ.-Prof. Dr. Bruno Buchberger

Betreut durch
Univ.Ass. Dr. Peter Paule

RISC-Linz, August 1996

Abstract

Holonomic functions and sequences have the property that they can be represented by a finite amount of information. Moreover, these holonomic objects are closed under elementary operations like, for instance, addition or (termwise and Cauchy) multiplication. These (and other) operations can also be performed "algorithmically". As a consequence, we can prove any identity of holonomic functions or sequences automatically. Based on this theory, the author implemented a package that contains procedures for automatic manipulations and transformations of univariate holonomic functions and sequences within the computer algebra system Mathematica . This package is introduced in detail. In addition, we describe some different techniques for proving holonomic identities.

Zusammenfassung

Holonomische Funktionen und Folgen haben die Eigenschaft, daß sie durch eine endliche Menge an Information dargestellt werden können. Darüberhinaus sind diese holonomischen Objekte unter elementaren Operationen wie z. B. Addition oder (gliedweise und Cauchy-) Multiplikation abgeschlossen. Diese (und andere) Operationen können auch „algorithmisch" durchgeführt werden. Als Konsequenz daraus, kann man jede Identität holonomischer Folgen und Reihen automatisch beweisen. Basierend auf diese Theorie implementierte der Autor ein Programmpaket, das Prozeduren zum automatischen Manipulieren und Transformieren univariater holonomischer Funktionen und Folgen innerhalb von Mathematica enthält. Dieses Paket wird im Detail vorgestellt. Zusätzlich beschreiben wir einige verschiedene Techniken zum Beweisen holonomischer Identitäten.

Part of this work was supported by a sholarship of the Austrian ministry for science and research.

Für Silvia, Theresa und Hannah

Contents

Chapter 0

Introduction

0.1 Summary

We want to give here a short summary of the contents of this thesis. This summary contains many heuristics, exact definitions, problem specifications, and proofs can be found in the subsequent chapters.

The context of all considerations in this thesis is the world of holonomic (or D-finite) functions and holonomic (or P-recursive, P-finite) sequences. A holonomic function is a solution of a linear differential equation with polynomial coefficients. Examples of holonomic functions are all algebraic functions, in particular polynomials and rational functions, as well as the most important transcendental functions like $\sin(x)$, $\cos(x)$, e^x. Also "multivariate" functions like the classical orthogonal polynomials are holonomic, if we view them as functions of the continuous variable.

A holonomic sequence satisfies a linear recurrence equation with polynomial coefficients. Examples of holonomic sequences are the important family of hypergeometric sequences or classical combinatorial sequences as the Fibonacci and the Catalan numbers. Considered as sequences in the discrete variable, the classical orthogonal polynomials are also holonomic.

There are two main reasons that motivate the work with holonomic functions and sequences.

- Every holonomic object (function or sequence) has the property that it is *uniquely defined* by a *finite* amount of information, i.e., by a holonomic (differential or recurrence) equation and some initial conditions. (This implies that we can perform zero recognition in the holonomic world.) This holonomic equation together with a sufficient number of initial values can be regarded as a *holonomic representation* for a holonomic object.

- A lot of unary, binary and $n-$ary operations preserve holonomicity: Holonomic functions are closed under addition, multiplication, integration, differentiation, and composition with algebraic functions. Holonomic sequences are closed under addition, multiplication, shifts, differences, partial summation and interlacement.

Moreover, if we have the holonomic representations of some objects, we can use *algorithms* (and therefore also computers) to give holonomic representations for manipulations and transformations of these holonomic objects. For instance, if we know holonomic differential equations that are satisfied by $\sin(x)$ and e^x, respectively, it is possible to come up with a holonomic differential equation that is satisfied by $\sin(x)e^x$, by $\sin(x) - e^x$ or even by $\int_0^x (t^2+1)\sin(t^2-3t)e^{-t}\,dt$. No need to say that the computer is able to do these jobs.

Another consequence of these two properties is the fact that it is possible to "routinely" verify any identity involving holonomic functions or sequences that are manipulated by operations that preserve holonomicity: All we have to do is, to bring everything to one side of the equation, and perform zero recognition on the resulting object. For example, we can automatically verify Cassini's identity for the Fibonacci numbers:

$$F_{n+1}F_{n-1} - F_n^2 = (-1)^n.$$

Another simple identity, which can be proved automatically, is the binomial theorem

$$(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k.$$

The organization of this thesis is as follows: In Chapter 1 we establish the theory of holonomic functions and sequences. We take a look at the algebraic background, establish a relation between holonomic functions and sequences, and give the closure properties, which are a main motivation for the work with holonomic objects. In addition, we give most of the proofs in a constructive way, in order to make it easy to extract the relevant algorithms that are needed to automatize the manipulations and transformation of holonomic objects. We also give a lot of examples (some of them containing solutions to "real life" problems); with respect to theory these examples should serve both as motivation and illustration.

Based on the theoretical background that is established in Chapter 1, the author implemented the Mathematica package `GeneratingFunctions`. This package, which follows the basic ideas of the Maple package `gfun` by B. Salvy and Paul Zimmermann, is introduced in detail in Chapter 2. We describe the functionality of `GeneratingFunctions` as well as the syntactical structure of admissible input. Hence, this chapter may be regarded as a manual for this Mathematica package. We also show how the procedures in the package may be used in the process of problem solving.

Given the first terms of a power series or sequence, it is possible to "guess" a holonomic equation that is satisfied by the whole function or sequence. In Chapter 3 we present some ideas, how "guessing" can help in situtations where we are faced with operations that do not satisfy the closure properties. Moreover we translate the theory of holonomic functions and sequences into the operator algebra language of Ore polynomial rings. This operator algebra is used to present some (more ore less efficient) methods to prove identies of holonomic sequences.

0.2 Notation

The symbols $\mathbf{N}$, $\mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$ and $\mathbf{C}$ stand for the sets of nonnegative integers, all integers, rational, real and complex numbers. Throughout this thesis the symbol $\mathbf{K}$ denotes a field of characteristic zero. For algorithmic purposes $\mathbf{K}$ should be computable, i.e., every element in the field should have a finite representation and it should be possible to carry out the field operations within a finite amount of time. (Note, that neither $\mathbf{C}$ nor $\mathbf{R}$ have this property). $\mathbf{K}[x]$ stands for the ring of polynomials, $\mathbf{K}(x)$ for the quotient field of rational functions over $\mathbf{K}$. By $[\mathbf{K}_n]$ and $(\mathbf{K}_n)$ we denote the ring of polynomial sequences and the field of rational sequences (with values in $\mathbf{K}$), respectively. $\mathbf{K}[[x]]$ denotes the ring of formal power series, $\mathbf{K}((x))$ is the field of formal Laurent series over $\mathbf{K}$.

For $k \in \mathbf{N}$, the symbol $n^{\overline{k}}$ denotes the kth rising factorial of n, which is defined as follows:

$$n^{\overline{k}} = n(n+1)\cdots(n+k-1), \text{ if } k > 0,$$

$$n^{\overline{0}} = 1.$$

If $f = \sum_{n \geq 0} a_n x^n$, then the coefficient of x^n in f, is given by $[x^n]f = a_n$.

0.3 Acknowledgements

I joined the combinatorics group at the RISC in autumn 1992. Since that time, it was Peter Paule, who taught me most of theory and practice one needs in the world of mathematics. I want to thank him for constantly keeping me at the right track in the process of writing this thesis.

I also would like to thank all the members of the combinatorics group, who contributed to this work in form of discussions, questions or by pointing out shortcomings. My special thanks go to István Nemes and Roberto Pirastu for always willingly answering my questions.

I am also grateful to Marko Petkovšek for the electronic discussions with him via e-mail.

Finally I must not forget to thank my family, who made all this possible.

Chapter 1

The Holonomic Universe

1.1 Introduction

This chapter is devoted to the discussion of holonomic functions (or more precisely: power series) and sequences. A Holonomic function satisfies a linear differential equation with polynomial coefficients, a holonomic sequence is the solution of a linear recurrence equation with polynomial coefficients. The basic theory and algebraic background are given in Sections 1.2 and 1.3, respectively, where we also present some famous members of the holonomic family. The results discussed there and in Section 1.4 are well known, a comprehensive exposition of the subject was given by Stanley [Sta80].

In fact the algebraic properties of holonomic functions and sequences, presented in Section 1.4 are the main motivation to work in the holonomic universe: Most elementary operations like addition, (termwise or Cauchy) multiplication and several univariate transformations preserve holonomicity. These properties, together with the fact that holonomic functions (and also sequences) are representable by a finite amount of information, make it possible to "compute" representations of functions or sequences that are built from other (elementary) holonomic functions or sequences via these operations. Moreover, any identity in this frame can be proved by performing some routine steps.

In Section 1.5 we briefly discuss multivariate holonomicity, without going into details. The interested reader may consult the literature we refer to.

We aimed to give most of the proofs in this chapter in a constructive way, which should make it easy to extract algorithms from these proofs. Some of the algorithms also can be found in [SZ94].

In order to illustrate the theory we give a lot of examples. Since they often contain solutions to "real life" problems, these examples shall also serve as a motivation for the whole theory.

1.2 Holonomic Functions

We are working with *formal power series* over a field $\mathbf{K}$. A power series $f(x)$ is an object of the form $f(x) = \sum_{n\geq 0} a_n x^n$, where $a_n \in \mathbf{K}$ for all $n \in \mathbf{N}$. One also says that $f(x)$ is the *generating function* of the sequence $(a_n)_{n\geq 0}$. We want to note here that power series are *formal objects*, rather than functions in the classical sense, i.e., mappings from the ground field $\mathbf{K}$ or from a subset of $\mathbf{K}$ into $\mathbf{K}$. A power series may correspond to a certain mapping from a subset of $\mathbf{K}$. For example, let $\mathbf{K} = \mathbf{C}$, $\sum_{n\geq 0} x^n$ corresponds to the *analytic function* $f : x \to (1-x)^{-1}$. This function is rational and has a pole at $x = 1$. The power series is convergent if and only if $|x| < 1$. Although the power series and the function have different analytic properties, we will not run into troubles, since we regard the power series as a formal object and do not care about poles or other *analytic* properties like radius of convergence, continuity, etc.

Every power series is the generating function of a sequence in the ground field. Hence we will also use the word *function* to denote a formal power series.

Let $f(x) = \sum_{n\geq 0} a_n x^n$ and $g(x) = \sum_{n\geq 0} b_n x^n$ be two power series. We define addition and (Cauchy) multiplication as follows:

$$f(x) + g(x) := \sum_{n\geq 0} (a_n + b_n) x^n$$

$$f(x) g(x) := \sum_{n\geq 0} (\sum_{k=0}^{n} a_k b_{n-k}) x^n$$

The domain of power series over $\mathbf{K}$ with addition and (Cauchy) multiplication is a commutative ring without zero divisors, denoted by $\mathbf{K}[[x]]$. The two functions $f(x)$ and $g(x)$ are equal, written as $f(x) = g(x)$, if and only if $a_n = b_n$ for all $n \in \mathbf{N}$.

The derivative $f'(x)$ is defined to be

$$f'(x) := \sum_{n\geq 0} (n+1) a_{n+1} x^n.$$

It follows immediately that

$$a_n = [x^n] f(x) = [x^0] f^{(n)}(x)/n! = f^{(n)}(0)/n!.$$

Definition 1.2.1 (holonomic functions) A function $f \in \mathbf{K}[[x]]$ is *holonomic* if and only if f satisfies a linear differential equation with polynomial coefficients (*holonomic differential equation*), i.e., there exist $d \in \mathbf{N}$ and polynomials $p_0, p_1, \ldots, p_d$ in $\mathbf{K}[x]$, $p_d \neq 0$, such that

$$p_0(x) f(x) + p_1(x) f'(x) + \cdots + p_d(x) f^{(d)}(x) = 0. \tag{1.2.1}$$

The nonnegative integer d denotes the *order* of the holonomic equation. The *degree* of equation (1.2.1) is given by $\max\{\deg(p_i(x))|0 \leq i \leq d\}$. In literature holonomic functions are also called *differentiably finite*, *D-finite* [Sta80], [Lip89], [GKP94] or *simple functions* [Koe92].

Subsequently, we will sometimes write f and p instead of $f(x)$ and $p(x)$, if the indeterminate x is clear from the context.

If we extend our working domain from the ring $\mathbf{K}[[x]]$ to $\mathbf{K}((x))$, the field of *formal Laurent series*, i.e, series of the form $\sum_{n\geq n_0} a_n x^n$ with $n_0 \in \mathbf{Z}$, we can multiply a power series with any rational function, because those can be expanded as Laurent series. Thus it makes sense to regard $\mathbf{K}((x))$ as a vector space over $\mathbf{K}(x)$.

The following theorem [Sta80] gives alternatives for defining holonomic functions.

Theorem 1.2.1 For $f \in \mathbf{K}[[x]]$ the following three conditions are equivalent.

(i) There are $d \in \mathbf{N}$ and polynomials $q, p_0, p_1, \ldots, p_d$ in $\mathbf{K}[x]$, $p_d \neq 0$, such that

$$q(x) + p_0(x)f(x) + p_1(x)f'(x) + \cdots + p_d(x)f^{(d)}(x) = 0. \tag{1.2.2}$$

(ii) f is holonomic.

(iii) The linear space spanned by $\{f^{(k)}(x)|k \in \mathbf{N}\}$ is a finite dimensional subspace of $\mathbf{K}((x))$ over $\mathbf{K}(x)$.

Proof.

(i)⇒(ii). Suppose f satisfies (1.2.2). Differentiate this equation to get, say, $(1.2.2)'$. f clearly satisfies $q'(x)(1.2.2) - q(x)(1.2.2)'$, which is a homogenous holonomic differential equation of order $d+1$.

(ii)⇒(iii). Let f satisfy equation (1.2.1). It is evident that $f^{(d)}(x)$ lies in $L := \langle f, f', \ldots, f^{(d-1)}\rangle$, the linear hull of $\{f, f', \ldots, f^{(d-1)}\}$. If we repeatedly differentiate equation (1.2.1), it becomes clear that for every $k \in \mathbf{N}$, we have $f^{(d+k)} \in L$. Hence $\langle f, f', \ldots\rangle = L$, which has dimension less or equal d over $\mathbf{K}(x)$.

(iii)⇒(i). Assume $\langle f^{(k)}(x)|k \in \mathbf{N}\rangle$ has finite dimension, say d, over $\mathbf{K}(x)$. This means that any $d+1$ elements of $\{f^{(k)}(x)|k \in \mathbf{N}\}$ are linearly dependent. In fact equation (1.2.2), with $q(x) = 0$, is a linear dependence relation with cleared denominators. □

In view of Theorem 1.2.1, we also call an inhomogenous equation of type (1.2.2) holonomic. Whenever we want to emphasize distinctions between equations (1.2.1) and (1.2.2), we will use the adjectives *homogenous* and *inhomogenous*, respectively.

We give some simple examples of holonomic functions.

Example 1.2.1

1. Let $f(x) = p(x)/q(x)$ where $p(x), q(x)$ are polynomials over $\mathbf{K}$. Rational functions are holonomic. They satisfy the holonomic differential equation

$$(p'(x)q(x) - p(x)q'(x))f(x) + (-p(x)q(x))f'(x) = 0.$$

2. Let $\mathbf{K} = \mathbf{C}$. The trigonometric functions $f(x) = \sin(x)$, $g(x) = \exp(x)$ and $h(x) = \arctan(x)$ are holonomic since they satisfy the differential equations

$$\begin{aligned} f(x) + f''(x) &= 0, \\ g(x) - g'(x) &= 0, \\ h'(x) &= \frac{1}{1+x^2}. \end{aligned}$$

(Note that the holonomicity of $h(x)$ follows from Theorem 1.2.1.)

3. The generating function $f(y) = \sum_{n \geq 0} C_n^\alpha(x) y^n$ of the *Gegenbauer polynomials* $C_n^\alpha(x)$ (see, e.g. [AS64]) is given by $f(y) = (1 - 2xy + y^2)^{-\alpha}$, where $\alpha \in \mathbf{R}$ satisfies $\alpha > -1/2$. This function is a solution of the differential equation

$$(-\alpha\, x + \alpha\, y)\ f(y) + \left(1 - 2\, x\, y + y^2\right)\ f'(y) = 0 \tag{1.2.3}$$

with the initial condition $f(0) = [y^0]f(y) = 1$. Considering α as an indeterminant in $\mathbf{R}$, we can set $\mathbf{K} = \mathbf{R}(\alpha, x)$.

□

The simple functions in the previous example should not mislead us to assume that all "elementary" functions are holonomic. For example, we can easily show that the tangent function $f(x) = \tan(x)$ is not holonomic (See also [KS94]): We know that $\tan(x)$ satisfies the derivation rule

$$\tan'(x) = 1 + \tan^2(x).$$

By an induction argument it follows that also higher derivatives of the tangent function are polynomials in $\tan(x)$. Now, suppose that $\tan(x)$ satisfies a holonomic differential equation of the form (1.2.1). Replacing the derivatives of $f(x)$ by the corresponding polynomials in $\tan(x)$, gives an algebraic equation for the tangent function. It is well known that the tangent function is not algebraic, therefore it can not satisfy a holonomic differential equation. (Analogous argumentations are used in Stanley's [Sta80] proof that $\sec(x)$ is not holonomic.)

1.3 Holonomic Sequences

In this section we briefly discuss the discrete counterpart to holonomic functions:

Definition 1.3.1 (holonomic sequences) A sequence $(a_n)_{n\geq 0} \in \mathbf{K}^{\mathbf{N}}$ is *holonomic* if and only if $(a_n)_{n\geq 0}$ satisfies a linear recurrence with polynomial coefficients (*holonomic recurrence equation*), i.e., there are polynomials $p_0, p_1, \ldots, p_d$ in $\mathbf{K}[x]$, $p_d \neq 0$, such that for all $n \in \mathbf{N}$:

$$p_0(n)a_n + p_1(n)a_{n+1} + \cdots + p_d(n)a_{n+d} = 0. \tag{1.3.1}$$

The nonnegative integer d is the *order* of the holonomic recurrence. As in the case of holonomic functions the integer $\max\{\deg(p_i(x)) | 0 \leq i \leq d\}$ is called the *degree* of recurrence (1.3.1). In literature holonomic sequences are sometimes called *polynomially recursive*, *P-recursive* [GKP94], [Sta80] or *P-finite* [Zei90].

Let $[\mathbf{K}_n]$ be the ring of polynomial sequences in $\mathbf{K}$ with addition and termwise (Hadamard) multiplication. Since $[\mathbf{K}_n]$ has a unit element and no zero divisors, we can define $(\mathbf{K}_n)$, the field of rational sequences, to be the set

$$(\mathbf{K}_n) = \{\frac{\mathbf{p}}{\mathbf{q}} \mid \mathbf{p} = (p(n))_{n\geq 0}, \mathbf{q} = (q(n))_{n\geq 0} \in [\mathbf{K}_n]\}/\!\sim,$$

where $\mathbf{p}/\mathbf{q} \sim \overline{\mathbf{p}}/\overline{\mathbf{q}}$ if and only if $p(n)\overline{q}(n) = q(n)\overline{p}(n)$ for all $n \in \mathbf{N}$. In the given case of rational sequences the relation $\sim$ is equivalent to $\sim_\infty$, where $(a_n)_{n\geq 0} \sim_\infty (b_n)_{n\geq 0}$ if and only if there is an $n_0 \in \mathbf{N}$ such that $a_n = b_n$ for all $n \geq n_0$. (Stanley [Sta80] called the equivalence classes of $\sim_\infty$ the *germs at infinity.*) Let $\mathbf{r} \sim \mathbf{p}/\mathbf{q}$, with $\mathbf{p}, \mathbf{q} \in [\mathbf{K}_n]$, be a rational sequence and let $\mathbf{a} = (a_n)_{n\geq 0}$ be an arbitrary $\mathbf{K}$-sequence. Now we define the product $\mathbf{r} \cdot \mathbf{a}$ to be the equivalence class (w.r.t. $\sim_\infty$) of the sequences that agree with $p(n)/q(n) \cdot a_n$ from some index n_0 on. Hence we can consider $\mathbf{K}^{\mathbf{N}}/\!\sim_\infty$ to be a vector space over $(\mathbf{K}_n)$. Subsequently we will use the symbol $[\mathbf{a}]$ to denote the equivalence class w.r.t. $\sim_\infty$ of a sequence $\mathbf{a}$.

Due to the similarities in the definitions of holonomic functions and holonomic sequences, the following result comes without surprise. Indeed this theorem [Sta80] as well as its proof are quite similar to Theorem 1.2.1.

Theorem 1.3.1 Let E be the *shift operator* on sequences, i.e., $Ea_n := a_{n+1}$. For $\mathbf{a} := (a_n)_{n\geq 0} \in \mathbf{K}^{\mathbf{N}}$ the following three conditions are equivalent.

(i) There are $d \in \mathbf{N}$ and polynomials $q, p_0, p_1, \ldots, p_d$ in $\mathbf{K}[x]$, $p_d \neq 0$, such that for all $n \in \mathbf{N}$:

$$q(n) + p_0(n)a_n + p_1(n)a_{n+1} + \cdots + p_d(n)a_{n+d} = 0. \tag{1.3.2}$$

(ii) $(a_n)_{n\geq 0}$ is holonomic.

(iii) The linear space spanned by $\{[E^k\mathbf{a}] | k \in \mathbf{N}\}$ is a finite dimensional subspace of $\mathbf{K}^{\mathbf{N}}/_{\sim_\infty}$ over $(\mathbf{K}_n)$.

Proof.

$(i) \Rightarrow (ii)$. Assume that $(a_n)_{n\geq 0}$ satisfies the recurrence (1.3.2). Shifting this equation by step 1 gives, say $E(1.3.2)$, which has the inhomogenous part $q(n+1)$. The homogenous recurrence equation $q(n+1)(1.3.2) - q(n)E(1.3.2)$, which is of order $d+1$, is satisfied by $(a_n)_{n\geq 0}$.

$(ii) \Rightarrow (iii)$. Suppose $\mathbf{a}$ satisfies (1.3.1). We choose $n_0 \in \mathbf{N}$ as follows: If p_d has nonnegative integer roots we take $n_0 = \max\{n \in \mathbf{N} | p_d(n) = 0\} + 1$, otherwise let $n_0 = 0$. Now we get

$$a_{n+d} = \sum_{i=0}^{d-1} a_{n+i} \frac{-p_i(n)}{p_d(n)} \qquad \text{for all } n \geq n_0. \tag{1.3.3}$$

Hence $[E^d\mathbf{a}]$ lies in $L := \langle [\mathbf{a}], [E\mathbf{a}], \ldots, [E^{d-1}\mathbf{a}] \rangle$. By repeated shifts of (1.3.1) and substitutions according to (1.3.3) we find that for each $k \in \mathbf{N}$, $[E^{d+k}\mathbf{a}]$ lies in L and we get $\langle [\mathbf{a}], [E\mathbf{a}], [E^2\mathbf{a}], \ldots \rangle = L$, which has finite dimension.

$(iii) \Rightarrow (i)$. If $\langle [E^k\mathbf{a}] | k \in \mathbf{N} \rangle$ has finite dimension, then (by linear dependency) there is an $n_0 \in \mathbf{N}$ such that (1.3.2) with $q = 0$ holds for all $n \geq n_0$. Multiply all polynomials $p_i(n)$ by $n(n-1)\cdots(n-n_0+1)$ and we get an equation of this type that holds for all $n \in \mathbf{N}$. □

If $q \neq 0$, equation (1.3.2) will be called an *inhomogenous* holonomic recurrence.

Holonomic sequences have the nice property that they can be represented by a finite amount of information, i.e., if we have a holonomic recurrence of type (1.3.2) and a set of initial values $\{a_0, a_1, \ldots, a_{n_0}\}$ we can compute the elements in the sequence with little effort (and storage) by just applying the recurrence. The index n_0 up to which the initial values must be supplied, depends on the order d of the recurrence and on the maximum of the nonnegative integer roots of the polynomial p_d: If $p_d(n)$ is not equal to zero for all $n \in \mathbf{N}$, then $n_0 = d - 1$ is sufficient. Otherwise we have to take $n_0 = \max\{n \in \mathbf{N} | p_d(n) = 0\} + d$.

We turn to some well known examples of sequences that are holonomic.

Example 1.3.1

1. Let $\mathbf{K} = \mathbf{Q}$. The *Fibonacci numbers* F_n are recursively defined by

$$F_{n+1} = F_n + F_{n-1} \quad \text{for } n \geq 1, \qquad F_0 = 0, \quad F_1 = 1. \tag{1.3.4}$$

In fact this recursive definition is a holonomic recurrence equation of order 2. If all polynomials $p_i(n)$ in a holonomic recurrence of type (1.3.1) are constants, as in this case, we call the recurrence as well as the corresponding sequence *C-finite* or *C-recursive* [NP95].

2. A sequence $(t_n)_{n\geq 0}$ is *hypergeometric* if and only if there exists a rational function $r(x) \in \mathbf{K}(x)$ such that for all $n \in \mathbf{N}$

$$\frac{t_{n+1}}{t_n} = r(n). \tag{1.3.5}$$

   Hypergeometric sequences are the class of sequences that satisfy a first order holonomic recurrence. A simple example of a hypergeometric sequence are the factorials $n!$. For this sequence $r(n) = n + 1$.

3. Let $\mathbf{K} = \mathbf{C}(\alpha, x)$. The *Gegenbauer polynomials* $C_n^\alpha(x)$ are defined by the recurrence

$$-(n+2\alpha)C_n^\alpha(x) + 2(n+\alpha+1)xC_{n+1}^\alpha(x) - (n+2)C_{n+2}^\alpha(x) = 0 \tag{1.3.6}$$

   with the initial values $C_0^\alpha(x) = 1$ and $C_1^\alpha(x) = 2\alpha x$. Hence the sequence of the Gegenbauer polynomials $(C_n^\alpha(x))_{n\geq 0}$ is holonomic.

$\square$

A lot of sequences we are faced with in "real" life are holonomic. But not all of them, as the example of the *Bernoulli numbers*, which are well known to be not holonomic[1], shows.

1.4 Closure Properties

Examples 1.2.1.3 and 1.3.1.3 have shown that both the sequence of the Gegenbauer polynomials and their generating function are holonomic.

At the beginning of this section we make explicit that there is a one-to-one correspondence (already observed by [Jun31, p.299]) between holonomic sequences and their generating functions. Moreover, we will see that it is possible to "compute" this correspondence, i.e., given a holonomic sequence $(a_n)_{n\geq 0}$ via a recurrence one can compute a holonomic differential equation satisfied by the generating function $f(x) = \sum_{n\geq 0} a_n x^n$, and vice versa.

Theorem 1.4.1 A formal power series $f(x) = \sum_{n\geq 0} a_n x^n \in \mathbf{K}[[x]]$ is holonomic if and only if $(a_n)_{n\geq 0}$ is holonomic.

[1] Let B_n denote the nth Bernoulli number. The exponential generating function $\hat{f}(x) = \sum_{n\geq 0} B_n \frac{x^n}{n!}$ is given by $\hat{f}(x) = x/(e^x - 1)$ (see [GKP94, p. 285]). An induction argument that is similar to the one in the proof that $\tan(x)$ is not holonomic on page 9, tells us that $\hat{f}(x)$ is not holonomic. Applying some results of the following section (Theorem 1.4.1, Theorem 1.4.3) proofs the non-holonomicity of the Bernoulli numbers.

Proof. Assume that $f(x) = \sum_{n\geq 0} a_n x^n$ is holonomic. We can easily check that

$$x^k f^{(j)} = x^k \sum_{n\geq 0} (n+1)^{\overline{j}} a_{n+j} x^n = \sum_{n\geq k} (n+1-k)^{\overline{j}} a_{n+j-k} x^n.$$

We transform (1.2.1) according to this relation and, by equating coefficients of same powers of x on both sides of the resulting equation, we get a linear recurrence equation with polynomial coefficients, satisfied by $(a_n)_{n\geq 0}$. An appropriate shift of this recurrence results in an equation of type (1.3.1).

Suppose the sequence $(a_n)_{n\geq 0}$ satisfies the holonomic recurrence (1.3.1). Multiply this equation by x^n and sum over all $n \geq 0$. Let θ be the operator defined by

$$\theta f = x \frac{\mathrm{d}f}{dx}.$$

If we replace the terms $n^k a_{n+j}$ according to the transformation rule

$$\sum_{n\geq 0} n^k a_{n+j} x^n = \theta^k \sum_{n\geq 0} a_{n+j} x^n = \theta^k \left(\frac{f - a_0 - a_1 x - \cdots - a_{j-1} x^{j-1}}{x^j} \right)$$

we get an inhomogenous linear differential equation with coefficients that are rational functions in x. Clearing the denominator results in a (possibly inhomogenous) holonomic differential equation, satisfied by f. □

Assume a holonomic power series $f(x) = \sum_{n\geq 0} a_n x^n$ is given by a holonomic differential equation of order d and degree k. Following the proof of Theorem 1.4.1, it is easy to see that the sequence $(a_n)_{n\geq 0}$ satisfies a holonomic recurrence of order $\leq d + k$ and degree $\leq d$.

Conversely, if $(a_n)_{n\geq 0}$ is a solution of a holonomic recurrence of order d and degree k, then its generating function satisfies an (inhomogenous) holonomic differential equation of order $\leq k$ and degree $\leq d + k$.

Theorem 1.4.1 makes it easy to establish connections between some of the functions and sequences in Example 1.2.1 and Example 1.3.1.

Example 1.4.1

1. (Cf. Example 1.2.1.1 and Example 1.3.1.1.) Let $(a_n)_{n\geq 0}$ be a C-finite sequence over $\mathbf{K}$, i.e., $(a_n)_{n\geq 0}$ satisfies a linear recurrence with constant coefficients

$$c_0 a_n + c_1 a_{n+1} + \cdots + c_d a_{n+d} = 0. \tag{1.4.1}$$

According to the rules presented in the proof of Theorem 1.4.1, this recurrence is transformed into an inhomogenous differential equation of order zero. It is evident that the solution of this equation is a rational function. Hence C-finite sequences have rational generating functions. A prominent example are the Fibonacci numbers F_n. Their generating function is given by

$$\sum_{n\geq 0} F_n x^n = \frac{x}{1-x-x^2}.$$

2. In Example 1.2.1 we mentioned that the function $f(x) = \sin(x)$ is holonomic, since it satisfies the differential equation

   $f + f'' = 0$. If $f(x) = \sum_{n\geq 0} a_n x^n$ then the sequence $(a_n)_{n\geq 0}$ is recursively defined by

   $$a_n + (n+1)(n+2)a_{n+2} = 0, \qquad a_0 = 0, \quad a_1 = 1.$$

   It is also possible to obtain a "closed" form for the coefficients a_n and we find that

   $$\sin(x) = \sum_{n\geq 0} \frac{(-1)^{2n+1}}{(2n+1)!} x^{2n+1}.$$

3. A *hypergeometric function* ${}_pF_q$ is a formal power series that is defined by

   $$ {}_pF_q\left(\begin{matrix} a_1, \ldots, a_p \\ b_1, \ldots, b_q \end{matrix} \middle| x \right) = \sum_{n\geq 0} \frac{a_1^{\overline{n}} \cdots a_p^{\overline{n}}}{b_1^{\overline{n}} \cdots b_q^{\overline{n}}} \frac{x^n}{n!}. \tag{1.4.2}$$

   (The rising factorial $n^{\overline{k}}$ is defined on page 5.) None of the lower parameters b_k is allowed to be a negative integer, otherwise we would get a division by zero.

   One can easily check (see [GKP94, Section 5.5]) that the coefficients t_n of x^n in the power series expansion of a hypergeometric function, which is given by (1.4.2), satisfy the recurrence

   $$r(n)t_n = s(n)t_{n+1}, \tag{1.4.3}$$

   where we can split the polynomials $r(n)$ and $s(n)$ into linear factors as follows

   $$r(n) = (n+a_1)(n+a_2)\cdots(n+a_p),$$
   $$s(n) = (n+b_1)(n+b_2)\cdots(n+b_q)(n+1).$$

   We will now derive a differential equation satisfied by the hypergeometric function F. (For a shorter notation we will often omit all the parameters). We start out by multiplying the left hand side of (1.4.3) by x^n and summing over all $n \geq 0$. Following the ideas presented in the proof of Theorem 1.4.1 we get

   $$\sum_{n\geq 0} (n+a_1)\cdots(n+a_p)\, t_n\, x^n = (\theta + a_1)\cdots(\theta+a_p)F.$$

Similar transformations of the right hand side result in

$$\sum_{n\geq 0}(n+b_1)\cdots(n+b_q)(n+1)\,t_{n+1}\,x^n =$$

$$x^{-1}\sum_{n\geq 0}(n+b_1-1)\cdots(n+b_q-1)\,n\,t_n\,x^n =$$

$$x^{-1}(\theta+b_1-1)\cdots(\theta+b_q-1)\theta F.$$

Hence a hypergeometric function F, as given by (1.4.2), satisfies the differential equation

$$x(\theta+a_1)\cdots(\theta+a_p)F = (\theta+b_1-1)\cdots(\theta+b_q-1)\,\theta F.$$

□

We want to mention here that Theorem 1.4.1, as a Theorem on formal power series, allows us to compute a differential equation that is satisfied by a power series, say $f(x)$, that analytically does not have a positive radius of convergence. For example, the series $f(x) = \sum_{n\geq 0} n!\,x^n$ does not converge if $x \neq 0$. Hence $f(x)$ does not correspond to a function in the analytical sense. However, as a formal power series, it can be defined to be the (unique) solution of the differential equation $-1+(1-x)f(x)-x^2f'(x) = 0$, with the initial condition $[x^0]f(x) = 1$. Note that sometimes we write $f(0)$ for $[x^0]f(x)$.

So far, we have seen that a holonomic (or P-recursive) sequence has a holonomic (or D-finite) generating function. Let us now consider the sequence of *Catalan numbers* C_n ([GKP94, p. 358–360]), which is recursively defined by

$$C_{n+1} = \sum_{k=0}^{n} C_k\,C_{n-k}, \qquad C_0 = 1. \tag{1.4.4}$$

If we are given the Catalan numbers up to index n, we have to perform $O(n)$ multiplications to compute C_{n+1}. Hence, the computation of C_n needs $O(n^2)$ operations, if we use recurrence (1.4.4). Let us see how the generating function $C(x) = \sum_{n\geq 0} C_n\,x^n$ helps us to compute the sequence more efficiently: For that multiply both sides of (1.4.4) with x^n and sum over all $n \geq 0$:

$$\sum_{n\geq 0} C_{n+1}\,x^n = \sum_{n\geq 0}\sum_{k=0}^{n} C_k C_{n-k}\,x^n$$

Since the right hand side is the Cauchy product $C(x)C(x)$, we get

$$(C(x)-1)/x = C^2(x),$$

which results in the quadratic equation

$$xC^2(x) - C(x) + 1 = 0. \tag{1.4.5}$$

The fact that $C(x)$ satisfies a quadratic equation does not really help us now; however, after we have proven *constructively* the following theorem—already given by Comtet [Com64] in the 60's—this will help a lot.

Theorem 1.4.2 If $f(x) \in \mathbf{K}[[x]]$ is algebraic, then $f(x)$ is holonomic.

Proof. Assume f satisfies the algebraic equation

$$p_0(x) + p_1(x)f(x) + \cdots + p_d(x)f^d(x) = 0, \tag{1.4.6}$$

where $p_i \in \mathbf{K}[x]$ $(0 \leq i \leq d)$, $p_d(x) \neq 0$. Let the symbol $A(f,x)$—sometimes written without the indeterminates f and x—denote the polynomial on the left hand side of (1.4.6). Without loss of generality we may assume that $A(f,x)$ is squarefree. We define two polynomials B and C by

$$B(f,x) = -\frac{\partial A(f,x)}{\partial x} \qquad \text{and} \qquad C(f,x) = \frac{\partial A(f,x)}{\partial f},$$

and get $f'(x) = B(f,x)/C(f,x)$.

The polynomial A is squarefree, hence A and C are relatively prime (w.r.t. f) and the extended Euclidean algorithm (with respect to f) gives polynomials $S(f,x)$, $T(f,x)$ and $g(x)$ such that $S(f,x)\,A(f,x) + T(f,x)\,C(f,x) = g(x)$. Now we get

$$f'(x) = \frac{B(f,x)\,T(f,x)}{C(f,x)\,T(f,x)} = \frac{B(f,x)\,T(f,x)}{g(x) - A(f,x)\,S(f,x)} = \frac{B(f,x)\,T(f,x)}{g(x)}. \tag{1.4.7}$$

(Note, that the last equality holds, since $A(f,x) = 0$.)

It is clear that f' and, by further differentiation and substitution of f' according to (1.4.7), also higher derivatives of f can be expressed by a linear combination of $\{1, f, f^2, \ldots\}$. Hence

$$\dim\langle 1, f, f', f'', \ldots\rangle \leq \dim\langle 1, f, f^2, \ldots\rangle = \dim\langle 1, f, f^2, \ldots, f^{d-1}\rangle \leq d$$

over $\mathbf{K}(x)$. Now, by Theorem 1.2.1, f is holonomic. $\square$

Remark. If we are given a function by an algebraic equation, the proof above can be used in a constructive way. This means, it also gives a method to compute a holonomic differential equation, that is satisfied by the function: We have to compute the functions $1, f, f', f'', \ldots, f^{(d-1)}$ as linear combinations of $1, f, f^2, \ldots, f^{d-1}$. This should be done by observing that, for arbitrary k, $f^{(k)}(x) = R_k(f,x) = \overline{R}_k(f,x)$, where $R_k(f,x)$ and

$\overline{R}_k(f,x)$ are polynomial in f and rational in x, and $\overline{R}(f,x) = R(f,x) \bmod A(f,x)$. (The residues $\overline{R}$ are computed with respect to f.)

Substituting these linear combinations into an inhomogenous holonomic equation of the form

$$q(x) + p_0(x)f(x) + p_1(x)f'(x) + \cdots + p_{d-1}(x)f^{(d-1)}(x) = 0$$

and equating all powers of f to zero, results in a linear system of equations (with indeterminates $q, p_0, p_2, \ldots, p_{d-1}$), that is guaranteed to have a non trivial solution. $\square$

We apply this method to the the function $C(x)$, the generating function of the Catalan numbers C_n. From (1.4.5) we deduce that

$$C' = \frac{C^2}{1 - 2xC}.$$

By extended gcd computations we obtain

$$(1 - 4x - 2xC)\left(xC^2 - C + 1\right) + (-1 + 2x)C + x^2C)\left(-1 + 2xC\right) = 1 - 4x$$

and we get

$$C' = \frac{C^2\left((-1+2x)C + x^2C\right)}{1-4x} \bmod \left(xC^2 - C + 1\right) = \frac{-1 + C(1-2x)}{x(-1+4x)}.$$

Now we have to solve the equation $r(x) + p_0(x)C(x) + p_1(x)C'(x) = 0$, which can be written as

$$r - \frac{p_1}{x(-1+4x)} + f\left(p_0 + \frac{p_1(1-2x)}{x(-1+4x)}\right) = 0,$$

for the rational functions $r(x), p_0(x), p_1(x)$. Equating all coefficients of same powers of f with zero, results in the differential equation

$$-1 + (1-2x)C(x) + x(1-4x)C'(x) = 0.$$

Following Theorem 1.4.1, we can transform this differential equation into a recurrence that is satisfied by the sequence of Catalan numbers:

$$2(1+2n)C_n - (2+n)C_{n+1} = 0, \qquad C_0 = 1 \tag{1.4.8}$$

This recurrence makes it possible to compute C_n by performing $\mathrm{O}(n)$ operations, whereas the computation of the nth Catalan number via (1.4.4) requires $\mathrm{O}(n^2)$ operations.

Moreover, equation (1.4.8) delivers the information that the sequence $(C_n)_{n\geq 0}$ is hypergeometric. We have

$$\frac{C_{n+1}}{C_n} = \frac{2(1+2n)}{2+n}$$

and, remembering Example 1.4.1.3, it follows that

$$C_n = \frac{4^n (\frac{1}{2})^{\overline{n}}}{(n+1)!} = \binom{2n}{n} \frac{1}{n+1}.$$

Remark. We want to note that, due to the fact that the generating function of the Catalan numbers satisfies a quadratic equation, the closed form representation for C_n could also be derived by other means. See, for instance, [GKP94, p. 358].

After we have proved that all algebraic functions are holonomic, we are going to show that some elementary (binary) operations on sequences and their generating functions preserve holonomicity. (Cf. for instance [Sta80].) In fact, the following theorem contains the main reason, why we work with holonomic functions and sequences.

Theorem 1.4.3 Let $f(x) = \sum_{n\geq 0} a_n x^n$ and $g(x) = \sum_{n\geq 0} b_n x^n$ be holonomic power series. Then

(a) $f(x) + g(x) = \sum_{n\geq 0}(a_n + b_n)x^n$, (sum)

(b) $f(x)\, g(x) = \sum_{n\geq 0}(\sum_{k=0}^{n} a_k b_{n-k})x^n$ (convolution, Cauchy product)

(c) $f(x) * g(x) := \sum_{n\geq 0} a_n b_n x^n$, (termwise or Hadamard product)

are holonomic power series.

Proof. We start the proof of the closure properties with the following observation: Suppose $f(x)$ satisfies a (possibly inhomogenous) holonomic differential equation of order d, hence we can express $f^{(d)}$ as a linear combination of $1, f, f', \ldots, f^{(d-1)}$:

$$f^{(d)} = p_d(x) + q_{d,0}(x)f + q_{d,1}(x)f' + \cdots + q_{d,d-1}(x)f^{(d-1)} \tag{1.4.9}$$

with $p_d, q_{d,j} \in \mathbf{K}(x)$. We differentiate (1.4.9) with respect to x and replace the derivative $f^{(d)}$ on the right hand side of the resulting equation according to (1.4.9) in order to express $f^{(d+1)}$ in terms of $1, f, f' \ldots, f^{(d-1)}$. Hence, given $k \in \mathbf{N}$, repeated differentiations and substitutions make it possible to find rational functions $p_{k,j}, q_k \in \mathbf{K}(x)$ such that $f^{(k)}(x) = q_k(x) + \sum_{j=0}^{d-1} p_{k,j}(x) f^{(j)}(x)$.

Given a sequence $(a_n)_{n\geq 0}$ that satisfies a holonomic recurrence of the form (1.3.2), let n_0 be the integer $n_0 = \max\{n \in \mathbf{N} | p_d(n) = 0\} + 1$ if p_d has nonnegative integer roots. Otherwise we set $n_0 = 0$. For any $k \in \mathbf{N}$, by repeated shifts and substitutions we find rational functions $p_{k,j}, q_k \in \mathbf{K}(x)$ such that for all $n \geq n_0$, $a_{n+k} = q_k(n) + \sum_{j=0}^{d-1} p_{k,j}(n) a_{n+j}$.

Now, let f and g satisfy (inhomogenous) holonomic differential equations of orders d_1 and d_2, respectively. To prove (a) and (b), we assume that for every $k \in \mathbf{N}$ the rational functions $p_{k,j}$, q_k, $r_{k,j}$, s_k have the property that

$$f^{(k)} = q_k + \sum_{j=0}^{d_1-1} p_{k,j} f^{(j)} \qquad \text{and} \qquad g^{(k)} = s_k + \sum_{j=0}^{d_2-1} r_{k,j} g^{(j)},$$

where we omitted the function variable x.

(a) Let $h = f + g$ and $k \in \mathbf{N}$, then we can express $h^{(k)}$ as a linear combination of $1, f, \ldots, f^{(d_1-1)}, g, \ldots, g^{(d_2-1)}$, i.e.,

$$h^{(k)} = f^{(k)} + g^{(k)} = q_k + s_k + \sum_{j=0}^{d_1-1} p_{k,j} f^{(j)} + \sum_{j=0}^{d_2-1} r_{k,j} g^{(j)}. \tag{1.4.10}$$

Now we search for the inhomogenous differential equation

$$t + u_0 h + u_1 h' + \cdots + u_d h^{(d)} = 0. \tag{1.4.11}$$

We plug (1.4.10) into (1.4.11) and get a homogenous system of d_1+d_2+1 linear equations in the $d+2$ variables $t, u_0, \ldots, u_d$. Hence, if we take $d = d_1 + d_2$, we can be sure that this linear system has a nontrivial solution.

(b) For $h = fg$ and $k \in \mathbf{N}$ we get (by the Leibniz rule)

$$h^{(k)} = \sum_{i=0}^{k} \binom{k}{i} f^{(i)} g^{(k-i)} = \sum_{i=0}^{k} \binom{k}{i} \Big(q_i + \sum_{j=0}^{d_1-1} p_{i,j} f^{(j)}\Big)\Big(s_{k-i} + \sum_{j=0}^{d_2-1} r_{k-i,j} g^{(j)}\Big), \tag{1.4.12}$$

which tells us that we can write $h^{(k)}$ as a linear combination of $(d_1+1)(d_2+1)$ functions. Therefore, a non trivial differential equation of type (1.4.11) exists, if we take $d = (d_1+1)(d_2+1) - 1$.

(c) Let $c_n = a_n b_n$. Since $c_{n+k} = a_{n+k} b_{n+k}$ for every $k \in \mathbf{N}$, an argument that is similar to the one applied in the proof of part (b), but which uses shifts instead of derivatives, tells us that $(c_n)_{n\geq 0}$ is a holonomic sequence, hence $\sum_{n\geq 0} c_n x^n$ is holonomic. $\square$

Remark. We will use the terminology: sum, Cauchy product (convolution) and Hadamard (or termwise) product both for sequences and for functions.

The proof of the previous theorem is constructive, i.e., given two holonomic functions (sequences) we can apply the presented proof to obtain the holonomic differential (recurrence) equation that is satisfied by their sum, Cauchy or Hadamard product. A short example serves as an illustration.

Example 1.4.2 Let $f(x) = \sin(x)$, $g(x) = \arctan(x)$. Following the proof of Theorem 1.4.3, we will derive a holonomic differential equation that is satisfied by $h(x) = f(x) + g(x)$:

Step1. (Find an upper bound for the order of the desired differential equation.)
We know that f and g are solutions of the differential equations

$$f + f'' = 0 \qquad \text{and}$$

$$1 - (1 + x^2)\, g' = 0,$$

respectively. Thus, h satisfies a holonomic differential equation of order 3.

Step2. (Express f, f', f'', f''' in terms of $1, f, f'$, and g, g', g'', g''' in terms of $1, g$.)
If we repeatedly differentiate and substitute according to the two differential equations that are satisfied by f and g, we get:

$$\begin{aligned} f &= f & g &= g \\ f' &= f' & g' &= \frac{1}{1+x^2} \\ f'' &= -f & g'' &= \frac{-2x}{(1+x^2)^2} \\ f''' &= -f' & g''' &= \frac{-2+6x^2}{(1+x^2)^3} \end{aligned}$$

Step3. (Solve the corresponding linear system.)
The search for a differential equation of the form

$$q(x) + p_0(x)h(x) + p_1(x)h'(x) + p_2(x)h''(x) + p_3(x)h'''(x) = 0$$

gives the linear system

$$\begin{pmatrix} 1 & 0 & \frac{1}{1+x^2} & \frac{-2x}{(1+x^2)^2} & \frac{-2+6x^2}{(1+x^2)^3} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} q \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} = 0. \tag{1.4.13}$$

We solve (1.4.13) and find that $h(x)$ is a solution of the differential equation

$$(1 - 8x^2 + 4x^4) + (1 + x^2)^3 h'(x) + (1 + x^2)^3 h'''(x) = 0.$$

□

We have seen that one step in the computation of a holonomic (recurrence or differential) equation, satisfied by a sequence or function that is built from other holonomic sequences or functions via holonomicity preserving operations, is to solve a homogenous system of linear equations. In other words, we have to compute the Nullspace of a certain matrix A, whose entries are rational functions. We assume now that A has $d+1$ rows and $d+2$ columns. Every element in the Nullspace of A corresponds to a (possibly inhomogenous) holonomic equation, whose order is less or equal d.

If the Nullspace of A has dimension, say k, with $k > 1$, the resulting (recurrence or differential) equation is not unique. (We regard two holonomic equations to be the same if they coincide up to a common factor.) In this case, an element $\mathbf{p} = (q, p_0, p_1, \ldots, p_d)$ in the Nullspace of A can be found, where the last $k-1$ components of $\mathbf{p}$ are all equal 0. Then $\mathbf{p}$ gives a holonomic equation of order less or equal $d-k+1$. Since $\mathbf{p}$ is a linear combination of the basis elements of the Nullspace of A, it can be computed by again solving a system of linear equations.

We also want to mention here that in the process of solving a linear system (for instance by Gaussian elimination), the coefficients of the elements in the solution space grow rather fast: Assume that we are given two sequences $\mathbf{a} = (a_n)_{n\geq 0}$ and $\mathbf{b} = (b_n)_{n\geq 0}$, where $\mathbf{a}$ satisfies a homogenous recurrence of order d_1 and degree k_1 and $\mathbf{b}$ is the solution of a homogenous recurrence of order d_2 and degree k_2. We know that $\mathbf{c} = \mathbf{a}+\mathbf{b}$ satisfies a recurrence equation of order d_1+d_2. Based on experimental results the author conjectures that the degree of a recurrence that is satisfied by $\mathbf{c}$ raises up to $k_1(d_2+1)+k_2(d_1+1)$ in the worst case. If we compute a recurrence for the termwise product of $\mathbf{a}$ and $\mathbf{b}$, it seems as if the result, which has order less or equal $d_1 d_2$, may be of degree up to $(1+(d_1-1)(d_2-1))(d_2 k_1 + d_1 k_2)$. (The same bounds were observed in the case of differential equations.) Although we did not prove these conjectures, it is evident that computations with holonomic objects that involve sums and products (and possibly other operations) quickly become extremely time consuming as the number of these operations increases.

Note also that the closure properties in the theorem above do not include fractions of holonomic functions or sequences. As we have seen (page 9) the function $\tan(x) = \sin(x)/\cos(x)$ is not holonomic, though both $\sin(x)$ and $\cos(x)$ are.

Part (a) of the closure properties (or its proof) tells us how to compute a differential equation that is satisfied by $f(x)+g(x)$, the sum of arbitrary solutions of two given holonomic differential equations. If these two equations are homogenous, it is clear that any linear combination of f and g (over $\mathbf{K}$) satisfies the computed equation. In particular $f-g$ has this property, and, by checking a finite number of initial values of $f-g$ to be zero, we can "prove" whether these two functions are identical in the ring of formal power series.

The same result holds for sequences. However, certain roots in the leading polynomial of a holonomic differential or recurrence equation may cause problems, if we want to perform zero recognition. This subject is discussed in more detail in Chapter 3.

Indeed, the property that the families of holonomic functions and sequences are algebras (with respect to addition and multiplication) together with the fact that zero recognition can be done, is the most important strength of holonomicity. It is thus possible to prove identities like $\sin^2(x) + \cos^2(x) = 1$ or Cassini's identity for the Fibonacci numbers $F_{n+1}F_{n-1} - F_n^2 = (-1)^n$ "automatically". (In real life, most identities will not look that nice.)

Let $\mathbf{K} = \mathbf{C}$. We know that the functions $\sin(x)$, $\cos(x)$ are holonomic. Applying Theorem 1.4.3, we immediately deduce that for a constant $a \in \mathbf{C}$ the function $\sin(x)\cos(a) + \cos(x)\sin(a)$, which is identical to $\sin(x + a)$, is holonomic. The following theorem delivers the holonomicity of $\sin(x + a)$ (and many more functions) without requiring the knowledge of this identity.

Theorem 1.4.4 Let $g(x) \in \mathbf{K}[[x]]$ be algebraic, let $f(x) \in \mathbf{K}[[x]]$ be holonomic. If $h(x) = f(g(x)) \in \mathbf{K}[[x]]$, then $h(x)$ is holonomic.

A theoretical argument for the holonomicity of $h(x)$ was given by Stanley [Sta80]. Here, we want to present a more constructive proof [SZ94]:

Proof.

Let $g(x)$ satisfy the squarefree algebraic equation $A(g, x) = 0$ with $A \in \mathbf{K}[x_1, x_2]$ and $deg_g(A) = d_1$. We suppose that f satisfies a homogenous holonomic differential equation of order d_2:

$$q_0(x)f(x) + q_1(x)f'(x) + \cdots + q_{d_2}(x)f^{(d_2)}(x) = 0. \tag{1.4.14}$$

Now define $G(g, x) = \gcd(A(g, x), q_{d_2}(g))$ (w.r.t. g).

We start out by giving an argument that, without loss of generality, we may assume that $G(g, x) = 1$: It is clear that $deg_x(G) \leq \deg_x(q_{d_2}(g)) = 0$. Now let $A = G \cdot T$; we know that $g(x)$ is a root of A, hence $G(g(x), x) = 0$ or $T(g(x), x) = 0$. If $g(x)$ is a root of G, then $g(x)$ is constant[2] and, if $f(g(x))$ is defined, then $h(x)$ is also constant and hence holonomic. In the case that $g(x)$ is a root of T, we observe that—note that A is squarefree—$\gcd(T, q_{d_2}(g)) = 1$ (w.r.t. g). Now we can continue the proof assuming that $g(x)$ is a root of a polynomial, say A, with $\deg_g(A) = d_1$ and, where A and $q_{d_2}(g)$ are relatively prime.

[2] Assume $g(x)$ satisfies the equation

$$c_0 + c_1 g + \cdots + c_k g^k = 0 \tag{1.4.15}$$

with $c_i \in \mathbf{K}$ $(0 \leq i \leq k)$. Differentiating (1.4.15) w.r.t x yields

$$(c_1 + 2c_2 g + \cdots + kc_k g^{k-1})g' = 0. \tag{1.4.16}$$

from (1.4.16) we deduce that $g(x)$ satisfies an algebraic equation of order $k - 1$ or that $g'(x) = 0$. Now, and induction argument proves that $g(x)$ is constant.

The chain rule tells us that for each $k \in \mathbf{N}$ there are polynomials $P_{k,j}$ such that

$$h^{(k)}(x) = \sum_{j=0}^{k} f^{(j)}(g)\ P_{k,j}(g, g', \ldots, g^{(k)}).$$

Following the proof of Theorem 1.4.2 we find functions $R_j(g,x)$ that are rational in x and polynomial in g of degree less than d_1 such that for every $j \in \mathbf{N}$ we have $g^{(j)}(x) = R_j(g,x)$. Hence we can write

$$h^{(k)}(x) = \sum_{j=0}^{k} f^{(j)}(g) B_{k,j}(g,x), \tag{1.4.17}$$

where $B_{k,j} \in \mathbf{K}(x)[g]$.

We compute $\overline{q}_j(g) = q_j(g) \bmod A$ (w.r.t. g) and get

$$f^{(d_2)}(g) = -\Big(\sum_{j=0}^{d_2-1} \overline{q}_j(g) f^{(j)}(g)\Big)/\overline{q}_{d_2}(g). \tag{1.4.18}$$

Now we have $G(g,x) = \gcd(A(g,x), q_{d_2}(g)) = \gcd(A(g,x), \overline{q}_{d_2}(g))$ (w.r.t. g).

Since $\deg_g(G) = 0$, we use the same argument as in the proof of Theorem 1.4.2 to get rid of g in the denominator of (1.4.18) and to obtain functions $C_{d_2,j}$ that are rational in x and polynomial in g (of degree less than d_1) such that

$$f^{(d_2)}(g) = C_{d_2,0}(g,x)f(g) + C_{d_2,1}(g,x)f'(g) + \cdots + C_{d_2,d_2-1}(g,x)f^{(d_2-1)}(g). \tag{1.4.19}$$

An induction argument makes it clear that—by repeatedly differentiating (1.4.14) and representing as in (1.4.19)—for each $k \in \mathbf{N}$ we can express $f^{(k)}$ as follows:

$$f^{(k)}(g) = C_{k,0}(g,x)f(g) + C_{k,1}(g,x)f'(g) + \cdots + C_{k,d_2-1}(g,x)f^{(d_2-1)}(g). \tag{1.4.20}$$

We put (1.4.17) and (1.4.20) together and see that

$$h^{(k)}(x) = \sum_{j=0}^{k} B_{k,j}(g,x) \sum_{i=0}^{d_2-1} C_{k,i}(g,x) f^{(i)}(g).$$

We reduce this expression, which is polynomial in g, modulo A. The result are rational functions $c_{k,j,i}$ in x such that

$$h^{(k)}(x) = \sum_{j=0}^{d_1-1} \sum_{i=0}^{d_2-1} c_{k,j,i}(x)\, g^j f^{(i)}. \tag{1.4.21}$$

Now we search for a holonomic equation of the form

$$q_0(x)h(x) + q_1(x)h'(x) + \cdots + q_d(x)h^{(d)}(x) = 0$$

by substituting according to (1.4.21) and equating the coefficients of $g^j f^{(i)}$ in this equation with zero, which gives a system of $d_1 d_2$ equations in $d+1$ variables. This linear system has a non trivial solution if $d = d_1 d_2$. □

The properties of holonomicity that we have seen in the previous theorems (and their proofs) in this section, enable us to solve the following combinatorial problem.

Example 1.4.3 (See [Wil90, p. 76].)
An *undirected 2-regular labeled graph* is a graph in which each vertex has exactly degree 2. Hence an undirected labeled 2-regular graph is a union of cycles, where each cycle has at least 3 vertices. Let g_n be the number of undirected 2-regular labeled graphs with n vertices. The exponential generating function $G(x)$ of the sequence $(g_n)_{n\geq 0}$ is known to be

$$G(x) = \sum_{n\geq 0} g_n \frac{x^n}{n!} = \frac{e^{-\frac{1}{2}x-\frac{1}{4}x^2}}{\sqrt{1-x}}.$$

Find a recurrence satisfied by g_n.

Considering the fact that the exponential function e^x is in our holonomic knowledge base, we can use the results of this section to solve this problem.

Step 1. (Compute a holonomic differential equation satisfied by $e^{-x/2-x^2/4}$.)
Let $h(x) = f(g(x))$ with $f(x) = e^x$ and $g(x) = -x/2 - x^2/4$. Following the proof of Theorem 1.4.4 we get

$$\begin{aligned} h &= f(g), \\ h' &= f(g)(-1-x)/2. \end{aligned}$$

The search for a holonomic differential equation of order one gives

$$p_0 h + p_1 h' = f(g)\big(p_0 - \frac{p_1}{2(1+x)}\big) = 0,$$

which results in

$$2\,h(x) + (1+x)\,h'(x) = 0.$$

Step 2. (Compute a holonomic differential equation satisfied by $1/\sqrt{1-x}$.)
Since $a(x) = 1/\sqrt{1-x}$ is a solution of the algebraic equation $1 - a^2(x)(1-x) = 0$, (by Theorem 1.4.2) we can find a holonomic equation that is satisfied by $a(x)$:

$$a(x) + 2(x-1)a'(x) = 0$$

Step 3. (Compute a holonomic differential equation satisfied by $e^{-x/2-x^2/4}/\sqrt{1-x}$.*)*
We apply Theorem 1.4.3 (b) and obtain a differential equation for the Cauchy product $G(x) = h(x)a(x)$:

$$x^2\, G(x) + 2(x-1)\, G'(x) = 0 \tag{1.4.22}$$

Step 4. (Compute a holonomic recurrence equation satisfied by $g_n/n!$.*)*
Let $\widehat{g}_n = g_n/n!$. We perform the transformations that are described in the proof of Theorem 1.4.1 for (1.4.22) and get

$$\widehat{g}_n + 2(n+2)\,\widehat{g}_{n+2} - 2(n+3)\,\widehat{g}_{n+3} = 0.$$

Step 5. (Compute a holonomic recurrence satisfied by g_n.*)*
Remembering the proof of Theorem 1.4.3 (c) we find a recurrence for the Hadamard product $\widehat{g}_n\, n!$:

$$(n+1)(n+2)\, g_n + 2(n+2)\, g_{n+2} - 2\, g_{n+3} = 0$$

□

It is important to note that the computations of each step in the previous example can be done by a computer algebra system. For example, the Maple package `gfun` by Salvy and Zimmermann [SZ94] is able to perform these (and many more) computations. With Mathematica, the package `GeneratingFunctions` which has been implemented by the author, does the calculations. This package is introduced in detail in Chapter 2. Another Mathematica package, called `RComp`, which was written by Nemes and Petkovšek [NP95], provides tools for manipulations of C-finite sequences.

We will now extend the holonomic closure to some unary operations on formal power series. To cause no confusion about integrals in fields like $\mathbf{C}$, we start out by defining the indefinite integral of a power series: Let $f(x) = \sum_{n\geq 0} a_n x^n \in \mathbf{K}[[x]]$, then

$$\int_0^x f(t)dt \quad := \quad \sum_{n\geq 1} \frac{a_{n-1}}{n} x^n. \tag{1.4.23}$$

Corollary 1.4.5 Let $f(x) = \sum_{n\geq 0} a_n x^n \in \mathbf{K}[[x]]$ be a holonomic power series. Then

(a) $\int_0^x f(t)dt$ (integral)

(b) $f'(x)$ (derivative)

are holonomic.

Proof. (See [KS94].) (a) Let $F(x) = \int_0^x f(t)dt$, and suppose f satisfies equation (1.2.2). Since $F'(x) = f(x)$, we have to replace $f^{(k)}(x)$ by $F^{(k+1)}(x)$ in this differential equation to get a holonomic equation that is satisfied by $F(x)$.

(b) Let f satisfy (1.2.2). If $p_0 = 0$, then $g(x) = f'(x)$ is a solution of the differential equation

$$q(x) + p_1(x)g(x) + p_2 g'(x) + \cdots + p_d(x)g^{(d-1)}(x) = 0.$$

If $p_0 \neq 0$, we can express f as a linear combination of $1, f', f'', \ldots, f^{(d)}$ in order to get a holonomic differential equation where the coefficient of f is 0. Then we can proceed as in the case that $p_0 = 0$. □

Let $f(x) = \sum_{n\geq 0} a_n x^n$. It is a well known (and easily verified) fact that

$$\sum_{n\geq 0} a_{2n} x^{2n} = \frac{f(x) + f(-x)}{2}.$$

Hence

$$\sum_{n\geq 0} a_{2n} x^{n} = \frac{f(\sqrt{x}) + f(-\sqrt{x})}{2},$$

and we immediately see that $(a_{2n})_{n\geq 0}$, the sequence of the even elements of $(a_n)_{n\geq 0}$, is holonomic (Theorem 1.4.3 and Theorem 1.4.4). Besides other propositions, the following corollary generalises this observation.

Corollary 1.4.6 Let $(a_n)_{n\geq 0}, (b_n^{(j)})_{n\geq 0} \quad (0 \leq j < m)$ be holonomic $\mathbf{K}$-sequences. If

(a) $c_n = a_{n+h}$, where $h \in \mathbf{N}$, or (shift)

(b) $c_n = \sum_{k=0}^{n} a_k$, or (partial sum)

(c) $c_n = \Delta a_n := a_{n+1} - a_n$, or (forward difference)

(d) $c_n = a_{dn+h}$, with $d, h \in \mathbf{N}$ or (subsequence, multisection)

(e) $c_n = b_q^{(r)}$ with $q \in \mathbf{N}$, $n = qm + r$ and $0 \leq r < m$, (interlacement)

then the $\mathbf{K}$-sequence $(c_n)_{n\geq 0}$ is holonomic.

Proof. (a) If $(a_n)_{n\geq 0}$ is the solution of a recurrence of type (1.3.2), then $c_n = a_{n+h}$ clearly satisfies

$$q(n+h) + p_0(n+h)c_n + p_1(n+h)c_{n+1} + \cdots + p_d(n+h)c_{n+d} = 0.$$

(b) Let $f(x) = \sum_{n\geq 0} a_n x^n$. If $c_n = \sum_{k\geq 0} a_k$, we get

$$g(x) = \sum_{n\geq 0} c_n x^n = \sum_{n\geq 0}\sum_{k=0}^{n} a_k x^n = \Big(\sum_{n\geq 0} a_n x^n\Big)\Big(\sum_{n\geq 0} x^n\Big) = \frac{f(x)}{1-x}.$$

By Theorem 1.4.3, $g(x)$ and also $(c_n)_{n\geq 0}$ are holonomic.

(c) Follows immediately from Theorem 1.4.3 and (a).

(d) Because of (a), we can assume $h = 0$. Let $f(x) = \sum_{n\geq 0} a_n x^n$ and assume that $\alpha \in \mathbf{C}$ is a dth root of unity[3], i.e.,

$$\alpha^p = 1 \Leftrightarrow p \equiv 0 \pmod{d}, \quad \text{for} \quad p \in \mathbf{N}. \tag{1.4.24}$$

We define

$$g(x) := \sum_{n\geq 0} b_n x^n := \frac{1}{d}\sum_{j=1}^{d} f(\alpha^j x).$$

By definition we have

$$b_n = \frac{a_n}{d}\left(\alpha^n + \alpha^{2n} + \cdots + \alpha^{dn}\right).$$

If $n \equiv 0 \pmod{d}$ we find that $b_n = a_n$.

If $n \equiv p \pmod{d}$ $(0 < p < d)$, we get $b_n = a_n s/d$, where $s = \alpha^p + \alpha^{2p} + \cdots + \alpha^{dp}$. Since $\alpha^{pd} = 1$, we observe that $s = \alpha^p(1 + \alpha^p + \cdots + \alpha^{p(d-1)}) = \alpha^p s$. With (1.4.24) we have $\alpha^p \neq 1$ and hence $s = 0$ and also $b_n = 0$.

Thus we have shown that $g(x) = \sum_{n\geq 0} a_{dn} x^{dn}$. Let

$$h(x) = g(x^{1/d}) = \sum_{n\geq 0} a_{dn} x^n = \sum_{n\geq 0} c_n x^n.$$

By Theorem 1.4.3 and Theorem 1.4.4, both $h(x)$ and $(c_n)_{n\geq 0}$ are holonomic.

(e) Assume $f_j(x) = \sum_{n\geq 0} b_n^{(j)} x^n$ for $(0 \leq j < m)$ and $g(x) = \sum_{n\geq 0} c_n x^n$. Now we obtain

$$g(x) = f_0(x^m) + x f_1(x^m) + \cdots + x^{m-1} f_{m-1}(x^m),$$

which is holonomic by Theorem 1.4.4 and 1.4.3. □

Remark. The proof of Corollary 1.4.6(d) tells us to compute the multisection $c_n = a_{nd}$ the following way: If $f(x) = \sum_{n\geq 0} a_n x^n$ and $h(x) = \sum_{n\geq 0} c_n x^n$, we get

$$h(x) = \frac{1}{d}\sum_{j=1}^{d} f(a_j(x)),$$

where the algebraic functions $a_j(x)$ satisfy ${a_j}^d(x) - x = 0$ $(1 \leq j \leq d)$. Hence $h(x)$ is the linear combination of d functions $f(a_j(x))$ $(1 \leq j \leq d)$, all of which satisfy a common differential equation, which can be computed if we follow the proof of Theorem 1.4.4. It

[3] As we will see in the remark, following this proof, it is not necessary that $\alpha \in \mathbf{K}$.

is clear that $h(x)$ is also a solution of this differential equation, and we need not compute a differential equation that is satisfied by the sum of d holonomic functions.

However, experimental results have shown that it is not advisable to do multisection of a sequence via algebraic substitution. A more efficient method would be as follows:

Suppose the sequence $(a_n)_{n\geq 0}$ is a solution of a recurrence of order d_1:

$$q(n) + p_0(n)a_n + p_1(n)a_{n+1} + \cdots + p_{d_1}(n)a_{n+d_1} = 0$$

Thus we also have

$$q(nd) + p_0(nd)a_{nd} + p_1(nd)a_{nd+1} + \cdots + p_{d_1}(nd)a_{nd+d_1} = 0. \tag{1.4.25}$$

Given $k \in \mathbf{N}$, we can use (1.4.25) to compute rational functions $r_{k,j}, s_k$ such that

$$a_{(n+k)d} = a_{nd+dk} = s_k(n) + \sum_{j=0}^{d_1-1} r_{k,j}(n)a_{nd+j}.$$

Now, the search for an inhomogenous holonomic recurrence of order d_1, that is satisfied by $(c_n)_{n\geq 0}$ with $c_n = a_{nd}$, corresponds to solving a system of $d_1 + 1$ homogenous linear equations in $d_1 + 2$ variables. This system has a nontrivial solution. □

Example 1.4.4 (by Herta T. Freitag, [Rab93, p. 212])
Let F_n be the nth Fibonacci number (page 11). L_n denotes the nth *Lucas number*, which is recursively defined by $L_{n+2} = L_{n+1} + L_n$ with $L_0 = 2$, and $L_1 = 1$.
Prove that for all positive integers n

$$\sum_{k=1}^{2n} F_{5k+1}L_{5k} \equiv 0 \pmod 5. \tag{1.4.26}$$

Via multisection we find that

$$F_{5(n+2)+1} = 11F_{5(n+1)+1} + F_{5n+1} \qquad \text{and} \qquad L_{5(n+2)} = 11L_{5(n+1)} + L_{5n}.$$

Now, the Hadamard product $G_n = F_{5n+1}L_{5n}$ satisfies

$$G_{n+3} = 122G_{n+2} + 122G_{n+1} - G_n,$$

and going via a differential equation, satisfied by the generating function $\sum_{n\geq 0} G_n x^n$ (see proof of Corollary 1.4.6(b)), we see that $H_n = \sum_{k=1}^{n} F_{5k+1}L_{5k}$ is a solution of the recurrence

$$H_{n+3} = 123H_{n+3} + 123H_{n+1} - H_n,$$

and we get for the subsequence H_{2n}

$$H_{2(n+3)} = 15128H_{2(n+2)} - 15128H_{2(n+1)} + H_{2n}.$$

This recurrence and the initial conditions $H_0 = 0$, $H_2 = 11035$ and $H_4 = 166937445$ prove (1.4.26). □

In Chapter 2 we will apply the theory discussed in this chapter to a lot of "real life" examples. We will see that a number of problems concerning identities between functions and sequences, summation, combinatorial enumeration, a.s.o., are covered by the holonomic universe. Therefore they can be solved almost completely "automatically", i.e., the computer does the routine jobs and we can concentrate on the creative parts.

1.5 The Multivariate Case

Zeilberger [Zei82] generalised the concept of univariate holonomic functions and sequences to the multivariate case. However, his definition does not preserve the one-to-one correspondence between holonomic (multivariate) sequences and their (multivariate) generating functions. Lipshitz [Lip89] pointed at this shortcoming and cured this deficiency by adjusting the definition of holonomic (or in his notation P-recursive) sequences in several variables appropriately.

Holonomic systems are obtained, if we consider multivariate functions of one or more continuous and/or discrete variables. (An instance of a holonomic system are the Gegenbauer polynomials $C_n^\alpha(x)$ given in Example 1.3.1).

Since this thesis concentrates on holonomic univariate functions, we omit exact definitions of multivariate holonomicity. Detailed descriptions and discussions were given, for example, by Chyzak [Chy94], Gessel [Ges90], Lipschitz [Lip89] and Zeilberger [Zei90].

Chapter 2

My *Mathematica* Package

2.1 Introduction

In Chapter 1 we discussed some properties of holonomic sequences and their generating functions. A closer look at the proofs of these properties reveals the fact that these proofs contain algorithms to compute holonomic differential or recurrence equations for functions or sequences that are built from other holonomic functions or sequences via holonomicity preserving operations. For example, if two holonomic functions are given by differential equations they satisfy, a holonomic differential equation for their sum can be computed, if we follow Theorem 1.4.3 and its proof. Considering also the property that holonomic functions and sequences are completely determined by a finite amount of information, i.e., a differential or recurrence equation together with finitely many initial values, it is evident that we have a machinery consisting of systematic methods to handle manipulations of holonomic functions and sequences.

Since it is possible to perform zero recognition[1] in the holonomic universe, we can automatically proof any identity that contains holonomicity preserving manipulations. In the case of sequences, these manipulations are additions, (termwise or Cauchy) multiplications, partial (indefinite) summation, differences, shifts, subsequences (with constant step width) and interlacements. Holonomic power series may be manipulated by additions, (termwise or Cauchy) multiplications, indefinite integrations, differentiations, and compositions with algebraic functions.

These computations can be performed if the power series and sequences are defined over a computable field $\mathbf{K}$, where $\mathbf{K}$ is $\mathbf{Q}$ or a finitely generated extension of $\mathbf{Q}$ (possibly containing complex algebraic numbers) or the field of rational functions in several

[1]The problem of verifying that $f(x) = 0$ for all $x \in \mathbf{R}$ is not in general decidable as shown by Richardson [Ric68].

indeterminates over **Q** or its extension, respectively.[2]

B. Salvy and P. Zimmermann [SZ94] (INRIA Paris, France) implemented these manipulations in Maple . Their package `gfun`[3] also contains procedures that perform guessing.

Based on the ideas and the philosophy of `gfun` the author implemented the Mathematica package `GeneratingFunctions`[3]. The implementation of some easy-to-use and user-friendly interfaces was inspired by I. Nemes' and M. Petkovšek's Mathematica package `RComp` [NP95], which contains procedures for manipulating and computing with *C-finite* or *C-recursive* sequences, i.e, holonomic sequences that satisfy recurrences with constant coefficients. A great part of the procedures require the solution of systems of linear equations where the coefficients are rational functions (in possibly more than one indeterminate). These linear systems are solved by a Nullspace procedure which has been implemented by E. Aichinger.

In this chapter we want to describe the structure and the functionality of the package `GeneratingFunctions`, hence it may be regarded as a manual. Moreover, the examples that are given, shall illustrate how the package may be used as an assistant that does the "routine jobs" in the process of solving problems.

2.2 Installation

The implementation consists of the following three files:

`GeneratingFunctions.m`	The Mathematica source code
`examples.txt`	A Mathematica session containing examples
`readme.txt`	Information about previous and current versions

By typing the Mathematica command `<<GeneratingFunctions.m` the whole package is installed. (Be sure that Mathematica finds the directory where this file is located.) If your operating system does not allow file names with more than eight characters (or if you don't want to type this lengthy name), simply rename the file that contains the source code into, for instance, `GF.m`. Then the package can be loaded by the command `<<GF.m`.

2.3 Classification

To describe the procedures, we split the package `GeneratingFunctions` into four basic parts.

[2] Though Mathematica is able to handle transcendental constants like π or e in most cases, non algebraic extensions of **Q** should be used carefully.

[3] See Appendix A for the availability of this package.

1. *Transformation part:* A holonomic sequence is characterized by a recurrence together with some initial values. A holonomic function (or power series) may be defined by a differential equation and initial values. Some of the procedures in this part transform a recurrence into a differential equation for the generating function of the sequence, and vice versa. We also provide tools to switch between the first terms of a sequence, which are given by a list, and the truncated power series of a certain type of generating function.

   Procedures (Aliases).

`ListToList`	(L2L)
`ListToSeries`	(L2S)
`SeriesToList`	(S2L)
`SeriesToSeries`	(S2S)
`RecurrenceEquationToDifferentialEquation`	(RE2DE)
`DifferentialEquationToRecurrenceEquation`	(DE2RE)
`RecurrenceEquationToList`	(RE2L)

2. *Guessing part:* Given the initial terms of a sequence or power series, these procedures try to *guess* a holonomic sequence or function (sometimes of a special type) that has these initial terms.

   Procedures (Aliases).

`GuessRecurrenceEquation`	(GuessRE)
`GuessDifferentialEquation`	(GuessDE)
`GuessRationalFunction`	(GuessRatF)
`GuessAlgebraicEquation`	(GuessAE)

3. *Closure properties:* This part is the "heart" of the package. It is an implementation of the results that are proved in Section 1.4.

   Procedures (Aliases).

`RecurrenceEquationPlus`	(REPlus)
`RecurrenceEquationHadamard`	(REHadamard)
`RecurrenceEquationCauchy`	(RECauchy)
`DifferentialEquationPlus`	(DEPlus)
`DifferentialEquationHadamard`	(DEHadamard)
`DifferentialEquationCauchy`	(DECauchy)
`AlgebraicEquationToDifferentialEquation`	(AE2DE)
`AlgebraicCompose`	(ACompose)
`RecurrenceEquationSubsequence`	(RESubsequence)
`RecurrenceEquationShadow`	(REShadow)

`RecurrenceEquationInterlace`	`(REInterlace)`
`HomogenousRecurrenceEquation`	`(HomogenousRE)`
`HomogenousDifferentialEquation`	`(HomogenousDE)`

4. *Interface to the system:* In principle, the procedures in the parts 1–3 contain enough tools to work with holonomic sequences and their generating functions (in the scope of the closure properties). However, these computations might demand some typing effort from side of the user. Therefore we provide some user interfaces that make it possible to *define* sequences and functions via holonomic equations. Once these definitions are performed, we can add, subtract, multiply, verify identities, a.s.o., just by typing `+`, `-`, `*`, `==`, etc.

   Procedures (Aliases).

`DefineSequence`	`(DefineS)`
`DefineFunction`	`(DefineF)`
`RecurrenceEquationOut`	`(REOut)`
`DifferentialEquationOut`	`(DEOut)`

   Additional "short form" operations for sequences.

   `+, -, *, ==, PSum, Delta, Shift`

   Additional "short form" operations for functions.

   `+, -, *, ==, Integrate, D, Series`

Remark. Following a Mathematica philosophy, the names given to the procedures in this package are fully spelt out, i.e., the words in the procedure calls are not abbreviated. However, the user has to do a lot of typing, whenever he calls some of these procedures. For this reason, the author supplied aliases for the procedure names that are easier typed and still descriptive enough to understand their actions.

In order to avoid ugly and lengthy Mathematica expressions, we subsequently will also use the shorter aliases in most cases.

2.4 The Input Equations

Most of the procedures in `GeneratingFunctions` take recurrence, differential and/or algebraic equations as input. In this section we discuss the structure of the input equations.

2.4.1 Recurrence Equations

We characterize a recurrence in a`[`n`]`. (a and n are symbols.)

- A *recurrence equation* (RE) may be a single *recurrence relation* (RR) or a list consisting of exactly one RR and an arbitrary number of *initial conditions for an RE* (ICR).
- An RR is a Mathematica expression R of the form

  $$R(n,\mathtt{1},a\mathtt{[}n\mathtt{+}b_0\mathtt{]},a\mathtt{[}n\mathtt{+}b_0+1\mathtt{]},\ldots,\ a\mathtt{[}n\mathtt{+}b_0+d\mathtt{]}),$$

  where b_0 is any integer, d any nonnegative integer, and R is rational in n and linear with respect to the other variables. We will refer to this expression as $R(a,n,b_0,d)$. (Note, that this definition allows inhomogenous recurrences of the form (1.3.2).)

  Alternatively, an RR may be given as an equation of the form

  $$lhs\mathtt{==}rhs,$$

  where this expression can be transformed into

  $$R(a,n,b_0,d)\mathtt{==0}$$

  just by moving rhs to the left and clearing denominators. (Note, that this may cause troubles if the common denominator vanishes for some nonnegative integer. Hence, in the case that this denominator is not free of the variable n, a warning message is given.)

  If $b_0 \geq 0$, the equation $R(a,n,b_0,d) = 0$ is interpreted to be valid for all $n \geq 0$.

  If $b_0 < 0$, then $R(a,n,b_0,d) = 0$ is assumed to hold for all $n \geq -b_0$ or, equivalently, $\overline{R}(a,n,0,d) = 0$ for $n \geq 0$, where $\overline{R}$ is obtained from R by replacing every occurrence of n by $n - b_0$.
- An ICR is an equation (containing possibly more than one "`==`"-sign) that involves at least one of the expressions `a[0]`, `a[1]`,...,`a[`n_0`]`, where n_0 is a nonnegative integer (which does not necessarily depend on b_0 or d). The ICR must not contain the variable n.

Following these rules, the sequence of Fibonacci numbers (see (1.3.4)), which starts with 0,1,1,2,3,5,8,13,etc., might be defined in several different ways:

```
{f[n+2]==f[n]+f[n+1], f[0]==0, f[1]==1}
{f[n+2]-f[n]-f[n+1], f[1]==f[2]==1}
{f[4]==3, f[n]==f[n-2]+f[n-1], f[5]==f[4]+2}
```

Possible recurrences for $n!$, the sequence of factorials, would be:

```
{a[n]==n*a[n-1], a[0]==1}
{a[n+1]/a[n]==n+1, a[0]==1}
{a[n-2]/a[n]==1/(n-1)/n, a[0]==a[1]==1}
```

A recursive definition of the Gegenbauer polynomials $C_n^{\alpha}(x)$ (see Example 1.3.1) might be given by:

```
{c[n+2]==(-(n+2*al)*c[n]+2*(n+1+al)*x*c[n+1])/(n+2),c[0]==1,c[1]==2al*x}
```

(We used the variable `al` for α.)

The recurrences may also be given without initial conditions; for instance, `f[n+2]== f[n]+f[n+1]` is an RE that represents the whole solution space of this recurrence. Other admissible input forms show up in the subsequent sections of this chapter.

Finally, we want to give a few examples of recurrences that are **not accepted**:

`a[2*n]==2*a[n]`	(This recurrence is not holonomic.)
`{a[n+1]==a[n]+a[0],a[0]==1}`	(Initial values must not show up in the RR.)
`{b[n]==b[n-1]+n!,b[0]==1}`	(The inhomogenous part is not rational in `n`).

2.4.2 Differential Equations

We characterize a differential equation in f`[`x`]`. (f and x are symbols.)

- A *differential equation* (DE) may be a single *differential relation* (DR) or a list consisting of exactly one DR and an arbitrary number of *initial conditions for a DE* (ICD).
- A DR is an expression D of the form

  $$D(x,\mathtt{1},f\mathtt{[}x\mathtt{]},f\mathtt{'[}x\mathtt{]},\ldots,\mathtt{Derivative[}d\mathtt{][}f\mathtt{][}x\mathtt{]})\,,$$

  where d is a nonnegative integer, and D is rational in x and linear with respect to the other variables. We will refer to this expression as $D(f,x,d)$. (Note, that this definition allows inhomogenous differential equations of the form (1.2.2).)

  Alternatively, a DR may be given as an equation of the form

  $$lhs\mathtt{==}rhs,$$

  where this expression can be transformed into

  $$D(f,x,d)\mathtt{==0}$$

  ust by moving rhs to the left and clearing denominators. (Note, that this may cause troubles if the common denominator vanishes at the origin. Hence, in the case that this denominator is not free of the variable x, a warning message is given.)

The equation $D(f, x, d) = 0$ is assumed to be valid for all x in the ground domain.

- An ICD is an equation (containing possibly more than one "=="-sign) that involves at least one of the variables `f[0]`,`f'[0]`,..., `Derivative[`n_0`][f][0]`, where n_0 is a nonnegative integer (which does not necessarily depend on d). The ICD must not contain the variable x.

Following these rules, the generating function of the sequence of Fibonacci numbers might be defined in several different ways:

```
f[x]==x/(1-x-x^2)
{(1+x^2)*f[x]+(-x+x^2+x^3)*f'[x],f[0]==0,f'[0]==1}
```

For example, we can define the sine function $\sin(x)$ via:

```
{s''[x]==-s[x],s[0]==Sin[0],s'[0]==Cos[0]}
```

The generating function $f(y) = \sum_{n\geq 0} C_n^\alpha(x)y^n$ of the Gegenbauer polynomials $C_n^\alpha(x)$ (see Example 1.2.1) might be given by:

```
{2*al(-x+y)*f[y]+(1-2*x*y+y^2)*f'[y],f[0]==1}
```

(We used the variable `al` for α.)

The differential equations may also be given without initial conditions. Other admissable input forms show up in the subsequent sections of this chapter.

The following examples are equations that are **not accepted**:

`f[x^2]==f'[x]+1`	(This equation is not holonomic.)
`{f'[x]==f[x],f[1]==1}`	(Initial values must be given at 0.)
`{f''[x]-f[x]+Exp[x],f[0]==1}`	(The inhomogenous part is not rational in `x`).

2.4.3 Algebraic Equations

We characterize an algebraic equation in `f[x]`. (f and x are symbols.)

- An *algebraic equation* (AE) may be a single *algebraic relation* (AR) or a list consisting of exactly one AR and at most one *initial condition for an AE* (ICA).

- An AR is an expression $A(f, x)$, that is rational in x and polynomial in `f[x]`. (For `f[x]` the abbreviation f is allowed.)

Alternatively a AR may be given as an equation of the form

$$lhs\texttt{==}rhs,$$

where this expression can be transformed into

$$A(f,x)\texttt{==0}$$

just by moving rhs to the left and clearing denominators. (Note, that this may cause troubles if the common denominator vanishes at the origin. Hence, in the case that this denominator is not free of one of the variables f`[`x`]` or x, a warning message is given.)

The equation $A(f,x) = 0$ is assumed to be valid for all x in the ground domain.

- An ICA is an equation that involves the variable f`[0]`. The ICA must not contain the variable x.

Following these rules, the generating function of the sequence of Catalan numbers (see page 15) might be defined in one of the following ways:

```
{(c[x]-1)/x==c[x]^2,c[0]==1}
{x*c^2-c+1,c[0]==1}
```

The algebraic equations may also be given without initial conditions. Other admissable input forms show up in the subsequent sections of this chapter.

The following examples are algebraic equations that are **not accepted**:

`f[x^2]==f[x]+1` (This equation is not algebraic.)

`{f[x]==Sqrt[1-x],f[0]==1}` (This equation does not define the algebraic function via an algebraic equation.)

2.5 The Transformation Part

Holonomic sequences have the nice property that the elements can be quickly computed (affording little computer memory). A procedure that is described below does this job, i.e., it takes a holonomic recurrence as input and returns some elements of the sequence.

Sometimes we try to solve a problem with holonomic sequences via their generating functions or vice versa.

In other cases we are given the first terms of a sequence and we want to get a truncated power series of the exponential generating function of this sequence.

The procedures that are described in this section perform these (and other) transformations.

We introduce some different types of generating functions: Let $(a_n)_{n\geq 0}$ be an arbitrary sequence. We define

(a) $f(x) = \sum_{n\geq 0} a_n x^n$, *(ordinary generating function)*

(b) $\widehat{f}(x) = \sum_{n\geq 0} a_n x^n/n!$, *(exponential generating function)*

(c) $f^*(x)$, where $f^*(f(x)) = x$ and *(reverse ordinary generating function)*

(d) $\widehat{f}^{\,*}(x)$, where $\widehat{f}^{\,*}(\widehat{f}(x)) = x$. *(reverse exponential generating function)*

We note that the reverse ordinary and reverse exponential generating functions are formal power series if and only if $a_0 = 0$ and $a_1 \neq 0$.

2.5.1 `ListToList (L2L)`

The procedure `L2L[`*list*`,`*transf*`]` computes the initial terms of a sequence that is obtained after performing the transformation *transf* to a sequence (or more precisely: to its generating function), which is given by its first terms *list*=`{`a_0`,`a_1`,`$\ldots$`,`a_k`}`.

The built-in transformations are `"ogf"`, (ordinary generating function), `"egf"` (exponential generating function) and `"revogf"` (reverse ordinary generating function) `"revegf"` (reverse exponential generating function), which means that the elements in the input list are regarded as power series coefficients of the ordinary generating function, and the elements in the output list are the first coefficients of the type of generating function that is specified by *transf*.

Examples.

```
In[1]:= ListToList[{0,1,2,3,4,5,6},"egf"]

                  1  1  1    1
Out[1]= {0, 1, 1, -, -, --, ---}
                  2  6  24  120

In[2]:= L2L[{0,1,a,a^2,a^3,a^4},"revogf"]

                   2    3   4
Out[2]= {0, 1, -a, a , -a , a }
```

We mention that the transformation `"ogf"` is the identity, and hence input and output list are equal. The same "result" is obtained, if no transformation is given.

2.5.2 ListToSeries (L2S)

The procedure `L2S[`*list*`,`*x*`,`*transf*`]` performs a transformation, as described above, to the input list *list*. The output is given as a truncated power series expansion in x. If the third parameter is omitted, the procedure takes the default value *transf*=`"ogf"`.

Examples.

```
In[3]:= ListToSeries[{1,2,3,4,5,6},x]

                    2       3       4       5        6
Out[3]= 1 + 2 x + 3 x  + 4 x  + 5 x  + 6 x  + O[x]

In[4]:= L2S[{0,1,a,a^2,a^3,a^4},x,"revogf"]

              2     2  3     3  4     4  5       6
Out[4]= x - a x  + a  x  - a  x  + a  x  + O[x]
```

2.5.3 SeriesToList (S2L)

Given a truncated power series, the procedure `S2L[`*series*`,`*x*`,`*transf*`]` outputs a list of the first coefficients of the power series, which is obtained after performing the transformation *transf* to the function. In case that the third parameter is omitted, no transformation is done. The input variable *series* may be given as a Taylor series (around the point 0) or as a polynomial in x.

Examples.

```
In[5]:= SeriesToList[Series[ArcSin[x],{x,0,8}],x]

                    1      3        5
Out[5]= {0, 1, 0, -, 0, --, 0, ---, 0}
                    6      40       112

In[6]:=S2L[x+x^2+2*x^3+3*x^4+5*x^5+8*x^6+13*x^7,x,"revegf"]

                  1   1   1      5     11     209
Out[6]= {0, 1, -(-), -, --, -(--), --, -(----)}
                  2   6   12     24    60     5040
```

2.5.4 SeriesToSeries (S2S)

The command `S2S[`*series*`,`*x*`,`*transf*`]` gives the same output as performing `S2L`. The output given here is a series instead of a list.

Example.

```
In[7]:= SeriesToSeries[Series[Sin[x],{x,0,8}],x,"revogf"]

              3       5       7
             x     3 x     5 x          8
Out[7]= x + -- + ---- + ---- + O[x]
             6      40     112
```

2.5.5 Adding New Transformations

The command `ListOfTransformations[]` gives a list of the currently implemented transformations that may be chosen in the procedure calls for `L2L`, `L2S`, `S2L`, `S2S`:

```
In[8]:= ListOfTransformations[]

Out[8]= {ogf, egf, revogf, revegf}
```

(Note that this output is a list of strings, i.e., to refer to a transformation we have to put it between quotes; for example `"revogf"`.)

In Section 2.6 we describe how all the supplied transformations may be used in the process of *guessing*. In some circumstances it might be desireable to have more transformations available. We explain in an example how the user can expand the system: Suppose we'd like to have a transformation, which, given the initial terms of a function (or series) f, outputs the initial terms of the reciprocal $1/f$. Let's call this transformation `"rcpogf"` (for *reciprocal ordinary generating function*).

We proceed in two steps: First, we define a procedure `Li2Li[list_,"rcpogf"]`, which does this transformation for the input `list`. Here, we can use the fact that Mathematica automatically computes the reciprocal of a series, if we type `1/SeriesData[_]`. However, we must not forget to check wheter the input list (series) in fact has a reciprocal:

```
(* Case 1: Series has reciprocal power series (coeff of x^0=!=0) *)
Li2Li[list_,"rcpogf"]:=Module[{x},
   CoefficientList[1/SeriesData[x,0,list,0,Length[list],1],x]
 ]/;list[[1]]=!=0;

(* Case 2: Series has no reciprocal power series (coeff of x^0===0) *)
Li2Li[list_,"rcpogf"]:=
   Print["Reciprocal is no power series."]/;list[[1]]===0;
```

(We note that due to the internal structure of the package the procedure name `Li2Li` must not be replaced by a different one.) The second step is to redefine the command `ListOfTransformations[]`. (This step may be skipped, if we do not want to use this command.)

```
ListOfTransformations[]:={"ogf","egf","revogf","revegf","rcpogf"};
```

Now we can use the transformation `"rcpogf"` just like the built in ones:

```
In[9]:= L2S[Table[ChebyshevT[n,x],{n,0,4}],y,"rcpogf"] //Simplify

                          2   2          3   3      2    4   4       5
Out[9]= 1 - x y + (1 - x ) y  + (x - x ) y  + (x  - x ) y  + O[y]
```

2.5.6 RecurrenceEquationToDifferentialEquation (RE2DE)

The procedure `RE2DE[`re`,`a`[`n`],`f`[`x`]]` gives a holonomic differential equation in f`[`x`]`, which is satisfied by $f(x)$, the ordinary generating function of a sequence $(a_n)_{n\geq 0}$, which satisfies the recurrence re in a`[`n`]`.

Example. Let D_n be the number of derangements of n letters. (A derangement is a permutation that has no fixed points.) If $a_n = D_n/n!$, then the sequence $(a_n)_{n\geq 0}$ is recursively defined by

$$a_n + (n+1)a_{n+1} - (n+2)a_{n+2} = 0, \qquad \text{and} \qquad a_0 = 1, \quad a_1 = 0.$$

We compute a differential equation that is satisfied by the exponential generating function $f(x) = \sum_{n\geq 0} a_n x^n$ of the sequence $(D_n)_{n\geq 0}$:

```
In[10]:= RE2DE[{a[n]+(n+1)*a[n+1]-(n+2)*a[n+2]==0,a[0]==1,a[1]==0},f[x]]

Out[10]= {x f[x] + (-1 + x) f'[x] == 0, f[0] == 1}
```

Inhomogenous recurrences are also handled:

Example.

```
In[11]:= RE2DE[a[n]==1/(n+1),a[n],f[x]]

CanRE::denom:
   Warning. The input equation will be multiplied by its
     denominator.

                                             2
Out[11]= {1 + (-1 + x) f[x] + (-x + x ) f'[x] == 0, f[0] == 1}
```

2.5.7 DifferentialEquationToRecurrenceEquation (DE2RE)

The procedure `DE2RE[`de`,`f`[`x`],`a`[`n`]]` gives a holonomic recurrence equation in a`[`n`]`, satisfied by $(a_n)_{n\geq 0}$ the sequence of power series coefficients of $f(x)$, which is given by the differential equation de in f`[`x`]`.

Example.

```
In[12]:= DE2RE[{y[x]+3*x*y'[x]+y''[x]==0,y[0]==0,y'[0]==1},y[x],c[n]]

Out[12]= {(1 + 3 n) c[n] + (1 + n) (2 + n) c[2 + n] == 0, c[0] == 0,

 c[1] == 1}
```

Rational functions yield recurrences with constant coefficients. A particular example is the bivariate generating functions of the binomials:

Example.

```
In[13]:= DE2RE[f[y]==1/(1-y-x*y),f[y],a[n]]

CanDE::denom:
   Warning. The input equation will be multiplied by its
     denominator.

Out[13]= {(1 + x) a[n] - a[1 + n] == 0, a[0] == 1}
```

2.5.8 RecurrenceEquationToList (RE2L)

The procedure `RE2L[`re`,`a`[`n`],`n_0`]` computes the elements of a sequence $(a_n)_{n\geq 0}$, which satisfies the recurrence re in a`[`n`]`, from a_0 to a_{n_0}, where $n_0 \in \mathbf{N}$.

Example. We compute an initial segment of the sequence of the Fibonacci numbers: (Note that the initial values do not contain `f[0]`.)

```
In[14]:= RecurrenceEquationToList[{f[n]==f[n-1]+f[n-2],f[1]==f[2]==1},
         f[n],10]

Solve::svars: Warning: Equations may not give solutions for all "solve"
     variables.

Out[14]= {0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55}
```

(Mathematica gives a warning in this case, since the system does not know in advance that the given initial values `f[1]` and `f[2]` uniquely determine the sequence.)

Now we extend the recurrence to be valid for *all* integers, which makes it possible to use negative indices, too: `RE2L[`re`,`a`[`n`],{`n_0`}]` gives just one element a_{n_0} of the sequence (as a list), where $n_0 \in \mathbf{Z}$:

Example.

```
In[15]:= RE2L[{a[n]==n*a[n-1],a[0]==1},a[n],{20}]

Out[15]= {2432902008176640000}
```

`RE2L[`re`,`a`[`n`],{`n_0`,`n_1`}]` computes the sequence from a_{n_0} up to a_{n_1}, if $n_0 \leq n_1$ and $n_0, n_1 \in \mathbf{Z}$.

Example.

```
In[16]:= RE2L[a[n]==n^2,a[n],{5,10}]

Out[16]= {25, 36, 49, 64, 81, 100}
```

If no singularties arise, we can compute a segment of the sequence containing negative indices.

Example.

```
In[17]:= RE2L[{f[n]+f[1+n]-f[n+2],f[0]==0,f[1]==1},f[n],{-5,5}]

RE2L::negative:
   Warning. The recurrence is extended to (some) negative
     integers.

Out[17]= {5, -3, 2, -1, 1, 0, 1, 1, 2, 3, 5}
```

2.6 The Guessing Part

Sometimes it happens that we are given some elements of a sequence and we do not know whether this sequence satisfies a holonomic recurrence.

In other cases we are faced with the problem that a holonomic sequence is given by a recurrence equation that can not be solved in explicit terms and we we would like to know whether the given sequence also satisfies a recurrence of a special type, which allows us to get an explicit expression for the nth element.

Perhaps, we are given a truncated power series, and we would like to know whether it might come from an algebraic function.

In these cases it may help, if we try to "guess" the desired equations. Following an idea by Bergeron and Plouffe [BP92], my package contains procedures that perform guessing.

Remark. The procedures described below, take lists as input. In order to get results, the elements in the list must be constants or rational functions (in several variables). For example, if the input list contains the *Chebyshev polynomials of the first kind* $T_n(x) = \cos(n \arccos x)$, it is necessary to set up the input list by

```
list={1,x,2*x^2-1,4*x^3-3*x,...}
```

rather than by:

```
list={Cos[0],Cos[ArcCos[x]],Cos[2*ArcCos[x]],Cos[3*ArcCos[x]],...}
```

2.6.1 GuessRecurrenceEquation (GuessRE)

`GuessRE[`*list*`,`*a*`[`*n*`],{`*minorder*`,`*maxorder*`},{`*mindeg*`,`*maxdeg*`},`*opts*`]` tries to find a holonomic recurrence equation that is satisfied by the elements from the list *list*. The last parameter *opts* is a (possibly empty) sequence of options. The procedure tries orders from *minorder* to *maxorder* and degrees from *mindeg* to *maxdeg*. In case that a recurrence is found, the output contains the recurrence in *a*`[`*n*`]` together with some initial conditions and a transformation that had to be performed on the sequence in order to obtain the recurrence. (See Section 2.5.) The output is `"FAIL"`, if no recurrence was found.

The default values *minorder* = 1, *maxorder* = 2, *mindeg* = 0, *maxdeg* = 3, are applied, if one of the short forms `GuessRE[`*list*`,`*a*`[`*n*`]]` or `GuessRE[`*list*`,`*a*`[`*n*`],`*maxorder*`,` *maxdeg*`]` is used.

`GuessRE` allows the following options (with the given default values):

- `AdditionalEquations->"All"` In order to avoid accidental results, `"All"` elements in the input list are used to build the equations for the coefficients of the recurrence. Setting this parameter to a positive integer, say k, causes the procedure to build $d + k$ equations, where d is the number of indeterminants. This option can be used in order to achieve a speed-up.

- `Hypergeom->False` If this parameter is set to be `True`, only m-hypergeometric recurrences are searched for. An m-hypergeometric recurrence has the form $p_0(n)a_n + p_1(n)a_{n+m} = 0$, from which it easy to extract a closed expression for a_n.

- `Transform->{"ogf","egf"}` This parameter gives a list of the transformations that are tried automatically. The option `Transform->ListOfTransformations[]` tells the procedure to try all implemented transformations.

Examples.

```
In[18]:= GuessRE[Table[ChebyshevT[n,x],{n,0,10}],T[n]]

Out[18]= {{T[n] - 2 x T[1 + n] + T[2 + n] == 0, T[0] == 1,

  T[1] == x}, ogf}
```

An example shows the usage of the option `Hypergeom->True`: Suppose, we want to derive a closed expression for the power series coefficients of $\sin(x)\exp(x)$:

```
In[19]:= list=S2L[Series[Sin[x]*Exp[x],{x,0,40}],x];

In[20]:= GuessRE[list,a[n]]

Out[20]= {{2 a[n] + (-2 - 2 n) a[1 + n] +

                       2
      (2 + 3 n + n ) a[2 + n] == 0, a[0] == 0, a[1] == 1}, ogf}
```

Since this recurrence has no special form that allows us to extract a closed expression for the coefficient `a[n]`, we try to guess a recurrence equation of this "special form", i.e, an m-hypergeometric recurrence:

```
In[21]:= GuessRE[list,a[n],Hypergeom->True]

Out[21]= FAIL
```

Perhaps a search for a recurrence of higher degree and order succeeds:

```
In[22]:= GuessRE[list,a[n],5,5,Hypergeom->True]

                                        2       3    4
Out[22]= {{4 a[n] + (24 + 50 n + 35 n  + 10 n  + n ) a[4 + n] ==

                                                      1
     0, a[0] == 0, a[1] == 1, a[2] == 1, a[3] == -}, ogf}
                                                      3
```

Although we can solve this recurrence for `a[n]` (see, e.g. [Koe92]), yet we don't know whether the power series coefficients of $\sin(x)\exp(x)$ are really determined by the guessed recurrence. In Chapter 3 we will see how we can check the validity of a guessed holonomic recurrence (or differential) equation.

It is also possible to use indeterminates:

```
In[23]:= GuessRE[{0,a[1],2,3,4,5,6,7,8,9,10},a[n]]

Out[23]= {{(1 - n) a[n] + 3 n a[1 + n] - 2 n a[2 + n] == 0,

  a[0] == 0, a[2] == 2}, ogf}
```

We have seen that the tangent function is not holonomic. `GuessRE` also can't find a holonomic recurrence:

```
In[24]:= list=S2L[Series[Tan[x],{x,0,20}],x]

                   1     2      17       62       1382
Out[24]= {0, 1, 0, -, 0, --, 0, ---, 0, ----, 0, ------, 0,
                   3     15     315      2835     155925

  21844        929569         6404582          443861162
 -------, 0, ---------, 0, -----------, 0, -------------, 0}
 6081075     638512875     10854718875     1856156927625

In[25]:= GuessRE[list,a[n]]

Out[25]= FAIL
```

However, if we try all available transformations, a recurrence that is satisfied by the power series coefficients of the compositional inverse of $\tan(x)$ is found:

```
In[26]:= GuessRE[list,a[n],Transform->ListOfTransformations[]]

Out[26]= {{n a[n] + (2 + n) a[2 + n] == 0, a[0] == 0,

  a[1] == 1}, revogf}
```

2.6.2 GuessDifferentialEquation (GuessDE)

`GuessDE[`*list*`,`*f*`[`*x*`],{`*minorder*`,`*maxorder*`},{`*mindeg*`,`*maxdeg*`},`*opts*`]` tries to find a holonomic differential equation in *f*`[`*x*`]` that is satisfied by the generating function of a sequence that has the initial elements *list*. The last parameter *opts* is a (possibly empty) sequence of options. The procedure tries orders from *minorder* to *maxorder* and degrees from *mindeg* to *maxdeg*. If a differential equation is found, the output contains this equation together with some initial conditions and a transformation that had to be performed on the sequence in order to get the recurrence. (See Section 2.5.) The output is `"FAIL"`, if no differential equation was found.

The default values $minorder = 1$, $maxorder = 2$, $mindeg = 0$, $maxdeg = 3$, are applied, if one of the short forms `GuessDE[`*list*`,`*f*`[`*x*`]]` or `GuessDE[`*list*`,`*f*`[`*x*`],`*maxorder*`,` *maxdeg*`]` is used.

`GuessDE` admits the following options (with the given default values):

- `AdditionalEquations->"All"` In order to avoid accidental results, `"All"` elements in the input list are used to build the equations for the coefficients of the differential equation. Setting this parameter to a positive integer, say k, causes the procedure to build $d+k$ equations, where d is the number of indeterminants. This option can be used to achieve a speed-up.
- `Inhomog->False` By default, the procedure searches for homogenous differential equations only. If this parameter is set to be `True`, the search also includes inhomogenous differential equations of the form (1.2.2). A search with this option may find a differential equation of lower order than the one delivered by a search with the default option.
- `Transform->{"ogf","egf"}` This parameter gives a list of the transformations that are tried. The option `Transform->ListOfTransformations[]` tells the procedure to try all implemented transformations.

Examples. We try to find a differential equation that is satisfied by `f[y]`, the generating function of the Chebyshev polynomials of the first kind:

```
In[27]:= list=Table[ChebyshevT[n,x],{n,0,15}];

In[28]:= GuessDE[list,f[y]]

Out[28]= {{2 f[y] + 4 (-x + y) f'[y] +

                    2
     (1 - 2 x y + y ) f''[y] == 0, f[0] == 1, f'[0] == x}, ogf}
```

If we set the option `Inhomog->True`, we get a much simpler differential equation of order 0 (instead of order 2).

```
In[29]:= GuessDE[list,f[y],Inhomog->True]
```

```
                                    2
Out[29]= {1 - x y + (-1 + 2 x y - y ) f[y] == 0, ogf}
```

Now we conjecture that `f[y]` is a rational function over the field $\mathbf{Q}(x)$.

If we suspect a sequence to have a rational generating function, we can also use a different (and faster) procedure to do the guessing:

2.6.3 `GuessRationalFunction (GuessRatF)`

`GuessRatF[`*list*`,`*x*`,`*maxdeg*`]` tries to guess a rational generating function in x for the sequence, whose initial terms are given in *list*. The maximum degrees of the numerator and denominator are given by *maxdeg*. Short form for the function call is `GuessRatF[`*list*`,`*x*`]`, where the default value $maxdeg = 5$ is used. `GuessRatF` has the same options and default values as `GuessRE`.

Example.

```
In[30]:= GuessRatF[{0,1,1,2,3,5,8,13,21},x]

                x
Out[30]= {----------, ogf}
                   2
          1 - x - x
```

2.6.4 `GuessAlgebraicEquation (GuessAE)`

`GuessAE[`*list*`,`*x*`,{`*minorder*`,`*maxorder*`},`*maxdeg*`,`*opts*`]` tries to guess an algebraic equation in f`[`x`]` for the generating function of the sequence, whose initial terms are given in *list*. The parameter *opts* is a (possibly empty) sequence of options. The maximum degrees of the polynomials in the algebraic equation are given by *maxdeg*. The orders (degree in f`[`x`]`) that are tried range from *minorder* to *maxorder*. Short forms for the function call are `GuessAE[`*list*`,`f`[`x`]]`, `GuessAE[`*list*`,`f`[`x`],`*maxorder*`,`*maxdeg*`]`, where the default values $minorder = 1$, $maxorder = 3$, $maxdeg = 3$ are used.

`GuessAE` has the options `AdditionalEquations` and `Transform`, where the usage and the default values are the same as in `GuessRE`.

Example. (Special instance of an m-Raney sequence with $m = 3$. See [GKP94, p. 360]) Let R_n be the number of sequences $(a_0, a_1, \ldots, a_{3n})$, where $a_k \in \{+1, -2\}$ for $(0 \le k \le 3n)$, whose partial sums are all positive and $\sum_{k=0}^{3n} a_k = 1$. We (somehow) compute a list that contains the elements R_n and try to guess an algebraic equation that is satisfied by the generating function $f(x) = \sum_{n \ge 0} R_n x^n$:

```
In[31]:= GuessAE[{1,1,3,12,55,273,1428,7752,43263,246675,1430715,
         8414640,50067108,300830572,1822766520,11124755664,
         68328754959,422030545335,2619631042665,16332922290300},
         f[x]]

                                    3
Out[31]= {{1 - f[x] + x f[x]  == 0, f[0] == 1}, ogf}
```

2.7 The Closure Properties

The most important reason for working with holonomic functions and sequences are the closure properties, discussed in Section 1.4. The results we obtained, have the consequence that all manipulations of holonomic sequences containing additions, (termwise or Cauchy) multiplications, partial (indefinite) summation, differences, shifts, subsequences (with constant step width) and interlacements can be done completely "automatically".

Holonomic functions may be manipulated by additions, (termwise or Cauchy) multiplications, indefinite integrations, differentiations, and compositions with algebraic functions.

Moreover, the computer can "prove" any identity that is built from these manipulations of holonomic functions and sequences.

2.7.1 `RecurrenceEquationPlus (REPlus)` `RecurrenceEquationHadamard (REHadamard)` `RecurrenceEquationCauchy (RECauchy)`

`REPlus[`re_1`,`re_2`,`a`[`n`]]` gives a recurrence equation that is satisfied by the sum of solutions of the recurrences re_1 and re_2 in a`[`n`]`.

Example. Suppose two sequences $(a_n)_{n\geq 0}$ and $(b_n)_{n\geq 0}$ are defined via the recurrences

$$(n+1)a_n + (n-1)a_{n+1} - 4a_{n+2} - a_{n+3} = 0 \quad \text{and}$$

$$(n+1)(n+2)b_n + (n+2)(n-2)b_{n+1} - (4n+5)b_{n+2} - (n+1)b_{n+3} = 0,$$

respectively. (The recurrences hold for all $n \in \mathbf{N}$.) Suppose that both sequences have the same initial values $a_0 = b_0 = 1$, $a_1 = b_1 = 1$ and $a_2 = b_2 = -2$. We inspect the first, say, ten terms of both sequences and, since all a_n's and b_n's are identical for these small values of n, we might conjecture that $a_n = b_n$ for all $n \in \mathbf{N}$. The procedure `REPlus` checks this identity:

`re1` is the recurrence satisfied by $(a_n)_{n\geq 0}$:

```
In[32]:= re1={(n+1)*a[n]+(n-1)*a[n+1]-4*a[n+2]-a[n+3]==0,a[0]==a[1]==1,
              a[2]==-2}

Out[32]= {(1 + n) a[n] + (-1 + n) a[1 + n] - 4 a[2 + n] -

     a[3 + n] == 0, a[0] == a[1] == 1, a[2] == -2}
```

`re2` is a recursive definition of the sequence $(-b_n)_{n\geq 0}$. (Since `REPlus` takes as input two recurrences in the same variable, we have to give `re2` in `a[n]`.)

```
In[33]:= re2={(n+1)*(n+2)*a[n]+(n+2)*(n-2)*a[n+1]-(4n+5)*a[n+2]
             -(n+1)*a[n+3]==0,a[0]==a[1]==-1,a[2]==2}

Out[33]= {(1 + n) (2 + n) a[n] + (-2 + n) (2 + n) a[1 + n] -

     (5 + 4 n) a[2 + n] - (1 + n) a[3 + n] == 0, a[0] == a[1] == -1,

   a[2] == 2}
```

Finally `REPlus` computes a recurrence that is satisfied by the sequence $(c_n)_{n\geq 0}$, where $c_n = a_n - b_n$:

```
In[34]:= REPlus[re1,re2,a[n]]

Out[34]= {(-1 - n) a[n] + (-1 - 2 n) a[1 + n] + (4 - n) a[2 + n] +

     5 a[3 + n] + a[4 + n] == 0, a[0] == 0, a[1] == 0, a[2] == 0,

   a[3] == 0}
```

The initial values $c_0 = c_1 = c_2 = c_3 = 0$ of this fourth order recurrence tell us that $c_n = 0$ for $n \in \mathbf{N}$.

`REHadamard[`re_1`,`re_2`,`a`[`n`]]` gives a recurrence equation that is satisfied by the Hadamard (or termwise) product of solutions of the recurrences re_1 and re_2.

`RECauchy[`re_1`,`re_2`,`a`[`n`]]` gives a recurrence equation that is satisfied by the Cauchy product (or convolution) of solutions of the recurrences re_1 and re_2. All recurrences are given in a`[`n`]`.

Example. Evaluate the definite sum

$$f_n := \sum_{k=0}^{n} \binom{n}{k}^2 .$$

Since

$$\binom{n}{k}^2 = \frac{(n!)^2}{(k!)^2((n-k)!)^2},$$

we can express the sum as $f_n = a_n \sum_{k=0}^{n} b_k b_{n-k}$, where $a_n = (n!)^2$ and $b_n = 1/(n!)^2$.

First, we recursively define the factorials and their reciprocal via the recurrences `factre` and `factrcpre`:

```
In[35]:= factre={a[n+1]==(n+1)*a[n],a[0]==1}

Out[35]= {a[1 + n] == (1 + n) a[n], a[0] == 1}

In[36]:= factrcpre={a[n+1]*(n+1)-a[n],a[0]==1}

Out[36]= {-a[n] + (1 + n) a[1 + n], a[0] == 1}
```

Next, we obtain recurrences `re1` and `re2`, which are satisfied by the sequences $(a_n)_{n\geq 0}$ and $(b_n)_{n\geq 0}$, respectively:

```
In[37]:= re1=REHadamard[factre,factre,a[n]]

                        2
Out[37]= {-((1 + n)  a[n]) + a[1 + n] == 0, a[0] == 1}

In[38]:= re2=REHadamard[factrcpre,factrcpre,a[n]]

                             2
Out[38]= {a[n] - (1 + n)  a[1 + n] == 0, a[0] == 1}
```

Finally, we get the recurrence `sumre`, which is satisfied by f_n:

```
In[39]:= sumre=REHadamard[re1,RECauchy[re2,re2,a[n]],a[n]];

Out[39]= {-2 (1 + 2 n) a[n] + (1 + n) a[1 + n] == 0, a[0] == 1}
```

Since `sumre` is a hypergeometric recurrence, it is possible to obtain a closed expression for f_n and we find (see Example 1.4.1.3):

$$f_n = \frac{4^n (\frac{1}{2})^{\overline{n}}}{n!} = \binom{2n}{n}$$

We want to mention here that, in the general situation, definite sums of the form $f_n = \sum_k g(n,k)$, where the summand $g(n,k)$ is hypergeometric in both variables, can be systematically treated by Zeilberger's algorithm (*method of creative telescoping*, see for instance [Zei90] or [PWZ96, Chapter 6]). Zeilberger's algorithm has been implemented in Mathematica by P. Paule and M. Schorn [PS95].

2.7.2 DifferentialEquationPlus (DEPlus) DifferentialEquationHadamard (DEHadamard) DifferentialEquationCauchy (DECauchy)

`DEPlus[`de_1`,`de_2`,`f`[`x`]]` gives a differential equation that is satisfied by the sum of solutions of the differential equations de_1 and de_2.

`DEHadamard[`de_1`,`de_2`,`f`[`x`]]` gives a differential equation that is satisfied by the Hadamard product of solutions of the differential equations de_1 and de_2.

`DECauchy[`de_1`,`de_2`,`f`[`x`]]` gives a differential equation that is satisfied by the product of solutions of the differential equations de_1 and de_2. All differential equations are given in f`[`x`]`

Examples. We use these procedures to prove the identity $\sin(2x) = 2\sin(x)\cos(x)$.

First we define the sine and the cosine functions via holonomic differential equations:

```
In[43]:= sinde={f[x]+f''[x]==0,f[0]==0,f'[0]==1}

Out[43]= {f[x] + f''[x] == 0, f[0] == 0, f'[0] == 1}

In[44]:= cosde={f[x]+f''[x]==0,f[0]==1,f'[0]==0}

Out[44]= {f[x] + f''[x] == 0, f[0] == 1, f'[0] == 0}
```

Next we are going to prove

$$\underbrace{\sin(2x)}_{f_1(x)} + \underbrace{(-2)\sin(x)\cos(x)}_{f_2(x)} = 0. \tag{2.7.1}$$

Since $\sin(2x) = \sum_{n\geq 0} a_n 2^n x^n$, where $\sin(x) = \sum_{n\geq 0} a_n x^n$, we get a differential equation for $f_1(x)$ via the Hadamard product of the functions $\sin(x)$ and $1/(1-2x) = \sum_{n\geq 0} 2^n x^n$:

```
In[45]:= f1de=DEHadamard[sinde,f[x]==1/(1-2*x),f[x]]

CanDE::denom:
   Warning. The input equation will be multiplied by its
     denominator.

Out[45]= {4 f[x] + f''[x] == 0, f[0] == 0, f'[0] == 2}
```

A holonomic differential equation for $f_2(x)$ can be computed via the ordinary (Cauchy) product of three functions:

```
In[46]:= f2de=DECauchy[DECauchy[sinde,cosde,f[x]],f[x]==-2,f[x]]

                             (3)
Out[46]= {4 f'[x] + f    [x] == 0, f[0] == 0, f'[0] == -2,

  f''[0] == 0}
```

We add the two functions `f1` and `f2` and get a third order differential equation:

```
In[47]:= DEPlus[f1de,f2de,f[x]]

                             (3)
Out[47]= {4 f'[x] + f    [x] == 0, f[0] == 0, f'[0] == 0,

  f''[0] == 0}
```

The initial values $f(0) = f'(0) = f''(0) = 0$ tell us that this differential equation has the unique solution $f(x) = 0$. Thus (2.7.1) is proved.

In Chapter 3 we will have a closer look at the process of proving holonomic identities.

2.7.3 AlgebraicEquationToDifferentialEquation (AE2DE)

`AE2DE[`ae`,`f`[`x`]]` computes a holonomic differential equation that is satisfied by the algebraic function $f(x)$, which is given by the algebraic equation ae in f`[`x`]`.

Example. Let $G_{n,k}$ be the number of plane unlabeled trees with n nodes and k leaves. (A leaf of a tree is a node with degree one.) We try to evaluate m_n, the mean number of leaves of the trees with n nodes. If $s_n = \sum_{k=0}^{n} G_{n,k}$, then the generating function $f(z) = \sum_{n\geq 0} s_n z^n$ is well-known ([FS93, p. 99]) to be

$$f(z) = \frac{1}{2}z + \frac{1}{2}\frac{z}{\sqrt{1-4z}}.$$

We call the procedures `AE2DE` and `DE2RE` to get a recursive definition of the sequence $(s_n)_{n\geq 0}$:

```
In[50]:= de=AE2DE[{(f-z/2)^2==z^2/4/(1-4z),f[0]==0},f[z]]

CanAE::denom:
   Warning. The input equation will be multiplied by its
    denominator.
```

```
           2                             2
Out[50]= {z  + (-1 + 2 z) f[z] + (z - 4 z ) f'[z] == 0, f[0] == 0}

In[51]:= re=DE2RE[de,f[z],s[n]]

                         2               2
Out[51]= {(2 - 6 n + 4 n ) s[n] + (n - n ) s[1 + n] == 0, s[0] == 0}
```

The recurrence `re` is hypergeometric and (although the initial conditions $s_1 = 1$ and $s_2 = 1$, which are required for a unique reperesentation of the sequence, are missing,) we can compute s_n in closed form:

$$s_n = \begin{cases} n & \text{if} \quad n \leq 1 \\ 2\,4^{n-2}\binom{n-3/2}{n-1} & \text{if} \quad n \geq 2 \end{cases}$$

The total number of unlabeled plane trees with n nodes is the nth Catalan number $C_n = \binom{2n}{n}/(n-1)$ (, see page 15 and [FS93, page 23]). Now the mean number of leaves $m_n = s_n/C_n$ is found to be

$$m_n = \begin{cases} 1 & \text{if} \quad n = 1 \\ n/2 & \text{if} \quad n \geq 2 \end{cases}.$$

(Since there is no tree with 0 nodes, m_0 is not defined.)

2.7.4 `AlgebraicCompose (ACompose)`

`ACompose[`de`,`ae`,`f`[`x`]]` computes a differential equation that is satisfied by a function $f(g(x))$, where $f(x)$ is a solution of the differential equation de in f`[`x`]`, $g(x)$ is a solution of the algebraic equation ae. (Note that both input equations as well as the output equation are given in f`[`x`]`.)

Example. We prove *Gauss's identity* [GKP94, p. 222] for hypergeometric functions. Using the notation introduced in Example 1.4.1.3, this identity reads as follows:

$$ {}_2F_1\left(\begin{array}{c} a,b \\ a+b+\frac{1}{2} \end{array}\middle| 4x(1-x)\right) = {}_2F_1\left(\begin{array}{c} 2a,2b \\ a+b+\frac{1}{2} \end{array}\middle| x\right) \tag{2.7.2}$$

Let t_n be defined by

$$t_n = [x^n]\,{}_2F_1\left(\begin{array}{c} a,b \\ a+b+\frac{1}{2} \end{array}\middle| x\right).$$

Via (1.4.3) we find a recurrence equation for the Taylor coefficients t_n, which we transform into a differential equation that is satisfied by its generating function.

```
In[52]:= de=RE2DE[{t[n+1]/t[n]==(n+a)*(n+b)/(n+a+b+1/2)/(n+1),
           t[0]==1},t[n],F[x]];
```

Now we derive the differential equation `de1`, satisfied by the function on the left of (2.7.2), via algebraic composition:

```
In[53]:= de1=ACompose[de,F[x]==4*x*(1-x),F[x]]

Out[53]= {8 a b F[x] + (-1 - 2 a - 2 b + 2 x + 4 a x + 4 b x)

                         2
     F'[x] + 2 (-x + x ) F''[x] == 0, F[0] == 1,

              8 a b
  F'[0] == -------------}
           1 + 2 a + 2 b
```

Similarly `de2` is satisfied by the function on the right of (2.7.2).

```
In[54]:= de2=RE2DE[{t[n+1]/t[n]==(n+2a)*(n+2b)/(n+a+b+1/2)/(n+1),
           t[0]==1},t[n],F[x]]
CanRE::denom:
   Warning. The input equation will be multiplied by its
    denominator.


Out[54]= {-8 a b F[x] + (1 + 2 a + 2 b - 2 x - 4 a x - 4 b x)

                        2
     F'[x] + 2 (x - x ) F''[x] == 0, F[0] == 1,

              8 a b
  F'[0] == -------------}
           1 + 2 a + 2 b
```

It is evident that the differential equations `de1` and `de2` define the same hypergeometric function.

2.7.5 RecurrenceEquationSubsequence (RESubsequence)

`RESubsequence[`*re*`,`*a*`[`*n*`],`*d*`*`*n*`+`*h*`]` gives a recurrence that is satisfied by a subsequence of the form $(a_{dn+h})_{n\geq 0}$ of every solution $(a_n)_{n\geq 0}$ of the input recurrence *re* in *a*`[`*n*`]`. *d* and *h* are assumed to be a positive and an arbitrary integer, respectively.

Example. (The American Mathematical Monthly, Problem 10473.) Prove that there are infinitely many positive integers m such that

$$s_m = \frac{1}{5 \cdot 2^m} \sum_{k=0}^{m} \binom{2m+1}{2k} 3^k$$

is an odd integer.

Suppose we already derived a recurrence for the sequence $(s_m)_{m \geq 0}$ (either by working with the procedures described above, or via Zeilberger's algorithm):

```
In[55]:= re={s[m]-4*s[m+1]+s[m+2]==0,s[0]==1/5,s[1]==1}

                                                     1
Out[55]= {s[m] - 4 s[1 + m] + s[2 + m] == 0, s[0] == -, s[1] == 1}
                                                     5
```

We look at the first terms of the sequence,

```
In[56]:= RE2L[re,s[m],10]

          1      19  71       989  3691        51409  191861
Out[56]= {-, 1, --, --, 53, ---, ----, 2755, -----, ------, 143207}
          5      5   5         5    5            5      5
```

and conjecture that all entries in the subsequence $(s_{3m+1})_{m \geq 0}$ are odd integers. This conjecture is easily verified:

```
In[57]:= RESubsequence[re,s[m],3m+1]

Out[57]= {s[m] - 52 s[1 + m] + s[2 + m] == 0, s[0] == 1, s[1] == 53}
```

Indeed, it is obvious that this recurrence (together with the given initial values) defines a sequence of odd integers.

2.7.6 `RecurrenceEquationShadow (REShadow)`

`REShadow[`re`,`a`[`n`]]` gives a recurrence that is satisfied by the shadow of any solution of the input recurrence, i.e., if the sequence $a_0, a_1, a_2, \ldots$, satisfying the input recurrence re in a`[`n`]`, can be extended to be valid for all $n \in \mathbf{Z}$, then this procedure gives a recurrence, which is satisfied by the sequence $a_0, a_{-1}, a_{-2}, \ldots$.

Example. The recurrence in this example defines a sequence, say $(a_n)_{n \geq 0}$, that starts with $1, 1, 2, \frac{5}{3}, \frac{9}{7}, \ldots$. If we extend the validity of this recurrence from $\mathbf{N}$ to the set of all integers, it is possible to come up with a recurrence that is satisfied by the sequence of elements with nonpositive indices:

```
In[58]:= REShadow[{a[n]+(n+1)*a[n+1]-(2*n+1)*a[n+2]==0,
                  a[0]==a[1]==1},a[n]]

RE2L::negative:
   Warning. The recurrence is extended to (some) negative integers.

Out[58]= {(-1 - 2 (-2 - n)) a[n] + (-1 - n) a[1 + n] + a[2 + n] == 0,

  a[0] == 1, a[1] == -1}
```

The output recurrence is satisfied by the sequence $(a_{-n})_{n\geq 0}$ with the first terms 1,–1, –4,–3,19,....

2.7.7 RecurrenceEquationInterlace (REInterlace)

`REInterlace[`re_1`,`re_2`,...,`re_k`,`a`[`n`]]` gives a recurrence that is satisfied by interlacing solutions of the recurrences $re_1, re_2, \ldots, re_k$. This means, if $(a_{1\,n})_{n\geq 0}, (a_{2n})_{n\geq 0}, \ldots,$ $(a_{k\,n})_{n\geq 0}$ satisfy the input recurrences (which are all given in a`[`n`]`), then the interlacement of these sequences starts with $a_{1\,0}, a_{20}, \ldots, a_{k\,0}, a_{1\,1}, \ldots, a_{k\,1}, a_{1\,2}$ a.s.o.

Example.([GKP94, Exercise 5.64]) Evaluate $s_n = \sum_{k=0}^{n} \binom{n}{k} / \lceil \frac{k+1}{2} \rceil$, given an integer $n \geq 0$.

We can rewrite this sum as

$$s_n = n! \sum_{k=0}^{n} \frac{a_k}{k!} \, \frac{1}{(n-k)!}, \tag{2.7.3}$$

where $(a_n)_{n\geq 0}$ is given by

$$a_n = \left\lceil \frac{n+1}{2} \right\rceil^{-1} = \begin{cases} \frac{1}{m+1}, & \text{if} \quad n = 2m, \qquad m \in \mathbf{N} \\ \frac{1}{m+1}, & \text{if} \quad n = 2m+1, \quad m \in \mathbf{N} \end{cases}.$$

Now, we get the recurrence `re` that is satisfied by the sequence $(a_n)_{n\geq 0}$, by interlacing two (identical) rational sequences:

```
In[59]:= re=REInterlace[a[n]==1/(n+1),a[n]==1/(n+1),a[n]]

CanRE::denom:
   Warning. The input equation will be multiplied by its
    denominator.
```

```
CanRE::denom:
   Warning. The input equation will be multiplied by its
    denominator.

                                2
Out[59]= {(2 + 3 n + n ) a[n] - 2 (3 + n) (4 + n) a[2 + n] +

                                                                 1
   (5 + n) (6 + n) a[4 + n] == 0, a[0] == 1, a[1] == 1, a[2] == -,
                                                                 2

          1
 a[3] == -}
          2
```

Next, (2.7.3) tells us how to compute a recurrence for $(s_n)_{n\geq 0}$:

```
In[60]:= REHadamard[RECauchy[REHadamard[re,{(n+1)*a[n+1]==a[n],a[0]==1},
         a[n]],{(n+1)*a[n+1]==a[n],a[0]==1},a[n]],{a[n+1]==(n+1)*a[n],
         a[0]==1},a[n]]

                           2
Out[60]= {4 (1 + n)  (2 + n) a[n] -

   8 (1 + n) (2 + n) (3 + n) a[1 + n] +

   (3 + n) (4 + n) (9 + 5 n) a[2 + n] -

                     2
   (4 + n) (5 + n)  a[3 + n] + (5 + n) (6 + n) a[4 + n] == 0,

                                               7
 a[0] == 1, a[1] == 2, a[2] == -, a[3] == 6}
                                               2
```

It remains to find closed form solutions of this recurrence. We apply Petkovšek's algorithm HYPER, which is a method to find all hypergeometric solutions of a holonomic recurrence (see [Pet92]), to obtain these solutions and, after some simplifications, we find that

$$s_n = \frac{2(2^{n+1} - 1)}{n+2}.$$

2.7.8 `HomogenousRecurrenceEquation (HomogenousRE)`

`HomogenousRE[`*re*`,`*a*`[`*n*`]]` gives a homogenous recurrence equation that is satisfied by any solution of the (possibly) inhomogenous recurrence *re* in *a*`[`*n*`]`.

Example. In some cases, for example, if we want to find hypergeometric solutions of a recurrence via Petkovšek's algorithm HYPER [Pet92], it is necessary to represent a sequence via a homogenous recurrence.

```
In[61]:= HomogenousRE[b[k]==k*b[k-2]+1/k,b[k]]

CanRE::denom:
  Warning. The input equation will be multiplied by its
    denominator.

                          2                      2
Out[61]= (-4 - 4 k - k ) b[k] + (9 + 6 k + k ) b[1 + k] +

  (2 + k) b[2 + k] + (-3 - k) b[3 + k] == 0
```

2.7.9 `HomogenousDifferentialEquation (HomogenousDE)`

`HomogenousDE[`*de*`,`*f*`[`*x*`]]` gives a homogenous differential equation that is satisfied by any solution of the (possibly) inhomogenous differential equation *de* in *f*`[`*x*`]`.

Example.

```
In[62]:= HomogenousDE[{f'[x]==1/(1+x^2),f[0]==0},f[x]]

CanDE::denom:
   Warning. The input equation will be multiplied by its
     denominator.

                                  2
Out[62]= {-2 x f'[x] - (1 + x ) f''[x] == 0, f[0] == 0, f'[0] == 1}
```

2.8 The Interface to the System

The procedures that were introduced in the previous sections are tools to work with and to manipulate holonomic power series and sequences. These tools are powerful enough to solve a lot of problems that can be stated in the holonomic universe. In particular, it is possible to check and prove holonomic identities. However, it is often not very convenient to do the proofs by just using the so far introduced procedures.

We illustrate this point via a the following standard example: Suppose we want to prove *Cassini's identity*

$$F_{n+1}F_{n-1} - F_n^2 = (-1)^n \qquad \text{for } n \in \mathbf{N}, \tag{2.8.1}$$

where F_n is the nth Fibonacci number given by (1.3.4). Using Mathematica together with the procedures introduced so far, one might check (and hence also prove) this identity as follows:

```
In[63]:= re={f[n]==f[n-1]+f[n-2],f[0]==0,f[1]==1}    (* f[n] *)

Out[63]= {f[n] == f[-2 + n] + f[-1 + n], f[0] == 0, f[1] == 1}

In[64]:= re1=RE2Subsequence[re,f[n],n+1]            (* f[n+1] *)

Out[64]= {-f[n] - f[1 + n] + f[2 + n] == 0, f[0] == 1, f[1] == 1}

In[65]:= re2=RE2Subsequence[re,f[n],n-1]            (* f[n-1] *)

 RE2L::negative:
   Warning. The recurrence is extended to (some) negative
    integers.

Out[65]= {-f[n] - f[1 + n] + f[2 + n] == 0, f[0] == 1, f[1] == 0}

In[66]:= lhs1=REHadamard[re1,re2,f[n]]              (* f[n+1]*f[n-1] *)

Out[66]= {f[n] - 2 f[1 + n] - 2 f[2 + n] + f[3 + n] == 0,

   f[0] == 1, f[1] == 0, f[2] == 2}


In[67]:= lhs2=REHadamard[                           (* -f[n]^2 *)
                REHadamard[re,re,f[n]],
                f[n]==-1,
             f[n]]
```

```
Out[67]= {f[n] - 2 f[1 + n] - 2 f[2 + n] + f[3 + n] == 0,

   f[0] == 0, f[1] == -1, f[2] == -1}

In[68]:= lhs3={f[n]==-f[n-1],f[0]==-1}              (* -(-1)^n *)

Out[68]= {f[n] == -f[-1 + n], f[0] == -1}


In[69]:= REPlus[                          (* f[n+1]*f[n-1]-f[n]^2-(-1)^n *)
            REPlus[lhs1,lhs2,f[n]],
            lhs3,
         f[n]]

Out[69]= {f[n] - 2 f[1 + n] - 2 f[2 + n] + f[3 + n] == 0,

   f[0] == 0, f[1] == 0, f[2] == 0}
```

The last recurrence, which is satisfied by the sequence $F_{n+1}F_{n-1} - F_n^2 - (-1)^n$, is of order three and, since the first three initial values are equal to zero, identity (2.8.1) is proved.

Considering the amount of work this (quite easy!) proof has required, one is probably faster in doing the job with a pencil on a sheet of paper. However, this example gives the motivation to use the generic programming facilities of Mathematica as follows: We represent a holonomic sequence or power series by a certain data structure. If the system knows how to carry out operations like addition, multiplication, etc., for an input that matches this structure, it is possible to use the symbols `+`, `*`, a.s.o., for these and other operations.

The package `GeneratingFunctions` provides tools to transform a recurrence or differential equation into this data structure (and vice versa) and tells Mathematica how to handle expressions containing the sequences and functions that are represented by such a structure.

2.8.1 `DefineSequence (DefineS)`

Let re be the Mathematica expression for a recurrence equation of the form (1.3.2) in a`[`n`]`. Let n_0 be the highest index that occurs in the (possibly empty) list of ICR's in re.

The procedure `DefineS[`re`,`a`[`n`]]` returns an internal representation of the sequence or (in the case that the initial conditions do not define a unique sequence) of a family of sequences that is/are defined via re. The output is given as

$$\texttt{RE[\{\{}q(n)\texttt{,}p_0(n)\texttt{,}\ldots\texttt{,}p_d(n)\texttt{\},\{}a_0\texttt{,}\ldots\texttt{,}a_{n_0}\texttt{\}\},}a\texttt{[}n\texttt{]]}. \tag{2.8.1}$$

If two sequences $(a_n)_{n\geq 0}$ and $(b_m)_{m\geq 0}$ are represented by the Mathematica variables A and B via A=`DefineS[`re_1`,`a`[`n`]]` and B=`DefineS[`re_2`,`b`[`m`]]`, respectively, then all the procedures that were introduced in the sections 2.5 and 2.7 and that have recurrence equations in the input, may be called without giving the recurrence variable a`[`n`]`or b`[`m`]`. For example `REPlus[`A`,`B`]` is a valid procedure call, even if the recurrences are given in different variables.

In the case that the input recurrence(s) is/are represented in the internal form (2.8.1), all the procedures also return the output recurrences in this form. In addition, the following univariate and bivariate operations are available:

`Delta[`A`]`	forward difference	$a_{n+1} - a_n$
`PSum[`A`]`	partial sum	$\sum_{k=0}^{n} a_k$
`Shift[`A`,`h`]`	shift	a_{n+h}, $h \in \mathbf{Z}$
A`^`h	hth power	a_n^h, $h \in \mathbf{N}$
A`+`B, A`-`B	sum, difference	$a_n \pm b_n$
A`*`B	termwise or Hadamard product	$a_n b_n$
A`==`B	evaluation of	$a_n = b_n$ for all $n \in \mathbf{N}$

Concerning the bivariate operations, one of the input sequences may also be a rational expression in n or m, respectively.

Examples. Using these additional tools, Cassini's identity (2.8.1) can be quickly proved as follows:

```
In[70]:= F=DefineS[re,f[n]]

Out[70]= RE[{{0, -1, -1, 1}, {0, 1}}, f[n]]

In[71]:= Shift[F,1]*Shift[F,-1]-F^2==
         DefineS[{f[n]==-f[n-1],f[0]==1},f[n]]

 RE2L::negative:
   Warning. The recurrence is extended to (some) negative
    integers.

Out[71]= True
```

In the following example we illustrate, how to prove a special function identity: Let $L_n^{(\alpha)}(x)$ be the nth *Laguerre polynomial*, which is recursively defined by

$$L_{n+2}^{(\alpha)}(x) = \frac{1}{n+2}\left(-(n+1+\alpha)L_n^{(\alpha)}(x) + (2n+3+\alpha-x)L_{n+1}^{(\alpha)}(x)\right),$$

where $L_0^{(\alpha)}(x) = 1$ and $L_1^{(\alpha)}(x) = 1+\alpha-x$. We want to prove the identity ([AS64], p. 783)

$$L_n^{(\alpha+1)}(x) = \frac{1}{x}\left((x-n)L_n^{(\alpha)}(x) + (\alpha+n)L_{n-1}^{(\alpha)}(x)\right). \tag{2.8.2}$$

This proof might be performed as follows (α will be replaced by `al`):

```
In[72]:= lag[al_,x_]:=DefineS[{(1+al+n)*l[n]+(-3-al-2*n+x)*l[1+n]
                          +(2+n)*l[2+n]==0,l[1]==1+al-x,l[0]==1},l[n]]

In[73]:=lag[al+1,x]==1/x((x-n)*lag[al,x]+(al+n)*Shift[lag[al,x],-1])
        //Simplify

RE2L::negative:
   Warning. The recurrence is extended to (some) negative integers.

EqualRE::IntegerRoots:
   Warning. The result is correct, provided that the polynomial
    $LeadingPolynomial=
                                    3                 2
    (3 + n) (4 + <<1>>) (al - al  + 7 x + <<7>> + 2 n x )

     contains no integer root greater  -1.

Out[73]= True
```

The global variable `$LeadingPolynomial` is the leading coefficient of a recurrence satisfied by the sequence which is obtained by subtracting the right hand side from the left hand side of the input equation. To verify that all the elements in this sequence are identical zero, it is sufficient to check that a finite number of, say n_0, initial values are 0. The number n_0 depends on the maximum integer for which `$LeadingPolynomial` vanishes, hence it remains to determine the nonnegative integer roots of `$LeadingPolynomial`.

We do not perform this step here. The problem is discussed in more detail in Chapter 3.

2.8.2 DefineFunction (DefineF)

Let de be the Mathematica expression for a differential equation of the form (1.2.2) in f`[`x`]`. Let n_0 be the highest index that occurs in the (possibly empty) list of ICD's in *de*.

The procedure `DefineF[`de`,`f`[`x`]]` returns an internal representation of the function or (in the case that the initial conditions do not define a unique function) of a family of functions that is/are defined via de. The output is given as

$$\texttt{DE[\{\{}q(x)\texttt{,}p_0(x)\texttt{,}\ldots\texttt{,}p_d(x)\texttt{\},\{}f(0)\texttt{,}\ldots\texttt{,}f^{(n_0)}(0)\texttt{\}\},}f\texttt{[}x\texttt{]]}. \tag{2.8.2}$$

If two power series $f(x)$ and $g(y)$ are represented by the Mathematica variables F and G via F`=DefineF[`de_1`,`f`[`x`]]` and G`=DefineF[`de_2`,`g`[`y`]]`, respectively, then all the procedures that were introduced in the sections 2.5 and 2.7 and that have differential equations in the input, may be called without giving the function variable f`[`x`]` or g`[`y`]`. For example `DEPlus[`F`,`G`]` is a valid procedure call, even if the equations are given in different variables.

In the case that the input differential equation(s) is/are represented in the internal form (2.8.2), the procedures also return the output differential equations in this form. In addition, the following univariate and bivariate operations are available:

`D[`F`]`	derivative	$f'(x)$
`Integrate[`F`]`	indefinite integral	$\int_0^x f(t)dt$
`Series[`F`,`n_0`]`	truncated series expansion	$\sum_{n=0}^{n_0} f^{(n)}(0)/n!\, x^n$, $n_0 \in \mathbf{N}$
F`^`h	hth power	$f(x)^h$, $h \in \mathbf{N}$
F`+`G, F`-`G	sum, difference	$f(x) \pm g(x)$
F`*`G	Cauchy product	$f(x)g(x)$
F`==`G	evaluation of:	$f(x) = g(x)$ for all $x \in \mathbf{K}$

Concerning the bivariate operations, one of the input functions may also be a rational expression in x or y, respectively.

Examples. We prove the identity $\sin(2x) = 2\sin(x)\cos(x)$: The first step is to define the sine and the cosine function:

```
In[74]:= si=DefineF[{f[x]+f''[x]==0,f[0]==0,f'[0]==1},f[x]]

Out[74]= DE[{{0, 1, 0, 1}, {0, 1}}, f[x]]

In[75]:= co=D[si]

Out[75]= DE[{{0, 1, 0, 1}, {1, 0}}, f[x]]
```

It remains to type in the specified equation:

```
In[76]:= ACompose[si,f==2x]==2*si*co

Out[76]= True
```

As in the case of sequences, the leading polynomial of a differential equation that is satisfied by the difference of the left and the right hand side of the input equation, may cause troubles in the proof process. If this leading polynomial vanishes at $x = 0$, then zero recognition can not be performed directly. In this case the proof must be done by comparing the power series coefficients, i.e., one has to go via the recurrences.

2.8.3 `RecurrenceEquationOut (REOut)`

Let A be a holonomic sequence that is given in the internal representation (2.8.1). The procedure `REOut[`A`]` outputs the recurrence equation in a format that is readable by other Mathematica procedures like, for instance, `RSolve`.

Example.

```
In[77]:= A=DefineS[{a[n]==4*a[n-1]+(n^2-4)*a[n-2],
                   a[0]==1,a[1]==2},a[n]]

                           2
Out[77]= RE[{{0, -4 n - n , -4, 1}, {1, 2}}, a[n]]

In[78]:= REOut[PSum[A]]

                             2       3
Out[78]= {(20 + 34 n + 16 n  + 2 n ) a[n] +

                       2       3
   (-52 - 70 n - 28 n  - 3 n ) a[1 + n] +

                      2     3
   (12 + 30 n + 12 n  + n ) a[2 + n] + (24 + 7 n) a[3 + n] +

   (-4 - n) a[4 + n] == 0, a[0] == 1, a[1] == 3, a[2] == 11,

 a[3] == 53}
```

2.8.4 `DifferentialEquationOut (DEOut)`

Let F be a holonomic power series that is given in the internal representation (2.8.2). The procedure `DEOut[`F`]` outputs the differential equation in a format that is readable by other Mathematica procedures like, for instance, `DSolve`.

Example. Let `si` represent the function $\sin(x)$. The output line in this example is a differential equation that is satisfied by $\sin(x^2 - x)$.

```
In[79]:= DEOut[ACompose[si,f==x*(x-1)]]

                                   2       3
Out[79]= {(-1 + 6 x - 12 x  + 8 x ) f[x] - 2 f'[x] +

    (-1 + 2 x) f''[x] == 0, f[0] == 0, f'[0] == -1}
```

Chapter 3

Guesses, Proofs and Ore Polynomials

3.1 Introduction

Holonomic functions and sequences are closed under most standard unary, binary and n-ary operations. Unfortunately the closure properties discussed in Section 1.4 do not include objects like reciprocals, square roots or other rational powers of functions or sequences. Nevertheless, if we follow an idea presented in [BP92], an approach that is based on "guessing" might be able to handle problems involving these object types.

Moreover, it is sometimes possible to guess a "nice" equation that is satisfied by a given sequence or function. We call a recurrence or differential equation nice if it can be solved algorithmically. The process of guessing is discussed in Section 3.2.

To solve a problem via guessing, it is necessary to prove that the guessed (differential or recurrence) equation is correct. The theoretical background for some proof methods is given in Section 3.3, where we translate parts of the holonomic universe into operator algebra language or, more precisely, into the language of Ore polynomials.

Finally, we present some methods for proving holonomic identities in Section 3.4.

3.2 Guessing

We consider the following example.

Example 3.2.1 (The American Mathematical Monthly, Problem 10356.) Let X_n be defined by $X_0 = 0$, $X_1 = 1$, $X_2 = 0$, $X_3 = 1$ and for $n \geq 1$

$$X_{n+3} = \frac{(n^2+n+1)(n+1)}{n} X_{n+2} + (n^2+n+)\, X_{n+1} + \frac{n+1}{n} X_n .$$

Prove that X_n is the square of an integer for $n \geq 0$.

We want to illustrate how this problem can be solved by using my Mathematica package `GeneratingFunctions`, which is introduced in Chapter 2: We define the sequence `X` via the given recurrence and compute the square root of the first terms:

```
In[1]:= X=DefineS[{x[n+3]==(n^2+n+1)*(n+1)/n*x[n+2]+
        (n^2+n+1)*x[n+1]-(n+1)/n*x[n],
        x[0]==x[2]==0,x[1]==x[3]==1},x[n]]

                                 2    3                 2    3
Out[1]= RE[{{0, 1 + n, -n - n  - n , -1 - 2 n - 2 n  - n , n},

   {0, 1, 0, 1}}, x[n]]

In[2]:= Sqrt[RE2L[X,20]]

Out[2]= {0, 1, 0, 1, 2, 7, 30, 157, 972, 6961, 56660, 516901,

  5225670, 57999271, 701216922, 9173819257, 129134686520,

  1946194117057, 31268240559432, 533506283627401,

  9634381345852650}
```

Hence it looks as if the elements in the sequence are indeed squares of integers. We check this conjecture by "guessing" a recurrence that is satisfied by the first elements of the sequence $(a_n)_{n\geq 0}$, where $a_n^2 = X_n$ for $n \in \mathbf{N}$:

```
In[3]:= GuessRE[%,a[n]]

Out[3]= {{-a[n] - n a[1 + n] + a[2 + n] == 0, a[0] == 0,

   a[1] == 1}, ogf}
```

It remains to check that the elements in $(X_n)_{n\geq 0}$ are the squares of the elements in $(a_n)_{n\geq 0}$ (which is a sequence of integers):

```
In[4]:= A=DefineS[%[[1]],a[n]]

Out[4]= RE[{{0, -1, -n, 1}, {0, 1}}, a[n]]

In[5]:= A^2==X

Out[5]= True
```

□

The crucial step in the solution of this problem is to come up with a recurrence for the square root of the original sequence $(X_n)_{n\geq 0}$. This problem can not be solved directly within the holonomic universe, since the closure properties discussed in Section 1.4.3 do not include (multiplicative) powers of the form $(a_n^r)_{n\geq 0}$, where $(a_n)_{n\geq 0}$ is a holonomic sequence and r is a rational number. (In our example we had $r = 1/2$.)

However, given a holonomic sequence $(a_n)_{n\geq 0}$, an approach that is performed in two basic steps, might help to compute a recurrence for the sequence $(b_n)_{n\geq 0} = (a_n^r)_{n\geq 0}$, where $r \in \mathbf{Q}$:

1. *(Guess a holonomic equation.)*
   We use the recursive definition of $(a_n)_{n\geq 0}$ to compute a number of initial values $b_0 = {a_0}^r, \ldots, b_{n_0} = {a_{n_0}}^r$. Next we try to "guess" a holonomic recurrence equation that is satisfied by the initial terms of the sequence $(b_n)_{n\geq 0}$. To do so, the order and the degree of the desired recurrence must be fixed. Then indeterminates for the coefficients of the polynomials in the recurrence are used to build a system of linear equations. A solution of this system might contain a candidate for the recurrence, we are looking for.

   The problem here is, that no bounds for the order and the degree of a holonomic recurrence that is satisfied by $(b_n)_{n\geq 0}$ are known, nor do we have a (general) guarantee that this sequence is holonomic at all. Hence we do not know up to which order and degree, a recurrence should be searched and it might happen that the search terminates too early and that the approach fails. However, if we succeed, it is no problem to verify (or falsify) whether the guessed equation is correct.

2. *(Verify the guessed equation.)*
   If the search for a holonomic recurrence succeeded, we have to prove that the guessed equation is correct. This may be done as follows: Let $r = p/q$, with $p \in \mathbf{Z}$, $q \in \mathbf{N}$ and let $(b_n)_{n\geq 0}$ be the unique solution of the guessed recurrence equation. To prove
   $$a_n^{p/q} = b_n \text{ for } n \in \mathbf{N},$$
   we can show
   $$a_n^p = b_n^q \text{ for } n \in \mathbf{N}, \text{ if } p \geq 0 \text{ or} \tag{3.2.1}$$

$$1 = a_n^{-p} b_n^q \text{ for } n \in \mathbf{N}, \text{ if } p < 0, \tag{3.2.2}$$

which is no problem, if we consider Theorem 1.4.3. (To complete the proof, we also have to check a finite number of initial values of $(a_n^{p/q})_{n\geq 0}$ and $(b_n)_{n\geq 0}$ to agree, since (3.2.1) and (3.2.2), respectively, are necessary but not sufficient conditions.)

In Example 3.2.1 the final step proves $a_n^2 = X_n$, for $n \in \mathbf{N}$. This is done by computing a recurrence equation satisfied by the sequence $(a_n^2 - X_n)_{n\geq 0}$ and by checking a sufficient number of initial values of the two sequences to be equal. In Section 3.4 we also discuss different methods for proving identities of holonomic sequences.

It is evident that an approach that is analogous to the one, which is described above, may be used to compute holonomic differential equations that are satisfied by rational powers of holonomic power series.

We have seen that the process of guessing is useful for working with transformations of holonomic functions and sequences, if we can prove the validity of the guessed equations by using some kind of "inverses" of these transformations.

Guessing might also help in the following situation: Given a holonomic power series or sequence via a holonomic equation, it is sometimes desirable to know whether this holonomic object also satisfies a differential or recurrence equation of a certain form, which allows us to extract a closed expression for the sequence or function. Of special interest in this case are m-hypergeometric recurrences, which are a generalisation of the classical hypergeometric sequences introduced in Example 1.3.1 and Example 1.4.1: A sequence $(t_n)_{n\geq 0}$ is *m-hypergeometric* if there is a rational function $r(x) \in \mathbf{K}(x)$ and an integer $m \geq 1$ such that for all $n \in \mathbf{N}$

$$\frac{t_{n+m}}{t_n} = r(n). \tag{3.2.3}$$

In this case, the sequence $(t_n)_{n\geq 0}$ is the interlacement (see Corollary 1.4.6.e) of m hypergeometric sequences, and we can—at least in principle—give an "explicit" expression for the nth element. We illustrate this point by an example:

Example 3.2.2 (See [Han75], (5.8.17), p. 42.) Suppose, we want to compute a closed form for the coefficients a_n of the power series

$$f(x) = \sum_{n\geq 0} a_n x^n = \frac{1}{8}\big((x^{-2} - 1)\log\left(\frac{1-x}{1+x}\right) + 2(1 + x^{-2})\arctan(x)\big).$$

If we have a holonomic knowledge base that contains differential equations satisfied by $\arctan(x)$ and $\log((1-x)/(1+x))$, we can apply the theory discussed in Section 1.4.3 and find that the sequence $(a_n)_{n\geq 0}$ is a solution of the recurrence

$$-n(n+1)(n+2)(n+3)a_n + 2(n+4)(n+6)(n^2 + 10n + 33)a_{n+4} -$$

$$(n+7)(n+8)(n+9)(n+10)a_{n+8} = 0. \tag{3.2.4}$$

It turns out that the Mathematica procedure `RSolve` can not solve this recurrence, so we have to enter a little bit of human insight: An inspection of the recurrence and some initial values show that

$$f(x) = \sum_{n\geq 0} a_n x^n = \sum_{n\geq 0} b_n x^{4n+1},$$

where $b_n = a_{4n+1}$, and where the sequence $(b_n)_{n\geq 0}$ satisfies the second order recurrence

$$(n+1)(2n+1)(4n+1)(4n+3)b_n - (4n+5)(4n+7)(4n^2+12n+11)b_{n+1}$$
$$(n+2)(2n+5)(4n+9)(4n+11)b_{n+2}. \tag{3.2.5}$$

(Note that (3.2.5) is obtained from (3.2.4) by substituting $4n+1$ for n in the polynomial coefficients of (3.2.4).) Next we apply (3.2.5) to compute some initial values of $(b_n)_{n\geq 0}$. These values may be used to guess another holonomic recurrence equation for the sequence. This guessed equation should have a special form that allows us to extract a closed expression for the coefficients b_n.

Indeed, a guessing procedure (as performed by the function `GuessRE` introduced in Section 2.6.1) finds:

$$(-16n^2 - 16n - 3)b_n + (16n^2 + 48n + 35)b_{n+1} = 0 \tag{3.2.6}$$

This recurrence equation is easily solved and, after we have proved that the guessed recurrence is correct, we conclude that

$$f(x) = \sum_{n\geq 0} \frac{1}{(4n+1)(4n+3)} x^{4n+1}$$

□

Remark. Due to the fact that first order (inhomogenous) differential equations are used to represent the functions $\log((1-x)/(1+x))$ and $\arctan(x)$, the recurrences (3.2.4) and (3.2.5) in the previous example have nice shapes that help in solving the problem. Moreover it would also be possible to find all hypergeometric solutions of (3.2.5) by performing Petkovšek's algorithm HYPER [Pet92]. So, the process of "guessing" is not really necessary in this case.

However, if we rewrite $\log((1-x)/(1+x))$ as $\log(1-x) - \log(1+x)$ and represent both summands by homogenous second order differential equations, we finally come up with a recurrence of order 16 and degree 5. We do not recommend to try HYPER on this recurrence. In this case it seems as if "guessing" is the only way to solve the problem. In addition, this example shows that the sequence $(a_n)_{n\geq 0}$ does not satisfy a unique recurrence and that the "size" of the final recurrence depends on the differential equations that are used from the holonomic knowledge base. □

3.3 Ore Polynomials

In this and in the following section we translate the theory of holonomic functions and sequences into operator language. More precisely, we translate holonomic objects (and some of the operations that may be applied to them) into the operator language of *Ore polynomial rings*:

For the rest of this chapter we make the following assumptions:

1. The mapping $\sigma : \mathbf{K} \to \mathbf{K}$ is an injective endomorphism.
2. The mapping $\delta : \mathbf{K} \to \mathbf{K}$ has the property that for all $a, b \in \mathbf{K}$:
   (a) $\delta(a+b) = \delta(a) + \delta(b)$ and
   (b) $\delta(ab) = \sigma(a)\delta(b) + \delta(a)b$.

   We call δ to be a *pseudo-derivation with respect to* σ.

Definition 3.3.1 (Ore Polynomial Ring) An *Ore polynomial ring* w.r.t. σ and δ is the noncommutative (or skew) ring of polynomials in X over $\mathbf{K}$, where addition is defined as usual, and multiplication is given by

$$Xa = \sigma(a)X + \delta(a) \qquad \text{for } a \in \mathbf{K}. \tag{3.3.1}$$

This polynomial ring is denoted by $\mathbf{K}[X, \sigma, \delta]$. It's members are called *Ore polynomials*.

Note that any Ore polynomial can be written uniquely in expanded form $\sum_{k=0}^{d} c_k X^k$, $c_k \in \mathbf{K}$; relation (3.3.1) serves to represent the noncommutative product of two Ore polynomials in this canonical form.

The canonical form can be achieved by recursively applying the associativity rule for monomials:

$$(aX^n)(bX^m) = (aX^{n-1})(Xb)X^m = (aX^{n-1})(\sigma(b)X^{m+1} + \delta(b)X^m), \tag{3.3.2}$$

where $a, b \in \mathbf{K}$, $m, n \in \mathbf{N}$ and $n \geq 1$. Then two Ore polynomials are multiplied by carrying over the distributivity law into the noncommutative world of the Ore polynomials.

The following examples show that holonomic differential and recurrence equations correspond to elements in certain Ore polynomial rings.

Example 3.3.1 Let δ be the derivation operator on $\mathbf{K}(x)$ w.r.t x, and let $\mathbf{1}$ be the identity on $\mathbf{K}$. Then $\mathbf{K}(x)[D, \mathbf{1}, \delta]$ is the ring of polynomial differential operators. Let $A = \sum_{k=0}^{d} a_k(x)D^k$ and $B = \sum_{k=0}^{e} b_k(x)D^k$ be two Ore Polynomials in this ring. Using the Leibniz rule

$$D^n b = \sum_{k=0}^{n} \binom{n}{k} b^{(k)} D^{n-k}, \text{ for } b \in \mathbf{K}(x), n \in \mathbf{N}$$

we derive that the product of A and B is given by

$$AB = \Big(\sum_{i=0}^{d} a_i D^i\Big)\Big(\sum_{i=0}^{e} b_i D^i\Big) = \sum_{i=0}^{d}\sum_{j=0}^{e} a_i (D^i b_j) D^j = \sum_{i=0}^{d}\sum_{j=0}^{e}\sum_{k=0}^{i} \binom{i}{k} a_i b_j^{(k)} D^{i+j-k}. \tag{3.3.3}$$

It is obvious that multiplication is not commutative, for instance, we have the commutation rule

$$Dx = xD + 1. \tag{3.3.4}$$

AB may be regarded as the composition of the two differential operators A and B. □

Example 3.3.2 Let σ be the automorphism $\sigma : n \to n{+}1$ on $(\mathbf{K}_n)$; the ring $(\mathbf{K}_n)[E,\sigma,\mathbf{0}]$ contains the set of linear recurrence operators with polynomial coefficients. For $A = \sum_{k=0}^{d} a_k(n)E^k$ and $B = \sum_{k=0}^{e} b_k(n)E^k$, we have the product

$$AB = \sum_{k=0}^{d+e} \Big(\sum_{j=0}^{k} a_j(n) b_{k-j}(n+j)\Big) E^k. \tag{3.3.5}$$

Again this ring is noncommutative, for instance, we have

$$En = (n+1)E. \tag{3.3.6}$$

□

Remark. It is possible to define an Ore polynomial ring over a commutative ring rather than over a field. Subsequently we will sometimes work with the restriction of $(\mathbf{K}_n)[E,\sigma,\mathbf{0}]$ to $[\mathbf{K}_n][E,\sigma,\mathbf{0}]$, (where σ is defined as in Example 3.3.2) and abbreviate the domains with $(\mathbf{K}_n)[E]$ and $[\mathbf{K}_n][E]$, respectively.

It is the purpose of this section to introduce the amount of theory that is required to discuss and compare different techniques for proving identities of holonomic sequences and power series. Since proofs of the latter ones can always be transformed to proving identities of sequences (see Theorem 1.4.1), we now focus our considerations on the ring of linear recurrence operators. General discussions may be found in [Ore33],[BP94] or in [Li96]. A thorough treatment of multivariate Ore polynomial rings is presented in [CS].

From now on we assume that σ is defined as in Example 3.3.2. Let A and B be two nonzero polynomials in $(\mathbf{K}_n)[E]$ with the respective degrees d and e in E. Without loss of generality we assume that $d \geq e$. Let a and b be the *leading coefficient* (lc) of A and B, respectively. If we set Q_0 to be the monomial

$$Q_0 = \frac{a(n)}{b(n+d-e)} E^{d-e},$$

it is easily seen that the leading monomial of Q_0B is aE^d, which implies that $A - Q_0B$ has degree less than d. In anology to the case of the usual commutative polynomials, we can use this observation to perform *right Euclidean division* as follows:

Algorithm: Right Euclidean division
Input: $A, B \in (\mathbf{K}_n)[E]$, with $\deg(A) = d$, $\deg(B) = e$
Output: $Q, R \in (\mathbf{K}_n)[E]$ such that $A = QB + R$, with $\deg(R) < e$

let $R = A$; *let* $Q = 0$;
let $d' = d$; *let* $b = \mathrm{lc}(B)$;
while $\deg(R) \geq e$ *do*
  let $Q_0 = (\mathrm{lc}(R)/\sigma^{d'-e}b)\ E^{d'-e}$;
  let $R = R - Q_0B$;
  let $d' = \deg(R)$;
  let $Q = Q + Q_0$;
end while;
return Q, R;

We call Q to be the *right quotient* of A and B, R is the *right remainder* of these two polynomials. If $R = 0$, then B is a *right divisor* of A.

Definition 3.3.2 For $k \in \mathbf{N}\backslash\{0\}$ and $r \in (\mathbf{K}_n)$ the *kth rising factorial* $[r]^{\overline{k}}$ is defined as

$$[r]^{\overline{k}} = \prod_{i=0}^{n-1} \sigma^i r = \prod_{i=0}^{n-1} r(n+i).$$

$[r]^{\overline{0}}$ is defined to be 1.

If we want to work in $[\mathbf{K}_n][E]$ only, we have to define pseudo division in this domain: Let $A, B \in [\mathbf{K}_n][E]$ have the respective degrees d and e with $d \geq e$, then the polynomials Q and R with $\deg(R) < e$, $b = \mathrm{lc}(B)$ and

$$[b]^{\overline{d-e+1}}A = QB + R$$

are called the *right pseudo quotient* ($Q = \mathrm{rpquot}(A, B)$) and the *right pseudo remainder* ($R = \mathrm{rprem}(A, B)$) of A and B.

A *greatest common right divisor* (gcrd) of A and B is a polynomial of highest degree that divides both A and B on the right. A gcrd of A and B can be computed by applying an analog to the Euclidean algorithm.

Running the extended Euclidean algorithm (with right division or right pseudo division) gives a *least common left multiple* (lclm) of A and B, which is a left multiple of least order of both A and B. One can show (see for instance [BP94]) that this algorithm does compute a nonzero multiple of least order. Moreover, the following relation between the degrees of the polynomials holds: Let G be a gcrd and let L be an lclm of A and B. Then

$$\deg(A) + \deg(B) = \deg(G) + \deg(L). \tag{3.3.7}$$

A proof for this result can be found in [Ore33].

It is well known that during the computation of a gcrd via the Euclidean algorithm, the coefficients of the (pseudo) remainders grow rather fast and that extraneous factors might be cancelled in order to keep the coefficients small. In commutative algebra, the *subresultant algorithm* by G.E. Collins (see e.g. [BT71]) slows down this growth, though still no greatest common divisors of the coefficients need to be computed.

This subresultant algorithm has been recently generalised to the noncommutative case of Ore polynomial rings by Ziming Li [Li96]. It would lead too far to discuss here the theory behind this generalisation in detail, instead we only want to present here the relevant algorithm [1].

Theorem 3.3.1 (Subresultant Algorithm) Let $A, B \in [\mathbf{K}_n][E]$, with $\deg(A) = d$, $\deg(B) = e$, $d \geq e$. Let $A_1, A_2, \ldots, A_k \neq 0, A_{k+1} = 0$ be a sequence that is computed as follows: Initialize

$$A_1 = A, \quad A_2 = B, \quad a_1 = 1, \quad a_2 = \mathrm{lc}(A_2), \quad b_1 = 1, \quad b_2 = [\mathrm{lc}(B)]^{\overline{d-e}}, \quad l_2 = d - e + 1.$$

For $i \geq 3$ let

$$l_i = \deg(A_{i-1}) - \deg(A_i) + 1, \qquad a_i = \mathrm{lc}(A_i), \qquad b_i = [a_i]^{\overline{l_i - 1}} / [\sigma b_{i-1}]^{\overline{l_i - 2}}$$

and

$$A_i = \mathrm{rprem}(A_{i-2}, A_{i-1}) / e_i,$$

where

$$e_i = (-1)^{l_{i-1}} [\sigma b_{i-2}]^{\overline{l_{i-1} - 1}} a_{i-2}.$$

Then for $1 \leq i \leq k$ we have $A_i \in [\mathbf{K}_n][E]$. Moreover A_k is a gcrd of A and B.

Proof. See [Li96, Lemma 1.4.6 and Theorem 1.4.7], where the proof is given for general Ore polynomials rings over commutative rings. □

Computing a least common left multiple of two Ore polynomials is in general much more time consuming than computing just a greatest common right divisor. This is due to the fact that the extended Euclidean algorithm requires two additional multiplications of Ore polynomials in every step of a while-loop, where the coefficients of the polynomials

[1] Ziming Li and István Nemes provide a modular algorithm to compute gcrd's in the case that $\mathbf{K} = \mathbf{Z}$. See [Li96, Chapter 3] or [LN96].

that are involved in these multiplications grow rather fast during the execution of the algorithm. Theorem 3.3.1 may also be used to slow the growth of these coefficients. (See also [Li96, Theorem 2.3.6].)

If we want to prove holonomic identities we are very often just interested in the leading coefficient of an lclm. The following corollary is useful in this case.

Corollary 3.3.2 Let $A, B \in [\mathbf{K}_n][E]$ with $\deg(A) = d$ and $\deg(B) = e$. Let g be the degree of the gcrd's of A and B. Then there is a lclm $L \in [\mathbf{K}_n][E]$ of A and B with

$$\mathrm{lc}(L) = (\sigma^{d-g}\mathrm{lc}(B))(\sigma^{e-g}\mathrm{lc}(A))\overline{b}, \tag{3.3.8}$$

where $\overline{b} = b_k$, which is computed as in Theorem 3.3.1.

Proof. The proposition follows from [Li96, Lemma 1.4.6, Theorem 1.4.7, Proposition 2.2.3 and Corollary 2.3.4]. □

3.4 Three Methods to Prove Holonomic Identities

We are going to establish a link between holonomic sequences and the Ore polynomial ring $[\mathbf{K}_n][E]$:

Definition 3.4.1 Let $A = \sum_{i=0}^{d} p_i(n)E^i \in [\mathbf{K}_n][E]$ and $\mathbf{a} = (a_n)_{n\geq 0} \in \mathbf{K}^{\mathbf{N}}$. Then A induces an action on $\mathbf{a}$ as follows:

$$A\mathbf{a} : \mathbf{N} \to \mathbf{K}, \qquad n \mapsto \sum_{i=0}^{d} p_i(n)a_{n+i}.$$

If $A\mathbf{a} = 0$ and $A \neq 0$, we say that A *annihilates* $\mathbf{a}$ or $\mathbf{a}$ *is a zero of* A.

It is now evident that holonomic sequences are exactly those sequences that are annihilated by some (nonzero) polynomial in $[\mathbf{K}_n][E]$. Conversely, every holonomic recurrence corresponds to an Ore polynomial in $[\mathbf{K}_n][E]$ that annihilates the solutions of this recurrence.

Example 3.4.1 Let $\mathbf{c} = (C_n)_{n\geq 0}$ be the sequence of Catalan numbers, defined by (1.4.8). In operator language, we would say that $A\mathbf{c} = 0$, where

$$A = 2(1 + 2n) - (2 + n)E.$$

□

Theorem 3.4.1 For $\mathbf{a} = (a_n)_{n\geq 0} \in \mathbf{K}^{\mathbf{N}}$ and $A, B \in [\mathbf{K}_n][E]$ the following propositions hold:

(a) $(AB)\mathbf{a} = A(B\mathbf{a})$.

(b) If $A\mathbf{a} = 0$, then $(BA)\mathbf{a} = 0$.

(c) If G is a gcrd of A and B, then $A\mathbf{a} = B\mathbf{a} = 0$ if and only if $G\mathbf{a} = 0$

Proof. See [BP94]. □

From Theorem 3.4.1 we immediately deduce that a least common left multiple L of two operators A and B annihilates both the zeros of A and the zeros of B. Using the terminology introduced in Chapter 1, we would say that L corresponds to a homogenous holonomic recurrence equation whose solution space contains all linear combinations of sequences $\mathbf{a}$ and $\mathbf{b}$, where $\mathbf{a}, \mathbf{b}$ satisfy the recurrences that correspond to A and B, respectively.

Lemma 3.4.2 (**Zero condition**) Let $\mathbf{a}$ be a zero of $A \in [\mathbf{K}_n][E]$ with $\deg(A) = d$. Let p be the leading coefficient of A. We define the integer n_0 as follows: If $p(n) \neq 0$ for all $n \in \mathbf{N}$, then $n_0 := d - 1$. Otherwise $n_0 := \max\{n \in \mathbf{N} | p(n) = 0\} + d$. Then

$$\mathbf{a} = 0 \qquad \text{if and only if} \qquad a_n = 0, \quad \text{for each} \quad n \in \{0, 1, \ldots, n_0\}.$$

Proof. Trivial. □

Convention. For the rest of this section, we assume that the two holonomic sequences $\mathbf{a}$ and $\mathbf{b}$ are annihilated by the operators A and B, respectively, with $\deg(A) = d$ and $\deg(B) = e$.

Lemma 3.4.2 and Theorem 1.4.3 suggest the following method to prove $(a_n)_{n\geq 0} = (b_n)_{n\geq 0}$:

Method 1. Compute a lclm, say C, of A and B. (Note that C annihilates both $\mathbf{a}$ and $\mathbf{b}$ as well as any linear combination of these sequences, in particular C annihilates $\mathbf{a} - \mathbf{b}$. C can be computed either by an extended Euclidean algorithm, where Theorem 3.3.1 could be applied, or by following the constructive proof of Theorem 1.4.3.) According to Lemma 3.4.2 we can verify or falsify the identity by comparing a number of initial terms of $(a_n)_{n\geq 0}$ and $(b_n)_{n\geq 0}$. This number depends on the degree of C in E and on the integers for which the leading coefficient of C vanishes.

Example 3.4.2 Let $(a_n)_{n\geq 0}$ and $(b_n)_{n\geq 0}$ be solutions of the respective recurrences

$$11(n+1)a_n - 10a_{n+1} = 0 \qquad \text{and}$$

$$(n+4)b_n - b_{n+1} = 0.$$

The sequence $(c_n)_{n\geq 0}$ with $c_n = a_n - b_n$ satisfies the second order recurrence

$$11(n-28)(n+1)(n+4)c_n + 3(-7n^2+179n+586)c_{n+1} + 10(n-29)c_{n+2} = 0.$$

To prove $(c_n)_{n\geq 0} = 0$, it is sufficient (and also necessary) to check if $c_0 = c_1 = \cdots = c_{31} = 0$. □

It is tempting to believe that a quick method to prove a holonomic identity might be as follows: Compute an upper bound for the maximum integer root of C, where this bound depends on the degrees and coefficients of A and B only, which means that no (extended) Euclidean algorithm has to be performed.

However, at the time this thesis is written, no suitable bounds are known. Lily Yen [Yen93] gave analogous bounds for the case of proofs of hypergeometric sum identities. But these bounds are—even for small examples—extremely high and it seems, as if useful bounds (i.e., bounds that save time in the process of proving) can hardly be found.

Nevertheless it is not necessary to compute least common left multiples. We present two methods that basically work with greatest common right divisors.

Method 2. (Reported in private communication to the author by Marko Petkovšek in a slightly different form.) Let G be a gcrd of A and B. By right pseudo division of G and A we compute F such that $pA = FG$, where p divides $\overline{[\mathrm{lc}(G)]^{d-\deg(G)+1}}$. Let $\mathbf{c} = (c_n)_{n\geq 0} = G\mathbf{a}$. Now we have

$$F\mathbf{c} = FG\mathbf{a} = pA\mathbf{a} = 0.$$

Hence the sequence $\mathbf{c}$ is annihilated by F and, following Lemma 3.4.2, we can check whether $\mathbf{c} = 0$ by inspecting some starting terms of this sequence.

If $\mathbf{c} = 0$, then $G\mathbf{a} = 0$ and, since B is a left multiple of G, we get $B\mathbf{a} = 0$. Now we know that B annihilates both $\mathbf{a}$ and $\mathbf{b}$, therefore we also have

$$B(\mathbf{a} - \mathbf{b}) = 0.$$

It remains to check a finite number of initial values of $\mathbf{a}$ and $\mathbf{b}$ to agree, where, according to Lemma 3.4.2, this number depends on the order of B and on the maximum integer root of $\mathrm{lc}(B)$.

Example 3.4.3 Let $\mathbf{a}$ and $\mathbf{b}$ be two sequences that are annihilated by the operators A and B, respectively, where

$$A = 2n(1+3n) + (4+19n-n^2)E + (-3-n-n^2)E^2 + 2(-1+n)E^3,$$

$$B = 3(1+3n) + (22+n+3n^2)E + 2(2-n)(1+n)E^2$$

and both sequences start with $a_0 = b_0 = 0, \ldots, a_3 = b_3 = 0, a_4 = b_4 = 1$. We compute G, which is a gcrd of A and B:

$$G = 3n + 1 - 2(n-3)E.$$

By right division of A and G we find

$$F = 2n + (n+1)E - E^2,$$

where $A = FG$. Now $\mathbf{c} = G\mathbf{a}$, which is a zero of F starting with $c_0 = c_1 = 0$, and we deduce that $\mathbf{c} = 0$. It follows that $\mathbf{a}$ is annihilated by B, and after observing that the first four values of $\mathbf{a}$ and $\mathbf{b}$ agree, we can conclude that $\mathbf{a} = \mathbf{b}$. □

Method 3. Compute a gcrd of A and B via the subresultant algorithm (Theorem 3.3.1). Let g be the degree of this gcrd. Then every lclm of A and B has degree $d+e-g$. Moreover, we can apply Corollary 3.3.2 and conclude that there is a lclm of A and B, whose leading coefficient, say p, is given by (3.3.8).

Let n_1 be the maximum integer root of $p(n)$. (If p has no integer roots we set $n_1 = -1$.) By Lemma 3.4.2, we can conclude that $\mathbf{a} = \mathbf{b}$, if (and only if) these sequences agree up to index $n_0 = n_1 + d + e - g$, i.e., if $a_0 = b_0, a_1 = b_1, \ldots, a_{n_0} = b_{n_0}$.

Example 3.4.4 Let

$$A = 2(1+2n)(4+3n) + (1+n)(19+18n+4n^2)E + (3+n)(4+n)E^2 +$$
$$(-3+2n)(13+3n)E^3 + (5+n)(6+n)(-3+2n)E^4$$

and

$$B = -4 - 3n - (2+n)(3+n)E + (10+3n)(4+n^2)E^2 + (4+n)(5+n)(4+n^2)E^3.$$

We apply the subresultant algorithm (Theorem 3.3.1) to compute a gcrd of A and B, which has degree one. As a byproduct we get

$$\overline{b} = (2+n)(3+n)^2(4+n)^2(5+n)^2(6+n)^2(7+n)$$
$$(1906 + 6808n + 8623n^2 + 6982n^3 + 4203n^4 + 1832n^5 + 604n^6 + 128n^7 + 16n^8).$$

Now we can conclude that there is an operator $L \in [\mathbf{K}_n][E]$ of degree 6 that annihilates the zeros of A as well as the zeros of B, and whose leading coefficient is given by

$$\mathrm{lc}(L) = (\sigma^3 \mathrm{lc}(B))(\sigma^2 \mathrm{lc}(A))\overline{b}.$$

We can easily see that this polynomial does not have any nonnegative integer root. Hence, if we want to check whether two sequences that are annihilated by A and B, respectively, are identical, it is sufficient to check the first 6 elements of the two sequences to agree. □

We know that an lclm, say L, of A and B has degree less or equal $d+e$. We have also seen the number of elements of $\mathbf{a}$ and $\mathbf{b}$ that have to be compared, if we want to prove $\mathbf{a} = \mathbf{b}$, depends on the degree of L and on the nonnegative integer roots of $\mathrm{lc}(L)$. In Example 3.4.2 we had the case that the L vanished at $n = 29$, though the leading coefficients of both A and B were constants. Hence, it would be necessary to check whether 32 initial values of A and B agree.

Now the question arises, whether any two sequences that are not identical, must differ already at one of the first $d+e$ initial values. Then we would not have to go so far to verify or falsify $\mathbf{a} = \mathbf{b}$. The following example gives a negative answer to this question.

Example 3.4.5 Let the sequences $\mathbf{a}$ and $\mathbf{b}$ be zeros of the operators

$$A = 187 + 1859n + 1672n^2 + (2800 - 3379n - 1672n^2)E + (-1350 + 1520n)E^2$$

and

$$B = 316 + 503n + 106n^2 + (56 - 609n - 106n^2)E + (-27 + 106n)E^2,$$

respectively. The sequences start with $a_0 = b_0 = 0$ and $a_1 = b_1 = 135$ and are therefore both uniquely defined zeros of these two operators. Moreover we have $a_2 = b_2$ and $a_3 = b_3$ and we conjecture that $\mathbf{a} = \mathbf{b}$.

If we compute a gcrd, say G, of A and B, we find that G has degree 0 in E. From Theorem 3.4.1 we can conclude that A and B have no nontrivial common zero, hence $\mathbf{a} \neq \mathbf{b}$.

Although both $\mathbf{a}$ and $\mathbf{b}$ are solutions of second order recurrences, it is not sufficient to check four initial conditions of the sequences to match. This is due to the fact that the leading coefficient of any lclm of A and B vanishes at $n = 0$. It follows that we also have to compare $a_4 = 3001$ and $b_4 = 3830$. Now it is evident that $\mathbf{a}$ and $\mathbf{b}$ are different sequences. □

Appendix A

Software and Availability

The Mathematica package `GeneratingFunctions` that is introduced in Chapter 2 is available via anonymous ftp at

`ftp.risc.uni-linz.ac.at:pub/combinatorics/mathematica/GeneratingFunctions`

or at the web address:

`http://info.risc.uni-linz.ac.at:70/labs-info/comblab/software/`
`Sequences/index.html`

The package can also be retrieved by sending a request via email to the author:

`Christian.Mallinger@risc.uni-linz.ac.at`

A lot of problems can be easily solved if the package `GeneratingFunctions` is used together with related software products. Thus we also point to other packages containing procedures that may interact with the procedures of our software. The list of packages, we refer to, is by far not complete. It should only be considered as a list of suggestions.

Although `GeneratingFunctions` is a Mathematica package, we also list some Maple packages that cover similar areas.

A.1 Maple Packages

`EKHAD`

Author: Doron Zeilberger (Temple University, Philadelphia, PA, USA)

Scope: Among others, the package `EKHAD` contains procedures that perform Zeilbergers method of creative telescoping.

Availability: The reader may consult the web page `http://www.math.temple.edu/~zeilberg`

`FormalPowerSeries` (FPS)

Author: Dominik Gruntz (ETH Zürich, Switzerland)

Scope: The function `FPS` tries to find a formal power series expansion for a function in terms of a formula for the coefficients.

Availability: `FPS` is a part of the Maple share library in the analysis directory.

`gfun`

Authors: Bruno Salvy, Paul Zimmermann and (since Jan. 1996) Eithne Murray. (INRIA Paris, France)

Scope: The package contains functions for manipulating holonomic sequences, generating functions, holonomic recurrence and differential equations. It can be regarded as the Maple forefather of the Mathematica package `GeneratingFunctions`, which is indeed based on the philosophy of `gfun`.

Availability: `gfun` is a part of the Maple share library in the analalysis directory. The latest version can be obtained via anonymous ftp from

`ftp.inria.fr:INRIA/Projects/algo/programs/gfun/`

A.2 Mathematica packages

`ENullSpace`

Author: Erhard Aichiniger (Universität Linz, Austria)

Scope: `ENullSpace` is a procedure that computes the nullspace of a matrix, whose entries are rational functions in several variables. This linear equation solver is considerably faster than the procedure `Nullspace`, which is built-in in the Mathematica kernel. (In fact many procedures in `GeneratingFunctions` use `ENullSpace` to solve systems of linear equations.)

Availability: A file with the source code is part of the package `fastZeil` (see below) and can be retrieved via ftp from the directory, where `fastZeil` is located.

`fastZeil`

Authors: Peter Paule and Markus Schorn (RISC, Universität Linz, Austria)

Scope: D. Zeilberger's method of creative telescoping for proving binomial coefficient identities is implemented. The package also contains an implementation of Gosper's algorithm for indefinite hypergeometric summation.

Availability: The program may be obtained via anonymous ftp from:

`ftp.risc.uni-linz.ac.at:pub/combinatorics/mathematica/PauleSchorn/`

or from the web address:

```
http://info.risc.uni-linz.ac.at:70/labs-info/comblab/software/
       Summation/index.html
```

HYPER

Author: Marko Petkovšek (University of Ljubljana, Slovenia)

Scope: This package contains procedures that find hypergeometric solutions of holonomic recurrence equations. Additionally, there are procedures to perform certain computations in the Ore polynomial ring $[\mathbf{K}_n][E]$. (See Chapter 3.)

Availability: The package is available at the web address:

`http://www.mat.uni-lj.si/ftp/pub/math/HYPER.M`

PowerSeries

Author: Wolfram Koepf (ZIB Berlin, Germany)

Scope: Given a holonomic function, the main procedure in the package tries to find a closed expression for the power series coefficients of this function. Another procedure that is a part of the package, tries to reverse this process, i.e, compute the generating function of a sequence.

Availability: The package can be retrieved from:

`ftp.zib-berlin.de:Pub/PowerSeries/`

RComp

Authors: István Nemes (RISC, Universität Linz, Austria) and Marko Petkovšek (University of Ljubljana, Slovenia)

Scope: `RComp` is a package for computing with the subfamily of holonomic sequences that satisfy recurrences with constant coefficients. Many operations with these sequences are implemented in a user friendly way. The design of the easy-to-use interfaces in this package inspired the author of this thesis to provide similar tools in his package `GeneratingFunctions`.

Availability: The program may be obtained from the ftp address:

`ftp.risc.uni-linz.ac.at:pub/combinatorics/mathematica/RComp/`

`RSolve`

Author: Marko Petkovšek (University of Ljubljana, Slovenia)

Scope: `RSolve` is an implementation of the method of generating functions for solving (systems of) linear recurrences. There are also tools that help in computing solutions of partial recurrence equations, finding power series expansions of analytic functions, and proving combinatorial identities.

Availability: `RSolve` is located in the `DiscreteMath` library of Mathematica .

Remark. After submitting the final version of this thesis, we have been informed that W. Koepf is preparing a package similar to `GeneratingFunctions`.

Bibliography

[AS64] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions.* Dover Publ., New York, 1964.

[BP92] F. Bergeron and S. Plouffe. Computing the generating function of a series given its first terms. *Journal of Experimental Mathematics*, 1:307–312, 1992.

[BP94] M. Bronstein and M. Petkovšek. On Ore rings, linear operators and factorisation. *Programming and Computational Software*, 20:14–26, 1994.

[BT71] W. S. Brown and J. F. Traub. On Euclid's algorithm and the theory of subresultants. *Journal of the ACM*, 18:505–514, 1971.

[Chy94] F. Chyzak. Holonomic systems and automatic proofs of identities. Rapport de recherche 2371, INRIA, Rocquencourt, 1994.

[Com64] L. Comtet. Calcul pratique des coefficients de Taylor d'une fonction algèbrique. *L'Enseignement Mathèmatique*, 10:267–270, 1964.

[CS] F. Chyzak and B. Salvy. Non-commutative elimination in Ore algebras proves multivariate identities. To appear in the *Journal of Symbolic Computation.*

[FS93] P. Flajolet and R. Sedgewick. The average case analysis of algorithms: Counting and generating functions. Rapport de recherche 2026, INRIA, Rocquencourt, 1993.

[Ges90] I. M. Gessel. Symmetric functions and P-recursiveness. *Journal of Combinatorial Theory Series A*, 53:257–285, 1990.

[GKP94] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison Wesley, second edition, 1994.

[Han75] E. R. Hansen. *A Table of Series and Products.* Prentice-Hall, Englewood Cliffs, NJ, 1975.

[Jun31] R. Jungen. Sur le series de Taylor n'ayant que des singularités algébrico-logarithmiques sur le cercle de convergence. *Comment. Math. Helv.*, 3:266–306, 1931.

[Koe92] W. Koepf. Power series in computer algebra. *Journal of Symbolic Computation*, 13:581–603, 1992.

[KS94] W. Koepf and D. Schmersau. Spaces of functions satisfying simple differential equations. Technical Report TR 94-2, Konrad Zuse-Zentrum Berlin (ZIB), 1994.

[Li96] Z. Li. A subresultant theory for linear differential, linear difference and Ore polynomials, with applications. Ph. D. thesis, RISC-Linz, J.-Kepler-University A-4040 Linz, 1996.

[Lip89] L. Lipschitz. D-finite power series. *Journal of Algebra*, 122:353–373, 1989.

[LN96] Z. Li and I. Nemes. A modular algorithm for computing greatest common right divisors of Ore polyynomials. To appear as Technical Report, RISC-Linz, J.-Kepler-University A-4040 Linz, 1996.

[NP95] I. Nemes and M. Petkovšek. RComp: A Mathematica package for computing with recursive sequences. *Journal of Symbolic Computation*, 20:745–753, 1995.

[Ore33] O. Ore. Theory of non-commutative polynomials. *Annals of Mathematics*, 34:480–508, 1933.

[Pet92] M. Petkovšek. Hypergeometric solutions of linear recurrences with polynomial coefficients. *Journal of Symbolic Computation*, 14:243–264, 1992.

[PS95] P. Paule and M. Schorn. A Mathematica version of Zeilberger's algorithm for proving binomial coefficient identities. *Journal of Symbolic Computation*, 20:673–698, 1995.

[PWZ96] M. Petkovšek, H. S. Wilf, and D. Zeilberger. *A=B*. A. K. Peters, Wellesley, MA, 1996.

[Rab93] S. Rabinowitz. *Index to Mathematical Problems 1980–1984*. Math Pro-Press, 1993.

[Ric68] D. Richardson. Some unsolvable problems involving elementary functions of a real variable. *Journal of Symbolic Logic*, 33:514–520, 1968.

[Sta80] R. P. Stanley. Differentiably finite power series. *European Journal of Combinatorics*, 1:175–188, 1980.

[SZ94] B. Salvy and P. Zimmermann. Gfun: A package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20:163–177, 1994.

[Wil90] H. S. Wilf. *generatingfunctionology.* Academic Press, Boston, 1990.

[Yen93] L. Yen. Contributions to the proof theory of hypergeometric identities. Ph. D. thesis, University of Pennsylvania, 1993.

[Zei82] D. Zeilberger. Sister Celine's technique and its generalizations. *Journal of Mathematical Analysis and Applications*, 85:114–145, 1982.

[Zei90] D. Zeilberger. A holonomic systems approach to special functions identities. *Journal of Computational and Applied Mathematics*, 32:321–368, 1990.