Order Number 9133361

Finding closed-form solutions of difference equations by symbolic methods

Petkovšek, Marko, Ph.D.

Carnegie-Mellon University, 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Finding Closed-Form Solutions of Difference Equations by Symbolic Methods

Marko Petkovšek

September 20, 1990
CMU-CS-90-171

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Carnegie Mellon

School of Computer Science

DOCTORAL THESIS
in the field of
Computer Science

Finding Closed-Form Solutions of Difference Equations by Symbolic Methods

MARKO PETKOVŠEK

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

ACCEPTED:

Dana S Scott	5/9/91
MAJOR PROFESSOR	DATE
[illegible]	13 May '91
DEAN	DATE

APPROVED:

[illegible]	6/10/91
PROVOST	DATE

Abstract

This thesis investigates symbolic computation in the domain of difference equations (or recurrence relations). The goal is to obtain explicit solutions of a given equation automatically and, when possible, in closed form.

The main contributions of the thesis are:

- A comprehensive implementation of the method of generating functions as a *Mathematica* package called RSolve.m. The package can automatically compute ordinary and exponential generating functions for, and find closed-form solutions of, linear difference equations with constant coefficients, certain linear equations with nonconstant coefficients, equations which contain convolutions, and systems of such equations. Used as a collection of tools, the package can be employed to compute closed-form solutions of certain partial difference equations, to obtain recurrences for power-series coefficients of analytic functions, and to prove combinatorial identities.

- An existence and uniqueness theorem for partial difference equations in the nonnegative orthant.

- A proof that the generating function corresponding to the solution of a linear partial difference equation with constant coefficients with at most exponentially growing initial conditions is analytic.

- An algorithm for finding all polynomial solutions of a homogeneous linear difference equation with polynomial coefficients.

- An algorithm for finding all hypergeometric solutions of a homogeneous linear difference equation with polynomial coefficients. A sequence (h_n) is *hypergeometric* if the quotient h_{n+1}/h_n is a rational function of n. Combined with an algorithm of Zeilberger, which, given a definite sum $a_n = \sum_{k=-\infty}^{\infty} F(n,k)$ where $F(n,k)$ is hypergeometric in both n and k, produces a linear recurrence for a_n with polynomial coefficients, it solves the long standing problem of deciding whether a definite sum such as a_n above is hypergeometric or not. For example, the algorithm can be used to prove that the number of involutions of an n-element set is not hypergeometric.

- A proof of the theorem that the Galois group of a linear difference operator with polynomial coefficients over the difference ring of germs at infinity of sequences over a field is an algebraic matrix group.

Acknowledgements

I first met my advisor Dana Scott at a seminar in Dubrovnik in 1983. At that time I did not know what impact this was going to have on my life. I came to study with him in Pittsburgh and a new world opened in front of me.

I thank him for his support, direction, and encouragement. His enthusiasm for the wonderful potential of symbolic computation has been contagious, and has finally led to this thesis. I also thank him and his wife Irene Schreier Scott for their great care for my family.

I had the privilege to be part of the computer science community at Carnegie Mellon University. The excellent facilities and the supportive and stimulating atmosphere made it an ideal working environment. I learned much from the faculty, visitors, and other students. I had helpful discussions with Ed Clarke and Doug Tygar, and a problem of Peter Luksch who was visiting started me to think about partial difference equations. My fellow students Roberto Minio, Pino Rosolini, Jean-Philippe Vidal, Todd Wilson, and Peter Jansen not only provided encouragement and support but also became good friends. I thank especially Todd Wilson who was always there to discuss a problem. He also read parts of the draft and made valuable suggestions.

It was Herb Wilf, from University of Pennsylvania, whose sincere enthusiasm for my results brought me out of despair not just once. Doron Zeilberger of Temple University was always ready to answer my questions, and so was Philippe Flajolet of INRIA-Rocquencourt.

Thanks to Stephen Wolfram, I spent a summer at WRI, Inc. in Champaign, Ill. which was perhaps decisive for my final choice of thesis topic. Ever since, Igor Rivin has been most supportive of my work, posing me interesting problems and offering advice.

Tomaž Pisanski agreed to come to Pittsburgh and serve on my thesis committee. Sandi Klavžar has given me constant encouragement and a good example. I also want to thank my mathematics professors in Ljubljana, especially Zvonimir Bohte and Egon Zakrajšek.

And finally, my greatest thanks go to my wife, Eva, whose infinite love and generosity made it all possible.

Contents

Chapter 1

Introduction and Summary

Despite the title this is not a monograph on linear difference equations; it is a discourse on linear recurrence relations. The distinction is simple. In a difference equation the argument varies continuously; in a recurrence relation the argument takes on only (equally spaced) discrete values. Usage has blurred the distinction in terminology. Many people use the terms interchangeably. We shall too.

— KENNETH S. MILLER, *Linear Difference Equations (1968)*

In the past decade, the field of symbolic computation has experienced intensive growth. On the software and systems side, the older and more mature systems such as *Macsyma* [PW85], *Reduce* [Mac89], and *Scratchpad* [Jen84] have been joined by new systems with enhanced algorithmic, graphic, and/or programming capabilities such as *Maple* [C^+85] and *Mathematica*$^{\text{TM}}$ [Wol88]. Symbolic computation systems have been ported to many different platforms, ranging from personal computers through workstations to supercomputers. On the algorithmic side, many important advances have been made. To mention just three: Buchberger's [Buc65] algorithm for construction of Gröbner bases in polynomial ideals (which went largely unnoticed for several years) became a powerful tool for dealing with systems of multivariate polynomial equations, and found numerous applications [Buc85]. Thanks to the work of Risch [Ris69], Trager [Tra84], Bronstein [Bro90], and others, there now exists a complete algorithm for symbolic indefinite integration of elementary functions which have elementary integrals. Singer [Sin81] and Kovacic [Kov86], building on previous work, have developed algorithms for finding Liouvillian solutions of linear differential equations with rational coefficients. Built into powerful symbolic computation systems, these algorithms can significantly increase productivity of any scientific or engineering work which uses mathematical tools. However, what has been achieved is likely to prove only a glimpse at the vast potential which symbolic computational methods promise to have. A great deal of work remains to be done, both in finding better algorithms in the established areas, and in automating other mathematical disciplines which have hitherto not received sufficient attention.

This thesis investigates symbolic computation in the domain of difference equations (which we sometimes also call recurrence relations or recursions, in the spirit of the above quotation). The goal is to obtain solutions of a given equation automatically and, if possible, in closed form.

The thesis consists of three parts. In Chapter 2 we consider issues related to implementation of the method of generating functions. In Chapter 3 we investigate solutions and generating functions defined by partial difference equations. In Chapters 4 and 5 we examine ordinary linear difference equations with polynomial coefficients and algorithms for finding solutions of such equations belonging to certain classes of functions.

The main contributions of the thesis are:

- A comprehensive implementation of the method of generating functions as a *Mathematica*$^{\text{TM}}$ package called RSolve.m. The package can automatically compute ordinary and exponential generating functions for, and find closed-form solutions of, linear difference equations with constant coefficients, certain linear equations with nonconstant coefficients, equations which contain convolutions, and systems of such equations. Used as a collection of tools, the package can be employed to compute closed-form solutions of partial difference equations, to obtain recurrences for power-series coefficients of analytic functions, and to prove combinatorial identities.

- An existence and uniqueness theorem for partial difference equations in the nonnegative orthant.

- A proof that the generating function corresponding to the solution of a linear partial difference equation with constant coefficients with at most exponentially growing initial conditions is analytic.

- An algorithm for finding all polynomial solutions of a homogeneous linear difference equation with polynomial coefficients.

- An algorithm for finding all hypergeometric solutions of a homogeneous linear difference equation with polynomial coefficients. A sequence h_n is *hypergeometric* if the quotient h_{n+1}/h_n is a rational function of n. Let $F(n,k)$ be hypergeometric in both n and k, and let $a_n = \sum_k F(n,k)$ where summation ranges over all integers. Zeilberger [Zeib] discovered an algorithm which given $F(n,k)$ produces a linear recurrence for a_n with polynomial coefficients. In combination with Zeilberger's algorithm, the present algorithm solves the long standing problem of deciding whether a definite sum such as a_n is hypergeometric or not. For example, the algorithm can be used to prove that the number of involutions of an n-element set is not hypergeometric.

- A proof of the theorem that the Galois group of a linear difference operator with polynomial coefficients over the difference ring of germs at infinity of sequences over a field is an algebraic matrix group. This constitutes a major step towards an algorithm for finding Liouvillian solutions of difference equations with polynomial coefficients.

1.1 The Method of Generating Functions

Generating functions are treated in more or less any book on combinatorics, difference equations, discrete or concrete mathematics, such as [Jor60], [Bra66], [Mil68], [Com74], [Knu68], [GKP89], and [Wil90]. In control theory a variant of the method is known as the z-transform (see [Jur64], [Kac85]). For a survey and bibliography of the method of generating functions, see [Sta78].

If $\mathcal{F} = (f_n(z))_{n=0}^{\infty}$ is a fixed sequence of complex-valued functions of a complex argument, then the generating function of a sequence $a = (a_n)_{n=0}^{\infty}$ of complex numbers with respect to $\mathcal{F}$ is the formal series

$$G[a](z) = \sum_{n=0}^{\infty} a_n f_n(z).$$

For certain choices of $\mathcal{F}$, the corresponding generating functions have special names:

$f_n(z) = z^n$	(ordinary) generating function
$f_n(z) = z^n/n!$	exponential generating function
$f_n(z) = 1/n^z$	Dirichlet generating function

This definition can be generalized to sequences with several indices, in which case generating functions are formal series in several variables.

The process of solving difference equations by the method of generating functions consists of three main steps:

1. Transform difference equations for the unknown sequences into equations for their generating functions.
2. Solve the resulting functional equations.
3. Expand the obtained generating functions and read off the n-th coefficient.

The transformation of step 1 preserves linearity of the equations. Linear difference equations with constant coefficients give rise to linear algebraic equations for the corresponding ordinary generating functions, which are therefore rational functions of z. Linear difference equations with polynomial (or equivalently, rational) coefficients give rise to linear differential equations for the corresponding ordinary generating functions, with coefficients rational in z, and of order equal to the maximum degree of the coefficients of the original equation. This is shown in Section 2.2 (Transformation Rules). However, only those solutions of the differential equations which are analytic at the origin correspond to solutions of the difference equations, because the fundamental assumption of the method is that the generating function has a power series expansion around $z = 0$. Hence the method of generating functions will not give solutions which grow too fast, unless we can solve the associated differential equation as an equation in formal power series. This is the case, for example, with equations of the hypergeometric type which are linear differential equation with terms of the form $x^p y^{(q)}(x)$, and such that the set of differences $p - q$ for all terms consists of (at most) two consecutive integers. Other kinds of difference equations may give rise to other kinds of functional equations for the generating functions. It is appropriate to mention at this point that power series which satisfy a homogeneous linear differential equation with polynomial coefficients are called *D-finite*, and sequences which satisfy a homogeneous linear difference equation with polynomial coefficients are called *P-recursive* by Stanley [Sta80]. As already mentioned in part, generating functions of P-recursive sequences are D-finite, and vice versa, which is not hard to see. Zeilberger [Zeia] uses the term *P-finite* in both contexts.

The method of generating functions can be used without changes to solve systems of simultaneous difference equations. In this case the difference equations are transformed into a system of simultaneous (algebraic or differential) equations for the generating functions.

The real strength of the method when compared to others, however, lies in its ability to deal with nonlinear equations whose nonlinearity has the form of a convolution, and with certain types of partial difference equations. Since convolutions of sequences correspond generally to products of their generating functions, such equations give rise to nonlinear algebraic equations, and the corresponding generating functions are algebraic in z.

Some previous implementations of the method are described in [CK77], [Ivi78] (see also [Cel84]), and [FSZ89].

1.2 The Package RSolve.m

RSolve.m is implemented in *Mathematica*$^{\text{TM}}$ V1.2 and contains about 1200 lines of *Mathematica*$^{\text{TM}}$ code. From among *Mathematica*$^{\text{TM}}$'s built-in functions on which RSolve.m relies, the most important are `Solve[ ]` and `DSolve[ ]` which solve algebraic and differential equations, respectively. The functionality of the package in many cases depends directly on the functionality of these two functions.

The principal functions provided by RSolve.m are:

- `RSolve[ ]`
- `GeneratingFunction[ ]`
- `ExponentialGeneratingFunction[ ]`
- `PowerSum[ ]`
- `ExponentialPowerSum[ ]`
- `SeriesTerm[ ]`

`RSolve[ ]` is the top-level function which attempts to solve given difference equations automatically. First it invokes the method of ordinary generating functions, and second, in case of failure, the method of exponential generating functions. The user can specify the order and selection of methods to be tried, and add new methods. The transformation of difference equations into functional equations for the corresponding generating functions is performed by `PowerSum[ ]` and `ExponentialPowerSum[ ]`, respectively. Then `GeneratingFunction[ ]` or `ExponentialGeneratingFunction[ ]` attempt to solve the obtained functional equations by using the built-in function `Solve[ ]` in case they are algebraic, and `HSolve[ ]` (see below) in case they are differential. If `HSolve[ ]` fails the built-in function `DSolve[ ]` is tried. After the generating function has been successfully computed, `SeriesTerm[ ]` attempts to expand it into a power series and read off the n-th term.

The package allows great freedom in the format of the input, as well as in the choice of initial and/or boundary conditions given. The user can control the action of the functions by means of several optional parameters. The user can also specify typing information about individual variables by giving Boolean combination of inequalities, equations, and inequations which they satisfy.

The scope of difference equations which `RSolve[ ]` can handle includes linear equations with constant coefficients, linear equations with nonconstant coefficients which either are first-order homogeneous and have rational coefficients, or are such that `HSolve[ ]` can solve the associated differential equation, or `DSolve[ ]` can solve the associated differential equation and the solution grows no faster than $n!\,a^n$ for some constant a. It can also solve nonlinear equations where the nonlinearity has the form of a convolution. Finally, it can solve systems of simultaneous equations of these types.

Used not automatically but rather as a collection of tools, the package RSolve.m can be employed to solve partial difference equations, to obtain recurrences for power-series coefficients of analytic functions, and to prove combinatorial identities.

Besides the functions itemized above the package contains many useful auxiliary functions. Of these, the most important are `HSolve[ ]` and `ISolve[ ]`. `HSolve[ ]` computes a formal power-series solution of a linear differential equation whose terms are of the form $x^p y^{(q)}(x)$, and the set of differences $p - q$ for all terms consists of (at most) two consecutive integers, provided that the solution can be expressed in terms of polynomials and generalized hypergeometric series.

`ISolve[ ]` solves Boolean combinations of inequalities, equations, and inequations involving rational functions of a single variable. The answer is given in the form of a disjunction of pairwise disjoint intervals.

The implementation of RSolve.m takes advantage of *Mathematica*™'s rich collection of programming constructs which allow the choice among functional, procedural, and rule-based programming. On the micro level, functional programming style is used throughout the package. On the macro level, `RSolve[ ]`, `GeneratingFunction[ ]`, and `ExponentialGeneratingFunction[ ]` are written in the procedural style, whereas `PowerSum[ ]`, `ExponentialPowerSum[ ]`, and `SeriesTerm[ ]` are large collections of rewrite rules. This allows for new rules to be added with little or no change to the existing code.

1.3 Partial Difference Equations

Let A be a nonempty set. We consider d-dimensional partial difference equations of the form

$$a_{\mathbf{p}} = F(a_{\mathbf{p}+\mathbf{z}_1}, a_{\mathbf{p}+\mathbf{z}_2}, \dots, a_{\mathbf{p}+\mathbf{z}_k}), \quad \text{for } \mathbf{p} \geq \mathbf{s}, \tag{1.1}$$

where

- $\mathbf{s} \in \mathbb{N}^d$ is a given point,
- $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k \in \mathbb{Z}^d$ are given points, such that $\mathbf{s} + \mathbf{z}_i \in \mathbb{N}^d$ for $i = 1, 2, \dots, k$,
- $F : A^k \to A$ is a given function, and
- $a : \mathbb{N}^d \to A$ is the unknown sequence.

Let $I := \{\mathbf{p} \in \mathbb{N}^d;\ \mathbf{p} \not\geq \mathbf{s}\}$ be the *initial set* for (1.1). We assume that the initial conditions are of the form

$$a_{\mathbf{p}} = f(\mathbf{p}), \quad \text{for } \mathbf{p} \in I, \tag{1.2}$$

where $f : I \to A$ is a given function.

Let $Z := \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k\}$. Suarez [Sua89] has shown that if $d = 2$ and the points of Z all precede the origin in lexicographic ordering, then (1.1), (1.2) has a unique solution. Here we characterize the sets $Z \subseteq \mathbb{Z}^d$ for which this existence and uniqueness result holds.

Definition 1 For $Z \subseteq \mathbb{Z}^d$ and $\mathbf{p}, \mathbf{q} \in \mathbb{N}^d$, let

$$\mathbf{p} \prec_Z \mathbf{q} \qquad \text{if} \qquad \mathbf{p} - \mathbf{q} \in Z \text{ and } \mathbf{q} + Z \subseteq \mathbb{N}^d . \tag{1.3}$$

We call the transitive closure $\overset{*}{\prec}_Z$ of $\prec_Z$ in $\mathbb{N}^d$ the *dependency relation* corresponding to Z. □

For $S \subseteq \mathbb{R}^d$, let $\operatorname{conv} S$ denote the convex hull of S.

Definition 2 Let $S \subseteq \mathbb{R}^d$. The *integer cone* of S is the set

$$\operatorname{icon} S = \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} = \sum_{i=0}^{k} \lambda_i \mathbf{s}_i,\ \lambda_i \in \mathbb{N},\ \mathbf{s}_i \in S\} . \quad \square$$

We recall that a transitive relation is *well-founded* if it is asymmetric and has no infinite descending chain.

Theorem 1 *Let $Z \subseteq \mathbb{Z}^d$ be a finite set, and $\overset{*}{\prec}_Z$ the corresponding dependency relation. The following assertions are equivalent:*

(i) $\overset{*}{\prec}_Z$ *is well-founded in* $\mathbb{N}^d$,

(ii) $\operatorname{icon} Z$ *is disjoint from* $\{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \geq \mathbf{0}\}$,

(iii) $\operatorname{conv} Z$ *is disjoint from* $\{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \geq \mathbf{0}\}$,

(iv) *there exists an* $\mathbf{a} \in \mathbb{R}^d$, $\mathbf{a} > \mathbf{0}$, *such that* $\mathbf{a} \cdot \mathbf{z} < 0$ *for all* $\mathbf{z} \in Z$,

(v) *there exists an* $\mathbf{a} \in \mathbb{N}^d$, $\mathbf{a} > \mathbf{0}$, *such that* $\mathbf{a} \cdot \mathbf{z} < 0$ *for all* $\mathbf{z} \in Z$,

(vi) $\overset{*}{\prec}_Z$ *can be embedded into a linear ordering of* $\mathbb{N}^d$ *of order type* ω.

Corollary 1 *Let $Z \subseteq \mathbb{R}^d$ be a finite set which satisfies any of the conditions of Theorem 1. Then* (1.1), (1.2) *has a unique solution.*

An ordinary difference equation with constant coefficients defines a generating function which is rational. We give an example which shows that a multivariate generating function defined by a partial difference equation with constant coefficients need not be rational, even though the lower-dimensional generating functions corresponding to the initial conditions are. However, we prove the following:

Theorem 2 *Let $Z \subseteq \mathbb{R}^d$ be a finite set which satisfies any of the conditions of Theorem 1. Then the generating function corresponding to the solution of* (1.1), (1.2) *is analytic (provided that the rate of growth of the initial conditions is at most exponential).*

It remains open whether or not such a generating function is always algebraic.

1.4 Difference Equations with Polynomial Coefficients

Let $p_0(x), p_1(x), \ldots, p_d(x)$ be given polynomials with coefficients in a field F, and such that $p_0, p_d \not\equiv 0$. Then

$$\sum_{i=0}^{d} p_i(n)\, a_{n+i} = 0\,, \qquad \text{for } n \geq n_0\,, \tag{1.4}$$

is a ***homogeneous linear difference equation with polynomial coefficients*** (an HLP, for short) over F for the unknown sequence a_n. The ***order*** of (1.4) is d, and the ***degree*** of (1.4) is $m = \max_{0 \leq i \leq d} \deg p_i(n)$. A sequence which satisfies an HLP is variably called *P-recursive* [Sta80], [Lip88], [Lip89], *P-finite* [Zeia], and *holonomic* [Zeia].

Unlike the case of equations with constant coefficients, there is no general method known for obtaining explicit solutions to arbitrary HLP's. Therefore it is natural to ask for a given equation whether it has a solution in a certain fixed class of functions. We start by deriving algorithm POLY which given an HLP finds a basis for the space of all its polynomial solutions. As it turns out, there exists a polynomial $P(x)$ of degree not exceeding the order of the HLP such that the degree of any non-zero polynomial solution of the HLP is a root of $P(x)$. The coefficients of $P(x)$ can be computed from those of the coefficients of the HLP. So one needs only to find the largest nonnegative integer root N of $P(x)$, substitute a generic polynomial $G(n)$ of degree N for a_n in the HLP, and solve a system of linear algebraic equations for the coefficients of $G(n)$.

Next we consider hypergeometric solutions. A non-zero sequence (a_n) is *hypergeometric* over a field F if it satisfies a first-order HLP over F, or equivalently, if the quotient a_{n+1}/a_n is a rational function of n over F. Examples of hypergeometric functions include non-zero constants, polynomials, rational functions, and functions like $n!$, $2^n/(n-1)!^3$, and $\alpha^{\overline{n}} = \alpha(\alpha+1)\cdots(\alpha+n-1)$ where α is a constant. If F is algebraically closed then the general form of a hypergeometric function is

$$C\frac{\alpha_0^{\overline{n}}\alpha_1^{\overline{n}}\cdots\alpha_p^{\overline{n}}}{\beta_0^{\overline{n}}\beta_1^{\overline{n}}\cdots\beta_q^{\overline{n}}}Z^n$$

where C, the α_i, the β_i, and Z are constants from F.

Hypergeometric sequences are closed under multiplication and taking reciprocals, but not under addition. For example, $2^n + 1$ is not hypergeometric. However, finite sums of hypergeometric sequences are closed under addition, and form an algebra both over F and over $F(x)$, the field of rational functions over F.

Using a decomposition of rational functions (which plays an important role in Gosper's algorithm [Gos78] as well), we develop algorithm HYPER which returns a basis for the F-space of solutions which are finite sums of hypergeometric sequences. The algorithm works by constructing a finite set of auxiliary HLP's (of the same order as the original one) such that the original HLP has a hypergeometric solution if and only if an auxiliary HLP has a non-zero polynomial solution. Algorithm POLY is then used on each auxiliary HLP to determine if it has any non-zero polynomial solutions. A hypergeometric solution of the original HLP is easily obtained from any non-zero polynomial (or, for that matter, hypergeometric) solution of an auxiliary HLP.

The auxiliary HLP's are generated in a loop which runs through all monic factors of the leading and constant coefficients of the given HLP, as well as through the roots of a polynomial whose degree does not exceed the order of the given HLP. A straightforward implementation of the algorithm thus requires computation with algebraic numbers.

Furthermore, algorithm HYPER can be used to determine hypergeometric solutions of non-homogeneous difference equations with polynomial coefficients, by applying it to an appropriate HLP of one-higher degree. Since given an HLP for a_n it is not hard to construct an HLP satisfied by Δa_n, algorithm HYPER can also be used to determine solutions whose k-th differences are hypergeometric, for any fixed k.

1.5 Definite Hypergeometric Summation

The problem of indefinite hypergeometric summation was solved by Gosper [Gos77], [Gos78] who designed an algorithm for deciding whether the indefinite sum

$$a_n = \sum_{k=0}^{n} f_k$$

where f_k is a given hypergeometric sequence, is itself hypergeometric (apart from an additive constant).

The analogous question about definite sums of the form

$$a_n = \sum_k F(n,k) \tag{1.5}$$

where summation ranges over all integers and $F(n,k)$ is hypergeometric in both arguments (i.e., $F(x+1,y)/F(x,y)$ and $F(x,y+1)/F(x,y)$ are rational functions of x and y), has long been open (cf. [Wil91]). In an important development, Zeilberger [Zeia] proved that every a_n of the form (1.5) satisfies an HLP, and in [Zeib] gave an algorithm which constructs such an equation. In general, this equation is not of minimum degree. However, using first Zeilberger's algorithm to find the recurrence, then algorithm HYPER to find a basis for the space generated by its hypergeometric solutions, and finally solving a system of linear algebraic equations to expand (1.5) in this basis, represents an algorithm to decide whether (1.5) is a hypergeometric sequence.

For example, this algorithm can be used to prove that the number of involutions of an n-element set

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

(cf. [Com74]) is not hypergeometric.

1.6 Galois theory of difference equations

Definition 3 A *difference ring* is a commutative ring k with unit, together with an automorphism τ, which is called the *transform* of k. A *difference field* is a difference ring which is a field.

The set $C(k) = \{x \in k;\ \tau x = x\}$ is a subring of k called the *ring of constants* of k. If k is a field then so is $C(k)$. □

In applications k will most often be the field of rational functions over a field of characteristic zero, and τ the shift operator.

Galois theory of difference equations in difference fields was developed in a series of papers by Franke: [Fra63], [Fra66], [Fra67], [Fra69], [Fra71], [Fra73], [Fra74]. It turns out that Galois theory of difference equations is more complicated than the corresponding theory for differential equations. One of the reasons is that if $P(y, y', \ldots, y^{(n)})$ is a differential polynomial, then its derivative is linear in $y^{(n+1)}$, while no such linearization occurs with difference polynomials.

A difference field K is a *Liouvillian extension* of k if there is a finite chain of difference fields $k = F_0 \subseteq F_1 \subseteq \ldots \subseteq F_k = K$ such that F_i is either finite algebraic over F_{i-1}, or is obtained from F_{i-1} by adjunction of some y which satisfies an equation of the form $y_{n+1} = a_n y_n$ or $y_{n+1} - y_n = b_n$ where a and b are elements of F_{i-1}. In order to obtain a more satisfactory theory, Franke [Fra66] expanded the definition of closed-form solutions to *q–Liouvillian extensions* which allow for adjunctions of solutions of $y_{n+q} = a_n y_n$ and $y_{n+q} - y_n = b_n$. Here q is a positive integer. Still, this development did not lead to a complete resolution of the problem of existence of Liouvillian solutions as did the analogous line of research in the case of differential equations (see [Kol73], [Kap57], [Sin81], [Kov86]). The main obstacle was the existence of difference equations with coefficients in a difference field k and such that any extension field K of k in which the equation has a non-zero solution (a *solution field* for the equation) contains new constants (i.e., constant elements which are not in k). An example is furnished by the simple equation $a_{n+1} + a_n = 0$ with k the field of rational functions over the complex numbers. If a_n satisfies this equation then a_n^2 is a constant, hence (assuming there are no new constants) a complex number. But then its square root, a_n, is a complex number and a constant as well. It now follows from the equation that $a_n = 0$. We conclude that any solution field for this equation contains new constants (such as a_n^2).

Instead of fields of functions, we work in rings of sequences. In order to get a difference ring, the shift operator must be made into an automorphism by identifying sequences which agree from some point on. This also works well for rational sequences since we don't have to worry about their poles. The units in this quotient ring are the sequences which are non-zero from some point on. The advantage of this setting is that we have a universal structure in which to work - the ring of all sequences over the field of coefficients, and we never get any new constants. The setback is the appearance of nonunits (which are also zero divisors).

We prove that in this setting the Galois group of a linear difference operator with rational coefficients is an algebraic matrix group, which is one of the major stepping stones in developing Galois theory. The others are establishing Galois correspondence, proving normality of extensions, and determining periodicity properties of nonunit solutions. This constitutes a major research program in itself.

Chapter 2

An Implementation of the Method of Generating Functions

Given a sequence $\langle g_n \rangle$ that satisfies a given recurrence, we seek a closed form for g_n in terms of n. A solution to this problem via generating functions proceeds in four steps that are almost mechanical enough to be programmed on a computer.

— GRAHAM, KNUTH, PATASHNIK, *Concrete Mathematics (1989)*

2.1 Introduction

This chapter describes the implementation and functionality of RSolve.m, a *Mathematica*™ package for solving difference equations. The package is based on the method of generating functions.

For a survey and bibliography of the method of generating functions, see Stanley [Sta78]. If $\mathcal{F} = (f_n(z))_{n=0}^{\infty}$ is a fixed sequence of complex-valued functions of a complex argument, then the generating function of a sequence $a = (a_n)_{n=0}^{\infty}$ of complex numbers with respect to $\mathcal{F}$ is the formal series

$$G(a, z) = \sum_{n=0}^{\infty} a_n f_n(z) \, .$$

For certain choices of $\mathcal{F}$, the corresponding generating functions have special names:

$f_n(z) = z^n$	(ordinary) generating function
$f_n(z) = z^n/n!$	exponential generating function
$f_n(z) = 1/n^z$	Dirichlet generating function

This definition can be generalized to sequences with several indices, with generating functions of such sequences being formal series in several variables.

The process of solving difference equations by the method of generating functions consists of three main steps:

1. Transform difference equations for the unknown sequences into equations for their generating functions.

2. Solve the resulting functional equations.

3. Expand the obtained generating functions into appropriate series.

The transformation of step 1 preserves linearity of the equations. Linear difference equations with constant coefficients give rise to linear algebraic equations for the corresponding generating functions, which are therefore rational functions of z. Linear difference equations with polynomial (or equivalently, rational) coefficients give rise to linear differential equations for the corresponding generating functions, with coefficients rational in z, and of order equal to the maximum degree of the coefficients of the original equation. However, only those solutions of the differential equations which are analytic at the origin correspond to solutions of the difference equations, because the fundamental assumption of the method is that the generating function has a power series expansion around $z = 0$. Solutions of the differential equation which have an essential singularity at the origin, however, can have the generating function as an asymptotic expansion (cf. Exercise 1.2.9. - 11 in [Knu68]), but these are much harder to compute than the power series expansions and therefore not as useful. Hence the method of generating functions will not give solutions which grow too fast, unless we can solve the associated differential equation as an equation in formal power series. This is the case, for example, with equations of the hypergeometric type. Other kinds of difference equations may give rise to other kinds of functional equations for the generating functions.

The method of generating functions can be used without changes to solve systems of simultaneous difference equations. In this case the difference equations are transformed into a system of simultaneous (algebraic or differential) equations for the generating functions.

The real strength of the method when compared to others, however, lies in its ability to deal with nonlinear equations whose nonlinearity has the form of a convolution, and with certain types of partial difference equations. Since convolutions of sequences correspond generally to products of their generating functions, such equations give rise to nonlinear algebraic equations, and the corresponding generating functions are algebraic in z.

Linear partial difference equations with constant coefficients translate into algebraic equations for the corresponding ordinary generating functions. Linear partial difference equations with polynomial (or equivalently, rational) coefficients translate into linear partial differential equations for the corresponding ordinary generating functions. Again, the method can be applied equally well to systems of simultaneous partial difference equations.

2.2 Transformation Rules

This section contains derivations of a representative sample of rules used in RSolve.m.

Let $(a_n)_{n=0}^{\infty}$ be a sequence of complex numbers. Its *ordinary* and *exponential generating functions* are defined as the formal power series

$$GF(a_n, z, n, n_0) = \sum_{n=n_0}^{\infty} a_n z^n$$

and

$$EGF(a_n, z, n, n_0) = \sum_{n=n_0}^{\infty} a_n \frac{z^n}{n!},$$

respectively. We list some transformation rules which relate operations with sequences to operations with their generating functions.

1. Linearity. If $\lambda, \mu \in \mathbb{C}$ then

$$\begin{aligned} GF(\lambda a_n + \mu b_n, z, n, n_0) &= \lambda GF(a_n, z, n, n_0) + \mu GF(b_n, z, n, n_0), \\ EGF(\lambda a_n + \mu b_n, z, n, n_0) &= \lambda EGF(a_n, z, n, n_0) + \mu EGF(b_n, z, n, n_0). \end{aligned}$$

2. Shifts and polynomial factors. Let $k \in \mathbb{N}$ and $x \in \mathbb{C}$. Then the rising and falling powers are defined as

$$n^{\overline{k}} = \prod_{j=0}^{k-1}(n+j)$$

and

$$n^{\underline{k}} = \prod_{j=0}^{k-1}(n-j),$$

respectively. If $k, d \in \mathbb{N}$ then

$$\begin{aligned} GF(n^{\underline{k}} a_{n+d}, z, n, 0) &= \sum_{n=0}^{\infty} n^{\underline{k}} a_{n+d} z^n \\ &= z^k \sum_{n=d}^{\infty} (n-d)^{\underline{k}} a_n z^{n-d-k} \\ &= z^k (\sum_{n=d}^{\infty} a_n z^{n-d})^{(k)} \\ &= z^k \left(GF(a_n, z, n, d)/z^d\right)^{(k)}. \end{aligned}$$

If $k, d+k \in \mathbb{N}$ then

$$\begin{aligned} EGF(n^{\underline{k}} a_{n+d}, z, n, 0) &= \sum_{n=0}^{\infty} \frac{n^{\underline{k}}}{n!} a_{n+d} z^n = \sum_{n=k}^{\infty} a_{n+d} \frac{z^n}{(n-k)!} \\ &= \sum_{n=-d}^{\infty} (n+d)^{\underline{d+k}} a_{n+d} \frac{z^n}{(n+d)!} = \sum_{n=0}^{\infty} n^{\underline{d+k}} a_n \frac{z^{n-d}}{n!} \\ &= z^k \left(\sum_{n=0}^{\infty} a_n \frac{z^n}{n!} \right)^{(d+k)} = z^k \left(EGF(a_n, z, n, 0)\right)^{(d+k)}. \end{aligned}$$

To convert ordinary powers to falling powers we use the well-known formula

$$n^k = \sum_{i=0}^{k} S(k,i) n^{\underline{i}}$$

where $S(k,i)$ are the Stirling numbers of the second kind (see, for example, [GKP89]).

3. Convolutions. If $r, s \in \mathbb{Z}$ then

$$\begin{aligned}
GF\left(\sum_{k=r}^{n+s} a(k)b(n,k), z, n, 0\right) &= \sum_{n=0}^{\infty}\sum_{k=r}^{n+s} a(k)b(n,k)z^n \\
&= \sum_{k=r}^{s-1} a(k)z^k \sum_{n=0}^{\infty} b(n,k)z^{n-k} + \sum_{k=\max\{r,s\}}^{\infty} a(k)z^k \sum_{n=k-s}^{\infty} b(n,k)z^{n-k} \\
&= \sum_{k=r}^{s-1} a(k)z^k \sum_{n=-k}^{\infty} b(n+k,k)z^n + \sum_{k=\max\{r,s\}}^{\infty} a(k)z^k \sum_{n=-s}^{\infty} b(n+k,k)z^n \\
&= \sum_{k=r}^{s-1} a(k)z^k GF(b(n+k,k), z, n, -k) + GF(a(k)GF(b(n+k,k), z, n, -s), z, k, \max\{r,s\}),
\end{aligned}$$

and

$$\begin{aligned}
EGF\left(\sum_{k=r}^{n+s} \binom{n}{k} a(k)b(n,k), z, n, 0\right) &= \sum_{n=0}^{\infty}\sum_{k=r}^{n+s} a(k)b(n,k)\frac{z^n}{k!\,(n-k)!} \\
&= \sum_{k=r}^{s-1} a(k)\frac{z^k}{k!} \sum_{n=0}^{\infty} b(n,k)\frac{z^{n-k}}{(n-k)!} + \sum_{k=\max\{r,s\}}^{\infty} a(k)\frac{z^k}{k!} \sum_{n=k-s}^{\infty} b(n,k)\frac{z^{n-k}}{(n-k)!} \\
&= \sum_{k=r}^{s-1} a(k)\frac{z^k}{k!} \sum_{n=-k}^{\infty} b(n+k,k)\frac{z^n}{n!} + \sum_{k=\max\{r,s\}}^{\infty} a(k)\frac{z^k}{k!} \sum_{n=-s}^{\infty} b(n+k,k)\frac{z^n}{n!} \\
&= \sum_{k=r}^{s-1} a(k)\frac{z^k}{k!} EGF(b(n+k,k), z, n, -k) \\
&\quad + EGF(a(k)EGF(b(n+k,k), z, n, -s), z, k, \max\{r,s\}).
\end{aligned}$$

2.3 The Package RSolve.m

The package **RSolve.m**, written in *Mathematica*$^{\text{TM}}$ [Wol88], implements the method of ordinary and exponential generating functions. The author has presented this package at the 1990 *Mathematica*$^{\text{TM}}$ conference in Redwood City, CA [Pet90]. The package is capable of solving a wide range of ordinary difference equations and systems of such equations, as well as some linear partial difference equations.

The rest of this section consists of a *Mathematica*$^{\text{TM}}$ notebook containing a description of the package and the functions it provides. The corresponding *Mathematica*$^{\text{TM}}$ code is in Appendix A. It runs under version 1.2 or higher.

The principal functions provided by **RSolve.m** are:

RSolve	alias	**RS**
GeneratingFunction	alias	**GF**
ExpGeneratingFunction	alias	**EGF**
PowerSum	alias	**PS**
ExpPowerSum	alias	**EPS**
SeriesTerm	alias	**ST**

RSolve uses the remaining functions in an attempt to solve the given equations automatically. If this fails, **RSolve.m** can be used as a collection of tools to get partial results.

2.4 Basic Examples

2.4.1 RSolve

```
In[1]:=   << RSolve.m

In[2]:=   RS[{a[n] == a[n-1] + a[n-2] /; n >= 2,
             a[0] == a[1] == 1},
             a[n], n]

Out[2]=        1   Sqrt[5] 1 + n    1   Sqrt[5] 1 + n
              (- - -------)        (- + -------)
               2      2             2      2
          {{-(------------------ - ------------------)}}
                   Sqrt[5]              Sqrt[5]
```

These are the Fibonacci numbers:

```
In[3]:=   Table[%[[1,1]] // Simplify, {n,0,10}]

Out[3]=   {1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89}
```

The syntax of **RSolve** is patterned after that of **DSolve**, the arguments being a list of equations, a list of unknowns, and the independent variable. A single equation or a single unknown need not be given in a list. Equations can have the form **eqn /; cond** where **cond** is a boolean combination of inequalities and/or equalities specifying the range of validity for the equation. The result is a list of alternative solutions, each solution being a list of values for individual unknowns (given in the order the unknowns were listed in the input). Unlike **Solve** or **DSolve**, a solution is not a list of transformation rules, but rather a list of the values themselves. Here is an example of a system of two equations with two unknowns. When the range of validity is not given, it is assumed to be **n >= 0**.

```
In[4]:=   RS[{x[n+1] - 3 y[n] - 4 x[n] == 1,
              x[n+1] + y[n+1] + y[n] == n,
              x[0] == y[0] == 0},
             {x[n], y[n]}, n]

Out[4]=                 n     n                        n
              1    (-2)    3 2               2   (-2)      n
          {{-(-) - ----- + ---- - (1 + n), - + ----- - 2  + n}}
              3      6      2                3     3
```

As with **Solve** and **DSolve**, simplification of results is left to the user.

```
In[5]:=   % // Expand

Out[5]=                 n     n             n
              4    (-2)    3 2      2   (-2)      n
          {{-(-) - ----- + ---- - n, - + ----- - 2  + n}}
              3      6      2       3     3
```

RSolve can solve any linear constant-coefficient equation, as well as any system of such equations. It can solve a linear variable-coefficient equation in the following two cases:

1. the equation is first-order homogeneous and its coefficients are rational functions of **n**, or

2. the solution does not grow faster than **n!a^n** for some constant **a**, and **DSolve** can solve the associated differential equation.

Here is an example of the latter:

```
In[6]:=    RS[{a[0] == a[1] == 2,
            (n + 1)(n + 2) a[n+2] - 2 (n + 1) a[n+1] - 3 a[n] == 0},
            a[n], n]

Out[6]=          n      n
             (-1)      3
           {{----- + --}}
              n!      n!
```

RSolve can also solve nonlinear equations when the nonlinearity comes from a convolution. The Catalan numbers provide a famous example.

```
In[7]:=    RS[{c[n+1] == Sum[c[k] c[n-k], {k, 0, n}] /; n >= 0,
               c[0] == 1},
               c[n], n]

Out[7]=     Binomial[2 n, n]
           {{----------------}}
                 1 + n
```

Sometimes a convolution is not completely obvious.

```
In[8]:=    RS[{a[n] == Sum[k a[k-1], {k, n}] /; n > 0,
               a[0] == 1},
               a[n], n]

Out[8]=     (1 + n)!   If[n == 0, 1, 0]
           {{-------- + ----------------}}
               2               2


In[9]:=    Table[%[[1,1]], {n,0,6}]

Out[9]=    {1, 1, 3, 12, 60, 360, 2520}
```

2.4.2 PowerSum

PowerSum[expr, {z, n, n0:0}] computes **Sum[expr z^n, {n, n0, Infinity}]**. If **a[n]** is a sequence, the name **Gf[a][z]** is used by **PowerSum** to denote **Sum[a[n] z^n, {n, 0, Infinity}]**. **PowerSum** is threaded over lists and equations. Alias: **PS**.

```
In[10]:=   PS[n^2 + 4 n If[EvenQ[n], 2^n, 3^n] + 1, {z, n}]

Out[10]=     1      z (1 + z)            3 z               z
           ----- - --------- + 4 (-------------- - ----------- +
           1 - z           3                  2             2
                   (-1 + z)        2 (-1 - 3 z)    (-1 - 2 z)

                  z               3 z
             ----------- + --------------)
                       2                2
             (-1 + 2 z)    2 (-1 + 3 z)


In[11]:=   CoefficientList[Series[%, {z,0,10}], z]

Out[11]=   {1, 14, 37, 334, 273, 4886, 1573, 61286, 8257, 708670, 41061}


In[12]:=   Table[n^2 + 4 n If[EvenQ[n], 2^n, 3^n] + 1,
                 {n, 0, 10}]

Out[12]=   {1, 14, 37, 334, 273, 4886, 1573, 61286, 8257, 708670, 41061}
```

2.4.3 ExponentialPowerSum

ExponentialPowerSum[expr, {z, n, n0:0}] computes **Sum[expr z^(n-n0) / (n-n0)!, {n, n0, Infinity}]**. If **a[n]** is a sequence, the name **EGf[a][z]** is used by **ExponentialPowerSum** to denote **Sum[a[n] z^n / n!, {n, 0, Infinity}]**. **ExponentialPowerSum** is threaded over lists and equations. Alias: **EPS**.

```
In[13]:=   EPS[n^2 + 4 n If[EvenQ[n], 2^n, 3^n] + 1, {z, n}]
```

```
Out[13]=                                                   3 z
           z    z                3 z       z       2 z    3 E    z
          E  + E  z (1 + z) + 4 (------ - ---- + E    z + --------)
                                   3 z    2 z                2
                                 2 E     E
```

```
In[14]:=  CoefficientList[Series[%, {z,0,10}], z] Table[n!,
             {n,0,10}]

Out[14]=  {1, 14, 37, 334, 273, 4886, 1573, 61286, 8257, 708670, 41061}
```

2.4.4 GeneratingFunction

The syntax agrees with that of **RSolve** except that the user also has to provide an argument for the generating function(s).

```
In[15]:=  GF[{a[n] == a[n-1] + a[n-2] /; n >= 2,
             a[0] == a[1] == 1},
             a[n], n, z]

Out[15]=        1
          {{----------}}
                     2
            1 - z - z
```

```
In[16]:=  GF[{x[n+1] - 3 y[n] - 4 x[n] == 1,
              x[n+1] + y[n+1] + y[n] == n,
              x[0] == y[0] == 0},
             {x[n], y[n]}, n, z] // Factor

Out[16]=                         2
                      z (1 + 2 z )
          {{-(-------------------------------),
                      2
              (-1 + z)  (-1 + 2 z) (1 + 2 z)

                                   2
                     z (1 - 2 z + 4 z )
             -------------------------------}}
                      2
             (-1 + z)  (-1 + 2 z) (1 + 2 z)
```

```
In[17]:=  GF[{c[n+1] == Sum[c[k] c[n-k], {k,0,n}] /; n >= 0,
             c[0] == 1},
             c[n], n, z]

Out[17]=            -2   4     1
            -Sqrt[z    - -] + -
                         z     z
          {{------------------}}
                    2
```

```
In[18]:=  Together //@ %

Out[18]=    1 - Sqrt[1 - 4 z]
          {{-----------------}}
                   2 z
```

```
In[19]:=  GF[{a[n] == Sum[k a[k-1], {k,1,n}] /; n > 0,
              a[0] == 1},
              a[n], n, z]

Out[19]=    1 + HypergeometricF[{1, 2}, {}, z]
          {{----------------------------------}}
                            2
```

This sequence grows faster than exponentially, therefore its generating function is not an analytic function but just a formal power series.

2.4.5 ExponentialGeneratingFunction

The syntax agrees with that of **GeneratingFunction**.

Fibonacci numbers:

```
In[20]:=  EGF[{a[n] == a[n-1] + a[n-2] /; n >= 2,
              a[0] == a[1] == 1},
              a[n], n, z]
Out[20]=                     ((1 - Sqrt[5]) z)/2
            (5 - Sqrt[5]) E
          {{---------------------------------- +
                              10

                              ((1 + Sqrt[5]) z)/2
              (5 + Sqrt[5]) E
              ----------------------------------}}
                                10
```

Bernoulli numbers:

```
In[21]:=  EGF[Sum[Binomial[n,k] B[k], {k,0,n}] ==
              B[n] + If[n == 1, 1, 0],
              B[n], n, z]
Out[21]=       z
          {{-------}}
                  z
            -1 + E

In[22]:=  CoefficientList[Series[%[[1,1]], {z,0,10}], z] Table[n!,
              {n,0,10}]
Out[22]=        1   1       1       1       1       5
          {1, -(-), -, 0, -(--), 0, --, 0, -(--), 0, --}
                2   6      30      42      30      66

In[23]:=  Table[BernoulliB[n], {n,0,10}]
Out[23]=        1   1       1       1       1       5
          {1, -(-), -, 0, -(--), 0, --, 0, -(--), 0, --}
                2   6      30      42      30      66
```

Bell numbers:

```
In[24]:=  EGF[{b[n] == Sum[Binomial[n-1, k] b[k], {k ,0, n-1}] /;
                                                       n >= 1,
              b[0] == 1},
              b[n], n, z]
Out[24]=              z
             -1 + E
          {{E         }}

In[25]:=  CoefficientList[Series[%[[1,1]], {z,0,10}], z] Table[n!,
              {n,0,10}]
Out[25]=  {1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975}

In[26]:=  Table[Sum[StirlingS2[n, k], {k,0,n}], {n,0,10}]
Out[26]=  {1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975}
```

2.4.6 SeriesTerm

SeriesTerm[f, {x, a, n}] gives the n-th coefficient of the Laurent series of f expanded as a function of x around x = a. Here **n** can be symbolic.

```
In[27]:=  ST[E^x, {x, 0, n}]
Out[27]=  1
          --
          n!

In[28]:=  ST[1/(1 - x), {x, 0, n}]
```

```
Out[28]=   If[n >= 0, 1, 0]
```

The answer is not simply 1 since this would be wrong when **n** is negative.

```
In[29]:=   Table[%, {n, -3, 3}]
Out[29]=   {0, 0, 0, 1, 1, 1, 1}
```

Similarly,

```
In[30]:=   ST[1/(x-x^3), {x,0,n}]
Out[30]=                                                    n
                                 If[n >= 0, 1, 0]    (-1)  If[n >= 0, 1, 0]
           If[n == -1, 1, 0] + ---------------- - ----------------------
                                        2                      2

In[31]:=   Table[%, {n, -1, 10}]
Out[31]=   {1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0}
```

Often we have additional information about **n**. We can specify it using the function **Info** , and **SeriesTerm** will try to simplify the answers accordingly. For example, if we are only interested in nonnegative **n**, we should say

```
In[32]:=   n /: Info[n] = n >= 0
Out[32]=   n >= 0
```

Now the results

```
In[33]:=   ST[1/(1 - x), {x, 0, n}]
Out[33]=   1

In[34]:=   ST[1/(x-x^3), {x, 0, n}]
Out[34]=          n
           1   (-1)
           - - -----
           2     2
```

look simpler. The information we can give about an integer variable **n** must have the form of a Boolean combination of inequalities and/or equalities which involve no other variables beside **n**.

If we forget to give the information before we plunge into a lengthy computation, we can reuse the previous result by first specifying the information and then substituting **When** for **If**:

```
In[35]:=   %%%%% /. If -> When
Out[35]=          n
           1   (-1)
           - - -----
           2     2
```

Here are a few more examples (now all assuming **n >= 0**):

```
In[36]:=   ST[1/(1-x^3), {x,0,n}]
Out[36]=              2 Pi n
                2 Cos[------]
           1             3
           - + -------------
           3          3

In[37]:=   ST[1/(1 + x + x^2 + x^3), {x,0,n}]
Out[37]=                 Pi    Pi n
               n    Cos[-- + ----]
           (-1)         4      2
           ----- + --------------
             2          Sqrt[2]
```

```
In[38]:=  ST[(1 + 3 x + 17 x^2)/(1 + x + x^2 + x^3), {x,0,n}]
Out[38]=
                        Pi n           Pi n
              n   13 Cos[----]   19 Sin[----]
          15 (-1)          2              2
          -------- - ------------ + ------------
             2           2              2

In[39]:=  ST[(2 + 3 x)/(1 - x + 15 x^2), {x,0,n}]
Out[39]=                  1/2 + n/2               21
          -2 Sqrt[5] 15            Cos[ArcTan[--------] + (1 + n) ArcTan[Sqrt[59]]]
                                              Sqrt[59]
          ------------------------------------------------------------------------
                                            Sqrt[59]
```

We can use Series as a check:

```
In[40]:=  Table[% // N, {n,0,7}]
Out[40]=  {2., 5., -25., -100., 275., 1775., -2350., -28975.}

In[41]:=  CoefficientList[Series[(2 + 3 x)/(1 - x + 15 x^2),
              {x,0,7}], x]
Out[41]=  {2, 5, -25, -100, 275, 1775, -2350, -28975}

In[42]:=  Clear[n]
```

2.4.7 Another example

This example is taken from [MOW84]. Here a[n] denotes the number of binary strings of length n which do not contain 111 as a substring.

```
In[43]:=  GF[{a[0] == 1, a[1] == 2,
              b[n] == 0 /; 0 <= n <= 2,
              2 a[n] == a[n+1] + b[n+1] /; n >= 0,
              a[n] == b[n+1] + b[n+2] + b[n+3] /; n >= 0},
             {a[n], b[n]}, n, z]
Out[43]=                  2              3
             -1 - z - z                 z
          {{-(----------------), ---------------}}
                       2    3            2    3
              1 - z - z  - z     1 - z - z  - z

In[44]:=  RS[{a[0] == 1, a[1] == 2,
              b[n] == 0 /; 0 <= n <= 2,
              2 a[n] == a[n+1] + b[n+1] /; n >= 0,
              a[n] == b[n+1] + b[n+2] + b[n+3] /; n >= 0},
             {a[n], b[n]}, n]
Out[44]=                    n                  n
          {{1.13745 1.83929  + 0.282707 0.737353

               Cos[2.07853 + 2.17623 n],

                                n                  n
             0.0993883 1.83929  + 0.956392 0.737353

               Cos[0.34322 - 2.17623 n] - If[n == 0, 1, 0]}}

In[45]:=  Transpose[Table[%[[1]] // Chop, {n,0,10}]] // MatrixForm
Out[45]=  1.    2.    4.    7.    13.    24.    44.    81.    149.   274.   504.

          0     0     0     1.    1.     2.     4.     7.     13.    24.   44.
```

We note that the additive constant under the cosine function given as 2.078 in [MOW84] should be changed to 2.079.

The authors also need the n-th coefficient in the power series expansion of the following function:

```
In[46]:=  f = (z - 3 z^2 + 6 z^3 - 8 z^4 + 4 z^5 - z^7)/
              (1 - 4 z + 6 z^2 - 5 z^3 + 2 z^4 + z^5 - z^6 + z^7)

Out[46]=           2     3     4     5   7
            z - 3 z  + 6 z  - 8 z  + 4 z  - z
          -------------------------------------------
                       2     3     4   5   6   7
          1 - 4 z + 6 z  - 5 z  + 2 z  + z  - z  + z


In[47]:=  n/: Info[n] = n >= 0;
          ST[f, {z, 0, n}]

Out[47]=             1   Sqrt[5] 1 + n       -1 - Sqrt[5]
                    (- - -------)       (1 - ------------)
                 n   2      2                      4
          1.75488  + ------------------------------------ -
                                    Sqrt[5]

            1   Sqrt[5] 1 + n      -1 + Sqrt[5]
           (- + -------)       (1 - ------------)
            2      2                     4
           ------------------------------------ +
                          Sqrt[5]

                            n                        Pi n
           2. 0.754878  Cos[1.40772 n] - Cos[----] - If[n == 0, 1, 0]
                                                       3
```

We note that in [MOW84] 1 should be subtracted from the result when $n = 0$, and the values given as 1.7548 and 0.754 should be changed to 1.7549 and 0.755 (or 0.7549), respectively.

```
In[48]:=  Table[N[%] // Chop, {n,0,10}]

Out[48]=  {0, 1., 1., 4., 7., 11., 19., 36., 67., 121., 216.}
```

2.5 The Options

2.5.1 RSolve

```
In[49]:=  Options[RS]

Out[49]=  {Methods -> {MethodGF, MethodEGF}, PrecisionHS -> Automatic,

            PrecisionST -> Automatic, UseApart -> Automatic,

            MakeReal -> True, UseMod -> True, SpecialFunctions -> True}
```

Methods

Except for **Methods**, all the optional parameters of **RSolve** actually belong to **SeriesTerm** or to **(Exp)GeneratingFunction** and we shall discuss them there. The optional parameter **Methods** specifies which methods and in what order are to be tried. At present, there are two methods available - **MethodGF** and **MethodEGF**, and the user can add his or her own ones as explained in Section 2.9. By default, **RSolve** tries **MethodGF** first, and if this fails, it tries **MethodEGF**. **MethodGF** succeeds when either the solution does not grow faster than exponentially and **DSolve** can solve the associated differential equation, or the equation is homogeneous first-order linear and its coefficients are rational functions of the independent variable **n**. When neither of these is true, we can save time by going directly to **MethodEGF**. Example:

```
In[50]:=  Timing[RS[{T[0] == 5, 2 T[n] == n T[n-1] + 3 n! /; n > 0},
                    T[n], n]]

Series::esss:

    Essential singularity encountered in

              -2                           2
     Exp[----------------- + 2 + O[RSolve`Private`z] ].
         RSolve`Private`z
```

```
Series::esss:

    Essential singularity encountered in

                -2                            3
     Exp[---------------- + 2 + O[RSolve'Private'z] ].
         RSolve'Private'z




Series::esss:

    Essential singularity encountered in

                -2                            2
     Exp[---------------- + 2 + O[RSolve'Private'z] ].
         RSolve'Private'z




General::stop:

    Further output of Series::esss
    will be suppressed during this calculation.

Out[50]=                         1 n
         {85.74 Second, {{(3 + 2 (-) ) n!}}}
                                 2
```

This grows faster than exponentially (as indicated by the complaints from Series about an essential singularity) and the equation is inhomogeneous, so we should have known that MethodGF will not work. Now we use MethodEGF only:

```
In[51]:=  Timing[RS[{T[0] == 5, 2 T[n] == n T[n-1] + 3 n! /; n > 0},
                    T[n], n, Methods -> MethodEGF]]
Out[51]=                         1 n
         {17.96 Second, {{(3 + 2 (-) ) n!}}}
                                 2
```

The actual timings depend of course on the machine used and on other things, but their ratio should give an impression of the savings.

2.5.2 SeriesTerm

```
In[52]:=  Options[ST]

Out[52]=  {PrecisionST -> Automatic, UseApart -> 1, MakeReal -> True,

            UseMod -> True, SpecialFunctions -> True}
```

In what follows we assume that SeriesTerm is invoked as **SeriesTerm[expr, {z, a, n}, opts]**. Also,

```
In[53]:=  n /: Info[n] = n >= 0

Out[53]=  n >= 0
```

UseMod, PrecisionST and MakeReal

The role of these parameters is illustrated in the following examples.

```
In[54]:=  ST[1/(1 - x^4), {x, 0, n}]

Out[54]=          n                    n/2
          1    (-1)    If[EvenQ[n], (-1)    , 0]
          - + ----- + -------------------------
          4     4                2
```

```
In[55]:=  ST[1/(1 - x^4), {x, 0, n}, UseMod -> False]
Out[55]=                       Pi n
                     n   Cos[----]
          1    (-1)          2
          - + ----- + ---------
          4     4          2


In[56]:=  ST[1/(1 + x^4), {x,0,n}]
Out[56]=                                        n/4
          If[SymbolicMod[n, 4] == 0, (-1)    , 0]
```

Remark: SymbolicMod is used instead of Mod, since Mod[n, 4] simplifies to n.

```
In[57]:=  ST[1/(1 + x^4), {x,0,n}, UseMod -> False]
Out[57]=  0.5 Cos[0.785398 n] + 0.5 Cos[2.35619 n]


In[58]:=  ST[1/(1 + x^4), {x,0,n}, UseMod -> False, PrecisionST -> 24]
Out[58]=  0.5 Cos[0.785398163397448309615660 n] +

            0.5 Cos[2.35619449019234492884698 n]


In[59]:=  ST[1/(1 + x^4), {x,0,n}, UseMod -> False, MakeReal -> False]
Out[59]=                                n
          0.25 (-0.707107 + 0.707107 I)  +

                                          n
            0.25 (-0.707107 - 0.707107 I)  +

                                         n
            0.25 (0.707107 - 0.707107 I)  +

                                         n
            0.25 (0.707107 + 0.707107 I)


In[60]:=  ST[1/(1 + x^4), {x,0,n}, UseMod -> False, PrecisionST -> Infinity]
Out[60]=     Pi n        3 Pi n
          Cos[----]   Cos[------]
               4            4
          --------- + -----------
              2            2


In[61]:=  ST[1/(1 + x^4), {x,0,n}, UseMod -> False, PrecisionST -> Infinity,
                                                          MakeReal -> False]
Out[61]=        1     I 4 + n  2 + n/2         1     I 4 + n  2 + n/2
          -((-(-) - -)       2          )   (-(-) + -)       2
                2   2                          2   2
          --------------------------- - ------------------------ -
                        4                           4

             1   I 4 + n  2 + n/2    1   I 4 + n  2 + n/2
            (- - -)       2         (- + -)       2
             2   2                   2   2
            -------------------- - ---------------------
                     4                      4
```

A caveat

It is not recommended to use the option **PrecisionST -> Infinity** when the roots of cubic or quartic polynomials are found exactly, since this can be very time- and space-consuming. In such cases, **PrecisionST -> Infinity** should only be used in conjunction with **MakeReal -> False**.

UseApart

This parameter specifies whether or not **Apart** should be used on **expr** before attempting the expansion, and if yes, in what form:

UseApart -> None	do not use **Apart**,
UseApart -> Automatic (default)	use **Apart[expr]**,
UseApart -> All	use **Apart[expr, z]**.

MethodGF invokes **SeriesTerm** with **UseApart -> Automatic.**

MethodEGF invokes **SeriesTerm** with **UseApart -> None.**

```
In[62]:=  RS[{p[k] == (1-a) p[k-1] + b (1 - p[k-1]) /; k >= 1,
              p[0] == x},
              p[k], k]

Out[62]=                     1 + k
              (1 - a - b)         x
          {{-(-------------------) +
                 -1 + a + b

                 If[k >= 1, 1, 0]
              b (-(----------------) -
                        -a - b


                          k     1          a         b
              (1 - a - b)  (------ - ------ - ------) If[k >= 1, 1, 0]
                            -a - b   -a - b   -a - b
              ----------------------------------------------------------)}}
                                    -1 + a + b
```

This is correct, but unnecessarily complicated. When there are parameters in the input (like a, b, and x in this example), it is advisable to use the option **UseApart -> All** so that **Apart** will distinguish between the variable with respect to which expansion is done and other symbolic parameters.

```
In[63]:=  RS[{p[k] == (1-a) p[k-1] + b (1 - p[k-1]) /; k >= 1,
              p[0] == x},
              p[k], k, UseApart -> All]

Out[63]=                          k             k
                 b          (-1)  (-1 + a + b)  (b - a x - b x)
          {{-(------) + ------------------------------------}}
              -a - b                   -a - b

In[64]:=  %/.{a_^k b_^k :> Expand[a b]^k}

Out[64]=                                k
                 b          (1 - a - b)  (b - a x - b x)
          {{-(------) + ----------------------------}}
              -a - b                -a - b

In[65]:=  % //. a_ f_ + b_ f_ -> (a + b) f

Out[65]=                            k
               -b + (1 - (a + b))  (b - (a + b) x)
          {{-(-------------------------------------)}}
                             a + b

In[66]:=  % /. -(a_ - b_) -> b - a

Out[66]=                          k
            b - (1 - (a + b))  (b - (a + b) x)
          {{-----------------------------------}}
                          a + b
```

UseApart -> Automatic is useful when the function to be expanded is an algebraic function containing radicals.

UseApart -> None is recommended when the function to be expanded contains the exponential function applied to a factored expression. Example:

```
In[67]:=  ST[E^((1 + Sqrt[5])z), {z, 0, n}]

Out[67]=                 z + Sqrt[5] z
          SeriesTerm[E              , z, n]
```

```
In[68]:=  ST[E^((1 + Sqrt[5])z), {z, 0, n}, UseApart -> None]
Out[68]=                 n
          (1 + Sqrt[5])
          --------------
                n!

In[69]:=  Clear[n]
```

SpecialFunctions

At present, the only special functions used by SeriesTerm are the Legendre polynomials.

Example: Legendre polynomials

```
In[70]:=  RS[{P[0] == 1, P[1] == x,
              (n + 1) P[n+1] - (2 n + 1) x P[n] + n P[n-1] == 0 /; n > 0},
             P[n], n]
Out[70]=  {{LegendreP[n, x]}}

In[71]:=  RS[{P[0] == 1, P[1] == x,
              (n + 1) P[n+1] - (2 n + 1) x P[n] + n P[n-1] == 0 /; n > 0},
             P[n], n, SpecialFunctions -> False]
Out[71]=                -n + 2 K[1]  -n + 2 K[1]
          {{Sum[((-2)              x             Binomial[K[1], n - K[1]]

                                                K[1]
                Binomial[2 K[1], K[1]]) / (-4)     , {K[1], 0, n}]}}

In[72]:=  Table[%[[1,1]] // Together, {n, 0, 5}]
Out[72]=                2            3            2       4
                -1 + 3 x    -3 x + 5 x    3 - 30 x  + 35 x
          {1, x, ---------, -----------, -----------------,
                    2            2                8

                       3       5
            15 x - 70 x  + 63 x
            --------------------}
                     8

In[73]:=  Table[LegendreP[n, x], {n, 0, 5}]
Out[73]=                2            3            2       4
                -1 + 3 x    -3 x + 5 x    3 - 30 x  + 35 x
          {1, x, ---------, -----------, -----------------,
                    2            2                8

                       3       5
            15 x - 70 x  + 63 x
            --------------------}
                     8
```

Example: Middle trinomial coefficients

```
In[74]:=  RS[{a[0] == 1, a[1] == 1,
              (n+1) a[n+1] == (2n+1) a[n] + 3n a[n-1] /; n > 0},
             a[n], n]
Out[74]=        n/2                   1
          {{(-3)    LegendreP[n, --------]}}
                                 Sqrt[-3]
```

These are the coefficients of x^n in the expansion of (1 + x + x^2)^n.

```
In[75]:=  Table[%[[1,1]], {n,0,6}]
Out[75]=  {1, 1, 3, 7, 19, 51, 141}

In[76]:=  Table[Coefficient[Expand[(1 + x + x^2)^n], x^n],
             {n, 0, 6}]
```

```
Out[76]=  {1, 1, 3, 7, 19, 51, 141}

In[77]:=  RS[{a[0] == 1, a[1] == 1,
            (n+1) a[n+1] == (2n+1) a[n] + 3n a[n-1] /; n > 0},
            a[n], n, SpecialFunctions -> False]

Out[77]=                  K[1]   n - K[1]
          {{Sum[(-1)      3          Binomial[2 n - 2 K[1], n - K[1]]

                                                             n
                Binomial[2 K[1], K[1]], {K[1], 0, n}] / 4 }}

In[78]:=  Table[%[[1,1]], {n,0,6}]
Out[78]=  {1, 1, 3, 7, 19, 51, 141}
```

K[1], K[2], .. are summation indices used by SeriesTerm when the result involves sums.

```
In[79]:=  %% /. K[1] -> k
Out[79]=                 k  -k + n
          {{Sum[(-1)  3          Binomial[2 k, k]

                                                                n
                Binomial[-2 k + 2 n, -k + n], {k, 0, n}] / 4 }}
```

2.5.3 GeneratingFunction and ExpGeneratingFunction

```
In[80]:=  Options[GF]
Out[80]=  {PrecisionHS -> Automatic}

In[81]:=  Options[EGF]
Out[81]=  {PrecisionHS -> Automatic}
```

PrecisionHS has the same role in computing parameters of hypergeometric series as **PrecisionST** has in finding poles of rational generating functions. Examples:

```
In[82]:=  GF[{(n + 1) a[n+1] == (n^2 + n - 1) a[n], a[0] == 1},
             a[n], n, z]
Out[82]=                           1 + Sqrt[5]  -(-1 + Sqrt[5])
          {{HypergeometricF[{-----------, ---------------}, {}, z]}}
                                2              2

In[83]:=  GF[{(n + 1) a[n+1] == (n^2 + n - 1) a[n], a[0] == 1},
             a[n], n, z, PrecisionHS -> 8]
Out[83]=  {{HypergeometricF[{1.618034, -0.61803399}, {}, z]}}

In[84]:=  GF[{(n + 1) a[n+1] == (n^3 + n - 1) a[n], a[0] == 1},
             a[n], n, z]
Out[84]=  {{HypergeometricF[{-0.682328, 0.341164 - 1.16154 I,

             0.341164 + 1.16154 I}, {}, z]}}

In[85]:=  GF[{(n + 1) a[n+1] == (n^3 + n - 1) a[n], a[0] == 1},
             a[n], n, z, PrecisionHS -> Infinity]
```

```
Out[85]=                                     -1
         {{HypergeometricF[{-(-------------------- +
                                 1   Sqrt[31]  1/3
                              3 (- + ---------)
                                 2   6 Sqrt[3]

              1   Sqrt[31]  1/3
             (- + ---------)    ),
              2   6 Sqrt[3]

                                      1   Sqrt[31]  2/3
           (-1 + Sqrt[-3] + 3 (- + ---------)     +
                                      2   6 Sqrt[3]

                             1   Sqrt[31]  2/3        1   Sqrt[31]  1/3
             3 Sqrt[-3] (- + ---------)    ) / (6 (- + ---------)    )
                             2   6 Sqrt[3]            2   6 Sqrt[3]

                                       1   Sqrt[31]  2/3
            , (-1 - Sqrt[-3] + 3 (- + ---------)     -
                                       2   6 Sqrt[3]

                             1   Sqrt[31]  2/3        1   Sqrt[31]  1/3
             3 Sqrt[-3] (- + ---------)    ) / (6 (- + ---------)    )
                             2   6 Sqrt[3]            2   6 Sqrt[3]

           }, {}, z]}}
```

2.6 Other Functions in the Package

2.6.1 The list of all functions provided by RSolve.m

```
In[86]:=  ?RSolve`*
```

ArgPi	PoleMultiplicity
ConjugateQ	PowerSum
EGf	PowerSum
Entails	PrecisionHS
ExponentialGeneratingFunction	PrecisionST
ExponentialGeneratingFunction	RSolve
ExponentialPowerSum	RSolve
ExponentialPowerSum	ReP
FactorialSimplify	RealQ
FirstPos	Reset
FreeL	SafeFirst
GeneratingFunction	SafeSeries
GeneratingFunction	SeriesTerm
Gf	SeriesTerm
HSolve	SimplifyComplex2
HypergeometricF	SimplifyComplex3
ISolve	SimplifyComplex4
ImP	SimplifyComplexE1
Info	SimplifyComplexE2
K	SimplifyComplexN
LeadingCoef	SimplifySum
ListComplement	SimplifyTrig
MakeList	SpecialFunctions
MakeReal	SymbolicMod
MakeTrinomial	TTQ
MethodEGF	UseApart
MethodGF	UseMod
Methods	UserSymbols
PartialFractions	When
PatternList	

Remark: In the above list, the six functions which have aliases (**EGF**, **EPS**, **GF**, **PS**, **RS**, and **ST**) are listed twice.

All these functions have usage messages which can be seen by typing ? followed by the name of the function. Example:

```
In[87]:=  ?PatternList

PatternList[expr, pattern] returns the list of all
   subexpressions of expr which match the given pattern,
   provided that expr does not contain Rule[].
```

2.6.2 HSolve

HSolve finds formal power-series solutions to a linear differential equation whose terms are of the form **x^p D[y[x], {x,q}]** or **x^(p+1) D[y[x], {x,q}]** for some nonnegative integers **p**, **q**. The solution has to be expressible in the form **x^k (p[x] + x^m F[x])** where **F[x]** is a generalized hypergeometric function, **p[x]** is a polynomial, and **k**, **m** are integers.

Here is the hypergeometric differential equation.

```
In[88]:=  HSolve[x (1 - x) y''[x] + (c - (a + b + 1) x) y'[x] -
                a b y[x] == 0, y[x], x]

Out[88]=  {{y[x] -> C[1] Hypergeometric2F1[a, b, c, x]}}
```

A couple of "random" examples:

```
In[89]:=  HSolve[x^2 y''[x] + 2 x y'[x] - 3 y[x] == x + 1, y[x], x]

Out[89]=                 1
          {{y[x] -> -(-) - x}}
                      3


In[90]:=  HSolve[x^2 y'''[x] + 2 x y'[x] - 3 y[x] == x + 1, y[x], x]

Out[90]=                 1           2
          {{y[x] -> -(-) - x + x  C[1]
                      3

                                   1
                HypergeometricF[{-}, {2, 3}, -2 x]}}
                                   2
```

2.6.3 ISolve and When

ISolve solves Boolean combinations of inequalities and/or equalities among rational functions of a single integer variable. Here is an example:

```
In[91]:=  ISolve[(n^2 - 3 n + 10)/(n^3 + 1) <= 1/(5 n^2 + 2) &&
                 (n + 9)^2 > 1]

Out[91]=  n <= -11 || -7 <= n <= -2
```

When has the same syntax and semantics as If except that it checks whether the information given by the user implies the truth or falsity of the condition. The information is specified using the function Info.

```
In[92]:=  ??n

n

In[93]:=  When[n^2 >= 10 n, a, b]

Out[93]=      2
          If[n  >= 10 n, a, b]
```

No information has been given about n, so When simply turns into If.

```
In[94]:=  n /: Info[n] = n < 1

Out[94]=  n < 1
```

When assumes that n is an integer variable, so this actually means that n <= 0.

```
In[95]:=  When[n^2 >= 10 n, a, b]

Out[95]=  a


In[96]:=  n /: Info[n] = (n - 5)^2 <= 1

Out[96]=          2
          (-5 + n)  <= 1


In[97]:=  When[n^2 >= 10 n, a, b]

Out[97]=  b


In[98]:=  n /: Info[n] = n <= 1

Out[98]=  n <= 1


In[99]:=  When[n^2 >= 10 n, a, b]

Out[99]=      2
          If[n  >= 10 n, a, b]
```

Here both possibilities are still open.

```
In[100]:= Clear[n]
```

When can be used to take care of special cases which are left out in *Mathematica* 's "generic values" paradigm.

```
In[101]:= Integrate[x^n, x]
Out[101]=  1 + n
          x
          ------
          1 + n


In[102]:= Table[%, {n, -3, 3}]

                                       1
Power::infy: Infinite expression - encountered.
                                       0

Out[102]=                                            2   3   4
            -1     1                                x   x   x
          {----, -(-), ComplexInfinity, x, --, --, --}
             2     x                                2   3   4
           2 x
```

We can redefine **Integrate** in this case.

```
In[103]:= Unprotect[Integrate];
          Integrate[x_^n_, x_] := When[n == -1, Log[x], x^(n+1)/(n+1)];
          Protect[Integrate];


In[104]:= Integrate[x^n, x]
Out[104]=                          1 + n
                                  x
          If[n == -1, Log[x], ------]
                                 1 + n


In[105]:= Table[%, {n, -3, 3}]
Out[105]=                            2   3   4
            -1     1                x   x   x
          {----, -(-), Log[x], x, --, --, --}
             2     x                2   3   4
           2 x
```

To achieve this we could just as well have used **If** instead of **When**. The advantage is that the information we might have about the possible range of **n** will now be taken into account.

```
In[106]:= n /: Info[n] = n >= 0
Out[106]= n >= 0


In[107]:= Integrate[x^n, x]
Out[107]=  1 + n
          x
          ------
          1 + n
```

2.7 Limitations

RSolve.m currently uses only the method of (ordinary and exponential) generating functions. Therefore it cannot solve certain difference equations which can be solved by other methods. These include:

1. first-order linear equations unless

a) they are homogeneous with coefficients rational in **n**, or

b) the solution is **O[n! a^n]** for some constant **a**, and **DSolve** is able to solve the associated differential equation satisfied by the generating function;

2. equations which require domain or range transformations (examples: **a[2n] == a[n] + 1 /; n >= 1** can be solved by setting **n == 2^k, b[k] == a[2^k]**; the equation **a[n+1] == a[n]^2 /; n >= 0** can be solved by setting **b[n] = Log[a[n]]**);

3. nonlinear equations or higher-order equations of special form such as Bernoulli equations, Euler equations etc.

Section 2.8 below describes how the functions provided in the package can be used to obtain more information about the equation and its solution when the automatic solving routine **RSolve** fails. Section 2.9 describes how to add new methods to the repertoire of **RSolve**.

2.8 RSolve.m as a Collection of Tools

2.8.1 Interactive mode

RSolve tries to do everything automatically. Currently it uses the methods of ordinary and exponential generating functions which consist of three major steps:

1. Transform recurrences into functional equations for the generating functions using **PowerSum** (**ExponentialPowerSum**).

2. Solve these equations using **Solve**, **HSolve**, and/or **DSolve**.

3. Find the **n**-th Taylor coefficient of the generating functions using **SeriesTerm**. (Multiply it by **n!**.)

If **RSolve** returns **Fail** not all hope is lost. If Step 1 succeeded but Step 2 failed to solve the functional equations which the generating functions satisfy we can look at these equations using **PowerSum** or **ExponentialPowerSum** and try to solve them interactively.

Example: Alternating multipliers

If we start with 1 and at each step multiply the previous term by 2 we get the sequence **2^n**. If we use 3 instead of 2 we get **3^n**. What if we alternate the multipliers?

```
In[108]:= RS[{a[n+1] == If[EvenQ[n], 2, 3] a[n],
               a[0] == 1},
              a[n], n]

Out[108]= Fail
```

RSolve failed. Can we see the equation that the generating function satisfies?

```
In[109]:= PS[a[n+1] == If[EvenQ[n], 2, 3] a[n],
             {z, n}]

Out[109]=   a[0] - Gf[a][z]      -Gf[a][-z]   5 Gf[a][z]
          -(---------------) == ---------- + ----------
                  z                 2            2
```

Now we see what the problem is! there is just one equation but two unknowns: **Gf[a][z]** and **Gf[a][-z]**. We can get another equation by substituting -z for z:

```
In[110]:= Solve[{%, %/.z -> -z}, Gf[a][z], Gf[a][-z]]

Out[110]=                  (1 + 2 z) a[0]
          {{Gf[a][z] -> --------------}}
                                 2
                            1 - 6 z
```

It remains to substitute the initial condition and to peel the generating function out of its shell:

```
In[111]:= Gf[a][z] /. %[[1,1]] /. a[0] -> 1

Out[111]= 1 + 2 z
          --------
                 2
          1 - 6 z
```

This completes Step 2. Now we perform Step 3 using SeriesTerm:

```
In[112]:= n /: Info[n] = n >= 0;
          ST[%, {z, 0, n}]

Out[112]=     n/2           2              n           2
          -(6     (-1 - -------))    (-6)   (-1 + -------)
                        Sqrt[6]                   Sqrt[6]
          ---------------------- - ---------------------
                    2                       n/2
                                         2 6
```

Here is a check:

```
In[113]:= Table[% // Simplify, {n,0,10}]

Out[113]= {1, 2, 6, 12, 36, 72, 216, 432, 1296, 2592, 7776}
```

```
In[114]:= Table[%[[n]] / %[[n-1]], {n, 2, 11}]

Out[114]= {2, 3, 2, 3, 2, 3, 2, 3, 2, 3}
```

With some effort we can simplify the result and find out that **a[2k] = 6^k** and **a[2k+1] = 2 6^k**.

2.8.2 Reduction of order

If we know one solution of a linear difference equation then we can lower its order by one. Let sol[n] be the known solution and a[n] the unknown sequence. The recipe is to substitute b[n] sol[n] for a[n] and rewrite the equation in terms of the difference d[n] = b[n+1] - b[n]. Here is an example.

```
In[115]:= eqn = (n + 4) a[n+2] + a[n+1] - (n + 1) a[n] == 0;
```

Suppose we somehow discovered that 1/((n+1)(n+2)) solves this equation.

```
In[116]:= sol[n_] := 1/((n+1)(n+2))
```

```
In[117]:= Together /@ (eqn /. a[n_] -> sol[n])

Out[117]= True
```

Following the recipe we can find another, independent, solution.

```
In[118]:= eqn = eqn /. a[n_] -> b[n] sol[n]

Out[118]=   b[n]           b[1 + n]          b[2 + n]
          -(-----) + ---------------- + -------- == 0
            2 + n     (2 + n) (3 + n)     3 + n
```

Now we want to express this equation in terms of d[n] = b[n+1] - b[n].

```
In[119]:= Solve[{eqn,
                 d[n]   == b[n+1] - b[n],
                 d[n+1] == b[n+2] - b[n+1]},
                 d[n], {b[n], b[n+1], b[n+2]}]

Out[119]=                      1
          {{d[n] -> (-1 + -----) d[1 + n]}}
                          3 + n
```

As predicted by theory, we now have a first order equation. **RS** can solve it:

```
In[120]:= RS[d[n] == (d[n] /. %[[1,1]]), d[n], n]

Out[120]=         n          n
              (-1)      (-1)  (1 + n)
          {{(----- + --------------) d[0]}}
               2            2
```

We only need a particular solution, so we can pick any nonzero value for d[0] (note that the choice d[0] = 0 leads to a constant multiple of sol[n]).

```
In[121]:= Factor[%[[1,1]] /. d[0] -> 1]
```

```
Out[121]=      n
          (-1)  (2 + n)
          -------------
                2
```

This is d[n]; let us find b[n]. This time we are free to choose b[0] = 0.

```
In[122]:= RS[{% == b[n+1] - b[n], b[0] == 0}, b[n], n]
Out[122]=            n        n
            3   (-1)     (-1)  (1 + n)
          {{- - ----- - -------------}}
            8     8            4
```

We can also multiply this by any nonzero constant.

```
In[123]:= Collect[8 %[[1,1]], (-1)^n]
Out[123]=          n
          3 + (-1)  (-3 - 2 n)


In[124]:= Map[Factor, %, 2]
Out[124]=          n
          3 - (-1)  (3 + 2 n)
```

This is b[n]. Hence the general solution of the original equation is

```
In[125]:= sol[n](C[1] + C[2] %)
Out[125]=                    n
          C[1] + (3 - (-1)  (3 + 2 n)) C[2]
          ---------------------------------
                    (1 + n) (2 + n)
```

where C[1] and C[2] are arbitrary constants.

2.8.3 Finding recurrences for power-series coefficients of functions

Given an analytic function, can we find the recurrence satisfied by its Taylor coefficients? This is the inverse transformation of the one computed by **GeneratingFunction**. We can do this using **SeriesTerm**.

A rational function

This example is taken from [BGMP86]. The authors show that the so-called matching polynomial a[m][x] of a certain one-parametric family of graphs G[m] satisfies the following recurrence of order 4:

```
In[126]:= GF[{a[m] - (x^3 - 4x)a[m-1] + (2x^4 - 8x^2 + 4)a[m-2] -
               (x^5 - 2x^3 + 8x)a[m-3] + (x^4 + 4x^2)a[m-4] == 0 /; m >= 4,
               a[0] == 1,
               a[1] == x^3 - 2x,
               a[2] == x^6 - 8x^4 + 13x^2 - 2,
               a[3] == x^9 - 14x^7 + 58x^5 - 76x^3 + 22x},
               a[m], m, z] // Factor
Out[126]=                                    2
                                1 + 3 x z + 2 z
         {{-----------------------------------------------------------}}
                          3       2      2 2     4 2        3     3 3
           1 + 5 x z - x  z + 4 z  - 3 x  z  + x  z  - 4 x z  - x  z
```

Since the denominator is cubic in z there is a recurrence of order 3. Let us find it.

```
In[127]:= f[z] == %[[1,1]];
```

Multiply the equation by the denominator and gather terms on one side:

```
In[128]:= Numerator[Together[%[[1]] - %[[2]]]]
Out[128]=                  2                          3              2
          -1 - 3 x z - 2 z  + f[z] + 5 x z f[z] - x  z f[z] + 4 z  f[z] -
```

```
        2 2          4 2              3           3 3
   3 x  z  f[z] + x  z  f[z] - 4 x z  f[z] - x  z  f[z]
```

Now use SeriesTerm to find the recurrence and the initial conditions:

```
In[129]:= (ST[%, {z, 0, m}] /. ST[f[z], z, m_] -> a[m]) == 0
Out[129]= -If[m == 0, 1, 0] - 3 x If[m == 1, 1, 0] - 2 If[m == 2, 1, 0] -

                               3                                   2
            4 x a[-3 + m] - x  a[-3 + m] + 4 a[-2 + m] - 3 x  a[-2 + m] +

             4                               3
            x  a[-2 + m] + 5 x a[-1 + m] - x  a[-1 + m] + a[m] == 0
```

Here are the initial conditions:

```
In[130]:= Solve[Table[% /. a[_?Negative] -> 0, {m,0,2}], {a[0],a[1],a[2]}]
Out[130]=                                3
          {{a[0] -> 1, a[1] -> -2 x + x ,

                                  2        4    6                3
            a[2] -> -2 + 28 x  - 11 x  + x  + 3 x (-5 x + x )}}

In[131]:= Expand //@ %
Out[131]=                               3                           2       4    6
          {{a[0] -> 1, a[1] -> -2 x + x , a[2] -> -2 + 13 x  - 8 x  + x }}
```

And here is the recurrence:

```
In[132]:= m /: Info[m] = m >= 4; %%% /. If -> When
Out[132]=                   3                                  2
          -4 x a[-3 + m] - x  a[-3 + m] + 4 a[-2 + m] - 3 x  a[-2 + m] +

            4                              3
           x  a[-2 + m] + 5 x a[-1 + m] - x  a[-1 + m] + a[m] == 0

In[133]:= Collect[#, Table[a[m+i], {i,-3,0}]]& /@ %
Out[133]=           3                          2    4
          (-4 x - x ) a[-3 + m] + (4 - 3 x  + x ) a[-2 + m] +

                   3
            (5 x - x ) a[-1 + m] + a[m] == 0

In[134]:= Clear[m]
```

We can check this using SeriesTerm.

```
In[135]:= GF[Append[%%%%[[1]] /. (a_ -> b_) -> (a == b),
                  %% /; m >= 3], a[m], m, z] // Factor
Out[135]=                                                       2
                                      1 + 3 x z + 2 z
          {{-------------------------------------------------------}}
                            3          2      2 2    4 2        3     3 3
            1 + 5 x z - x  z + 4 z  - 3 x  z  + x  z  - 4 x z  - x  z
```

We got f[z] back.

An algebraic function

Take

```
In[136]:= expr = 1 / Sqrt[3 x^2 - 4 x + 1]; f[x] == expr
Out[136]=                 1
          f[x] == --------------------
                                     2
                  Sqrt[1 - 4 x + 3 x ]
```

and differentiate

```
In[137]:= D[#, x]& /@ %
Out[137]=                   -(-4 + 6 x)
          f'[x] == ---------------------
                                  2 3/2
                   2 (1 - 4 x + 3 x )
```

Eliminating the square root from these two equations, we get a differential equation which f satisfies.

```
In[138]:= % /. 1 / a_^(3/2) -> f[x] / a
Out[138]=          -((-4 + 6 x) f[x])
          f'[x] == ------------------
                                  2
                   2 (1 - 4 x + 3 x )
```

After getting rid of fractions, we use **SeriesTerm** and denote coefficients of **f[x]** by **a[n]**:

```
In[139]:= Numerator[Together[%[[1]] - %[[2]]]]
Out[139]=                                                  2
          -2 f[x] + 3 x f[x] + f'[x] - 4 x f'[x] + 3 x  f'[x]


In[140]:= ST[%, {x, 0, n}] /. ST[f[x], x, n_] -> a[n]
Out[140]= 3 a[-1 + n] + 3 (-1 + n) a[-1 + n] - 2 a[n] - 4 n a[n] +

            (1 + n) a[1 + n]


In[141]:= Collect[%, PatternList[%, a[_]]]
Out[141]= 3 n a[-1 + n] + (-2 - 4 n) a[n] + (1 + n) a[1 + n]
```

This is a second order recurrence, so we compute **a[0]** and **a[1]** (using **Series**).

```
In[142]:= Series[expr, {x, 0, 1}]
Out[142]=                2
          1 + 2 x + O[x]
```

Here is a check of the recurrence we obtained.

```
In[143]:= GF[{%% == 0 /; n >= 1, a[0] == 1, a[1] == 2},
                a[n], n, x]
Out[143]=                 1
          {{--------------------------}}
            Sqrt[1 - 3 x] Sqrt[1 - x]


In[144]:= % /. a_^x_ b_^x_ :> Expand[a b]^x
Out[144]=              1
          {{---------------------}}
                               2
            Sqrt[1 - 4 x + 3 x ]
```

This is f[x] we started with.

Another way to get rid of the square root is by squaring. This leads to a recurrence with convolutions.

```
In[145]:= f[x]^2 == expr^2
Out[145]=      2           1
          f[x]  ==  --------------
                                2
                    1 - 4 x + 3 x


In[146]:= Numerator[Together[%[[1]] - %[[2]]]]
Out[146]=          2            2      2     2
          -1 + f[x]  - 4 x f[x]  + 3 x  f[x]
```

```
In[147]:= Collect[%, f[x]]
Out[147]=
                             2       2
          -1 + (1 - 4 x + 3 x ) f[x]


In[148]:= ST[%, {x, 0, n}] /. ST[f[x], x, n_] -> a[n]
Out[148]= -If[n == 0, 1, 0] + Sum[a[n - K[1]] a[K[1]], {K[1], 0, n}] -

            4 Sum[a[-1 + n - K[2]] a[K[2]], {K[2], 0, -1 + n}] +

            3 Sum[a[-2 + n - K[3]] a[K[3]], {K[3], 0, -2 + n}]


In[149]:= GF[{% == 0 /; n >= 2, a[0] == 1, a[1] == 2},
              a[n], n, x]
Out[149]=                1
          {{---------------------}}
                               2
            Sqrt[1 - 4 x + 3 x ]
```

2.8.4 Partial difference equations

The recipe

To solve a difference equation for **a[n,k]**, we can simply treat one of **n, k** as a parameter and solve with respect to the other. For the method of generating functions, this means that on Step 2 the functional equation for the generating function is itself a difference equation. Hence the recipe is as follows:

1. State in advance what is known about the range and type of indices using functions such as **Info** and **IntegerQ**.

2. Write **a[n,k]** as **a[n][k]**. Use **PS** (or **EPS**) with respect to **k** to get a difference equation for **Gf[a[n]][z]** (or **EGf[a[n]][z]**).

3. Rewrite **Gf[a[n]][z]** (**EGf[a[n]][z]**) as **h[n]** where **h** is any head; for example, rewrite it as **a[z][n]**. Use **RS** to solve the difference equation for **a[z][n]**.

4. Use **ST** to find the k-th term in the expansion of **a[z][n]** with respect to **z**.

We have the freedom to choose any ordering of the indices. Also, we have the freedom to compute many different generating functions: for every index, we can either leave it as a parameter or compute the ordinary or the exponential generating function with respect to it.

Example: Nondecreasing digit strings

Let **a[n, k]** be the number of strings of length **k** formed from **n** digits with the additional restriction that the digits in the string must not decrease from left to right. Considering the choice of the first digit we obtain the following recurrence:

a[n, k] == Sum[a[j, k-1], -j, n"] /; n, k >= 1

a[n, 0] == 1 /; n >= 1

These equations define **a[n,k]** for **n >= 1, k >= 0.** It is advisable to specify the information that we have about the range and type of the indices.

```
In[150]:= n /: Info[n] = n >= 1;
          k /: Info[k] = k >= 0;
          n /: IntegerQ[n] = True;
          k /: IntegerQ[k] = True;
```

We want the ordinary generating function with respect to **k**. The corresponding variable will be **y**.

```
In[151]:= PS[a[n][k] == Sum[a[j][k-1], {j, n}], {y, k, 1}]

Out[151]= Gf[a[n]][y] - a[n][0] == y Sum[Gf[a[j]][y], {j, 1, n}]
```

The initial condition is **a[n][0] == 1** . At the same time, we rename the generating function so that its argument will be **n**.

```
In[152]:= recur = % /. {a[_][0] -> 1, Gf[a[n_]][y] -> a[y][n]}
Out[152]= -1 + a[y][n] == y Sum[a[y][j], {j, 1, n}]
```

Now we solve this as a recurrence in **n**.

```
In[153]:= RS[recur /; n >= 1, a[y][n], n]
Out[153]=           -1 + n
               (-1)
          {{-(----------)}}
                      n
               (-1 + y)
```

Finally, we compute the **k**-th coefficient in the expansion with respect to **y**; this gives **a[n,k]**.

```
In[154]:= ST[%, {y, 0, k}]
Out[154]= {{Binomial[-1 + k + n, k]}}
```

We can also compute the bivariate ordinary generating function for **a[n,k]**. The variable associated with n will be **x**.

```
In[155]:= GF[recur /; n >= 1, a[y][n], n, x]
Out[155]=       x
          {{---------}}
            1 - x - y
```

Example: King's moves

The aim is to compute **a[n, k]** where

a[n+1, k+1] == a[n, k] + a[n, k+1] + a[n+1, k+1] /; n, k >= 0,

a[n, 0] == a[0, k] == 1 /; n, k >= 0

Here **a[n, k]** is the number of ways the chess king can reach the square **(n, k)** starting at **(0, 0)** if at each step he can move either up one or right one or both (i.e, diagonally).

```
In[156]:= n /: Info[n] = n >= 0;
          k /: Info[k] = k >= 0;
          n /: IntegerQ[n] = True;
          k /: IntegerQ[k] = True;

In[157]:= PS[a[n+1][k+1] == a[n][k] + a[n][k+1] + a[n+1][k], {x, k}]

Out[157]= Gf[a[1 + n]][x] - a[1 + n][0]
          ------------------------------ ==
                        x

                                              Gf[a[n]][x] - a[n][0]
            Gf[a[n]][x] + Gf[a[1 + n]][x] + ---------------------
                                                      x
```

The initial condition is **a[n][0] == 1** .

```
In[158]:= recur = % /. a[_][0] -> 1
Out[158]= -1 + Gf[a[1 + n]][x]     -1 + Gf[a[n]][x]
          -------------------- == ---------------- + Gf[a[n]][x] + Gf[a[1 + n]][x]
                   x                      x
```

This is a difference equation for **Gf[a[n]][x]** . What is the initial condition?

Since **a[0][k] == 1** , it is

```
In[159]:= init = Gf[a[0]][x] == PS[1, {x, k}]
Out[159]=                      1
          Gf[a[0]][x] == -----
                         1 - x
```

We have to rewrite **Gf[a[n]][x]** in the form **h[n]** , where **h** is any head.

```
In[160]:= {recur, init} = {recur, init} /. Gf[a[n_]][x] -> a[x][n]
Out[160]=  -1 + a[x][1 + n]     -1 + a[x][n]                                        1
          {---------------- == ------------ + a[x][n] + a[x][1 + n], a[x][0] == -----}
                 x                  x                                             1 - x
```

Now solve for **a[x][n]** :

```
In[161]:= RS[{recur, init}, a[x][n], n] // Factor
Out[161]=       1 + n        n
            (-1)      (1 + x)
          {{------------------}}
                       1 + n
             (-1 + x)
```

This has to be expanded as a function of **x** .

```
In[162]:= ST[%[[1,1]], {x, 0, k}]
Out[162]= Sum[Binomial[n, K[1]] Binomial[k + n - K[1], k - K[1]], {K[1], 0, k}]
```

This is **a[n,k]** . Differently from previous example it is not in closed form since it contains a sum. Before we quit let us see what happens on the diagonal, i.e., when **k = n** .

```
In[163]:= %/. k -> n
Out[163]= Sum[Binomial[n, K[1]] Binomial[2 n - K[1], n - K[1]], {K[1], 0, n}]
```

What is the generating function of **a[n, n]** ?

```
In[164]:= PS[%, {z, n}]
Out[164]=          1
          ------------------
                          2
          Sqrt[1 - 6 z + z ]
```

And here is **a[n, n]** - in closed form (well, sort of):

```
In[165]:= ST[%, {z,0,n}]
Out[165]= LegendreP[n, 3]
```

We can also get the "master" generating function depending on **x** and **y**:

```
In[166]:= Together[GF[{recur, init}, a[x][n], n, y]]
Out[166]=         1
          {{---------------}}
            1 - x - y - x y
```

2.9 Expanding the System

2.9.1 Adding new rules to PowerSum and ExpPowerSum

1/Sqrt[1 - 2 x z + z^2] is the ordinary generating function for Legendre polynomials. **SeriesTerm** knows about this, but **PowerSum** does not:

```
In[167]:= n /: Info[n] = n >= 0;
          ST[1/Sqrt[1 - 2 x z + z^2], {z,0,n}]
Out[167]= LegendreP[n, x]
```

```
In[168]:= PS[%, {z, n}]
Out[168]= PowerSum[LegendreP[n, x], z, n]
```

We can add a corresponding rule, using the "internal syntax" of **PS** (with three arguments, assuming the range of summation index starts at zero). To make it more general, we allow for a shift in the index of the polynomial.

```
In[169]:= PS[LegendreP[n_ + a_., x_], z_, n_] :=
              Block[{nn},
                  (1/Sqrt[1 - 2 x z + z^2] -
                   Sum[LegendreP[nn, x] z^nn, {nn, 0, a - 1}] ) /
                   z^a ] /; IntegerQ[a] && a >= 0
```

Let us try it out:

```
In[170]:= PS[LegendreP[n, 3], {z, n}]
Out[170]=         1
          -----------------
                         2
          Sqrt[1 - 6 z + z ]
```

Adding new rules to **ExpPowerSum** is analogous.

2.9.2 Adding new rules to SeriesTerm

There are many functions in *Mathematica* SeriesTerm does not know how to expand. For example:

```
In[171]:= ST[Sin[x], {x,0,n}]
Out[171]= SeriesTerm[Sin[x], x, n]
```

Let us add a rule for **Sin[x]**. We have to use the "internal syntax" of **ST** shown in the output above - with three arguments:

ST[Sin[x_], x_, n_] := ...

By assumption, we are expanding around **x = 0**. What is the **n**-th Taylor coefficient of **Sin[x]**? Maybe **ST** can help us out after all, since it knows how to expand **Exp[x]**:

```
In[172]:= ST[(E^(I x) - E^(-I x))/(2 I), {x,0,n}]
Out[172]= I      n    -I  n
          - (-I)      -- I
          2           2
          ------- + -----
            n!        n!
```

If we do not like real quantities to be expressed in terms of complex ones, we can use the appropriate one of the **SimplifyComplex** rulesets to make everything real. But first we have to tell the system that **n** is real (saying that it is integer would also work):

```
In[173]:= n /: RealQ[n] = True;
          % //. SimplifyComplexE1
Out[173]=        -(Pi n)
            Sin[-------]
                    2
          -(------------)
                 n!
```

```
In[174]:= % /. SimplifyTrig
Out[174]=     Pi n
          Sin[----]
                2
          ---------
             n!
```

So, here is our new rule. To make it more general, we add a factor to the argument:

```
In[175]:= ST[Sin[a_. x_], x_, n_] := a^n Sin[Pi n / 2] / n! /;
                                                  FreeQ[a, x]
```

Let us try it out:

```
In[176]:= ST[Sin[2x], {x,0,n}]

Out[176]=  n     Pi n
          2  Sin[----]
                  2
          ------------
              n!
```

In the same way, we find the rule for **Cos[x]**:

```
In[177]:= ST[Cos[a_. x_], x_, n_] := a^n Cos[Pi n / 2] / n! /;
                                                     FreeQ[a, x]
```

```
In[178]:= ST[Cos[x]^2, {x,0,n}]

Out[178]=          Pi (n - K[1])      Pi K[1]
               Cos[-------------] Cos[-------]
                         2               2
          Sum[-------------------------------, {K[1], 0, n}]
                    (n - K[1])! K[1]!
```

2.9.3 Adding new methods to RSolve

At present, **RSolve** uses two methods: **MethodGF** (ordinary generating functions) and **MethodEGF** (exponential generating functions). This means, among other things, that it is unable to solve many linear equations of first order although it is trivial to do so. Here is an example:

```
In[179]:= RS[{a[n + 1] == (n + 1)^2 a[n] + 1, a[0] == 1}, a[n], n]

DSolve::NotYet:

    Built-in procedures cannot solve this differential
    equation.



DSolve::NotYet:

    Built-in procedures cannot solve this differential
    equation.



DSolve::NotYet:

    Built-in procedures cannot solve this differential
    equation.



General::stop:

    Further output of DSolve::NotYet
    will be suppressed during this calculation.

Out[179]= Fail
```

The differential equation in both cases is nonhomogeneous of second order, therefore **DSolve** cannot handle it as yet. But even if it could it would not help since solutions to these differential equations have an essential singularity at z = 0. This is a consequence of the fast growth of **a[n]** (it grows approximately as **n!^2**). Both generating functions are just formal power series.

We can add a new method, called **MethodL1**, as a "quick and dirty" first-order linear equation solver. Here is the general recipe for adding new methods.

RSolve does the parsing and then evokes the methods (as specified by the optional parameter **Methods**) one by one until one succeeds. The arguments passed to a particular method by **RSolve** are:

recur - a list of equations, all valid for **n >= 0**;

conds - a list of initial conditions;

unknowns - a list containing the heads of all unknown sequences;

startValues - a list of starting values for **n**, one for each of the unknown sequences, in the order these are listed in **unknowns**;

n - the independent variable;

opts - the optional parameters that **RSolve** received.

The function **Reset[recur, conds, unknowns, startValues, n0]** can be used to reset the unknown sequences so that they all start with **n = n0**. Of course, this must be compensated for in the final result.

The result should be a double list in the manner of **Solve** (but containing just values, not rules). If the method fails, it should return **Fail**.

```
In[180]:= MethodL1[recur_, conds_, unknowns_, startValues_, n_,
           opts___Rule] :=
               Block[{eqn, a, rec, con, sv, kk},

               (* reset the unknown sequences so that they start
                  with n = 0 *)

                  {rec, con} = Reset[recur, conds, unknowns,
                               startValues, 0];

               (* this method works only for a single equation
                  and a single unknown *)

                   If[Length[rec] != 1, Return[Fail],
                       eqn = First[rec]];
                   If[Length[unknowns] != 1, Return[Fail],
                       a = First[unknowns]];
                   If[Length[startValues] != 1, Return[Fail],
                       sv = First[startValues]];

               (* express a[n+1] in terms of a[n] *)

                   sub = Solve[eqn, a[n+1]];
                   If[Length[sub] != 1, Return[Fail],
                                        sub = First[sub] ];
                   rhs = Collect[a[n+1] /. sub, a[n]];

                (* solution formula *)

                   sol = Replace[rhs,   c_. a[n] + d_. :>
                        a[0] Product[c /. n -> j, {j, 0, n - 1}] +
                        Sum[(d /. n -> k) Product[c /. n -> j, {j, k+1, n-1}],
                        {k, 0, n - 1} ]  /; FreeQ[{c, d}, a]];

                (* find a[0] from the initial condition *)

                   sol = {sol} /. Solve[con /.
                        (a[v_] :> (sol /. n :> v)), a[0]];

                (* restore the index *)

                   sol = sol /. n -> n - sv;
                   Return[sol /. a[k_] :> a[Expand[k + sv]] ]

                ];
```

Let us try this on our example:

```
In[181]:= RS[{a[n + 1] == (n + 1)^2 a[n] + 1, a[0] == 0}, a[n], n,
               Methods -> MethodL1]
Out[181]=                                 2
           {{Sum[Product[1 + 2 j + j , {j, 1 + k, -1 + n}], {k, 0, -1 + n}]}}


In[182]:= Table[%[[1,1]], {n,0,6}]
Out[182]= {0, 1, 5, 46, 737, 18426, 663337}
```

It works! But I called this solution "quick and dirty" because there are several problems with it, for instance:

1. There is no simplification of products and sums, e.g. the above solution can be written as **n!^2 Sum[1/k!^2, {k, n}]**.

2. The result will be incorrect if the product has factors with integer poles, e.g., if the equation is something like

```
In[183]:= RS[(n-3)a[n + 1] == (n + 1)^3 a[n], a[n], n, Methods -> MethodL1]
Out[183]=                                                   2        3
                                  1        3 j       3 j       j
        {{Sum[0, {k, 0, -1 + n}] + Product[------ + ------ + ------ + ------,
                                  -3 + j   -3 + j   -3 + j   -3 + j

              {j, 0, -1 + n}] a[0]}}

In[184]:= Table[%[[1,1]], {n,0,6}]

                                 1
Power::infy: Infinite expression - encountered.
                                 0

                                 1
Power::infy: Infinite expression - encountered.
                                 0

                                 1
Power::infy: Infinite expression - encountered.
                                 0

General::stop:

    Further output of Power::infy
    will be suppressed during this calculation.

Infinity::indt:

    Indeterminate expression ComplexInfinity + ComplexInfinity +
    ComplexInfinity + ComplexInfinity encountered.

Infinity::indt:

    Indeterminate expression ComplexInfinity + ComplexInfinity +
    ComplexInfinity + ComplexInfinity encountered.

Infinity::indt:

    Indeterminate expression ComplexInfinity + ComplexInfinity +
    ComplexInfinity + ComplexInfinity encountered.
```

```
General::stop:

    Further output of Infinity::indt
    will be suppressed during this calculation.

Out[184]=          -a[0]   4 a[0]
          {a[0], -----, ------, -36 a[0], Indeterminate, Indeterminate, Indeterminate}
                   3       3
```

By the way, this example can be handled by the built-in methods, since the equation is homogeneous:

```
In[185]:= RS[(n - 3)a[n + 1] == (n + 1)^3 a[n], a[n], n]
Out[185]=                          3
                                 n!
          {{If[n >= 4, ----------------, 0] a[4]}}
                       13824 (-4 + n)!

In[186]:= Table[%[[1,1]], {n,0,6}]
Out[186]= {0, 0, 0, 0, a[4], 125 a[4], 13500 a[4]}
```

Note that any solution to this equation must have **a[3]**, and therefore all the preceding terms, equal to zero.

2.10 If it doesn't work . . .

2.10.1 The input expression is returned unevaluated

Check the syntax against the description in the usage message. One common syntax error is to omit the generating function argument in **GF** or **EGF**.

```
In[187]:= <<RSolve.m

In[188]:= GF[a[n+1] == c a[n], a[n], n]
Out[188]= GeneratingFunction[a[n + 1] == c a[n], a[n], n]

In[189]:= GF[a[n+1] == c a[n], a[n], n, z]
Out[189]=    a[0]
          {{-------}}
            1 - c z
```

2.10.2 RS (or GF, or EGF) returns Fail

This means that the function was not able to handle the given equations. Perhaps rewriting the equations in a different way or changing the values of optional parameters might help. If not, use **PS** and/or **EPS** on the recurrence(s) to see if at least Step 1 (i.e., transformation of recurrences into functional equations for the generating functions) was successful. If yes, there is a chance that these equations might be solved with some help from the user, or that they could be useful in some other way. See the first example in Section 2.8.

It should be noted that an answer of the form {} means that the given equation or system of equations has no solution.

2.10.3 Series::esss

This message means that the solution grows too fast causing the generating function to have an essential singularity at the origin. In the default setting **RSolve** will not give up but proceed with the method of exponential generating functions. If the solution grows faster than **a^n** for any constant **a**, but not faster than **n! b^n** for some constant b we might see this message but nevertheless get the solution in the end (see the example in Subsection 2.5.1).

2.10.4 DSolve::NotYet

This message means that **DSolve** was not able to solve the differential equations satisfied by the generating functions. See the example in Subsection 2.9.3. We might still get the solution if the equation for the exponential generating function is simpler than the one for the ordinary generating function.

Chapter 3

Partial difference equations

A generating function is a clothesline
on which we hang up a sequence of numbers for display.
— HERBERT S. WILF, *generatingfunctionology (1990)*

3.1 Introduction

We shall denote points in $\mathbb{R}^d$ with small boldface letters, and their coordinates with the corresponding plain letters indexed by coordinate names. For example, $\mathbf{p} = (p_1, p_2, \ldots, p_d)$ and $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{id})$. The canonical basis for $\mathbb{R}^d$ will be denoted by $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d\}$.

Let $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$. We shall write $\mathbf{p} \le \mathbf{q}$ if $p_i \le q_i$ for $i = 1, 2, \ldots, d$, and $\mathbf{p} < \mathbf{q}$ if $p_i < q_i$ for $i = 1, 2, \ldots, d$.

Let A be a nonempty set. We shall consider d-dimensional partial difference equations of the form

$$a_{\mathbf{p}} = F(a_{\mathbf{p}+\mathbf{z}_1}, a_{\mathbf{p}+\mathbf{z}_2}, \ldots, a_{\mathbf{p}+\mathbf{z}_k}), \quad \text{for } \mathbf{p} \ge \mathbf{s}, \tag{3.1}$$

where $\mathbf{s} \in \mathbb{N}^d$ and $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k \in \mathbb{Z}^d$ are given points such that $\mathbf{s} + \mathbf{z}_i \in \mathbb{N}^d$ for $i = 1, 2, \ldots, k$, where $a : \mathbb{N}^d \to A$ is the unknown sequence, and $F : A^k \to A$ is a given function. Let $I := \{\mathbf{p} \in \mathbb{N}^d;\ \mathbf{p} \not\ge \mathbf{s}\}$ be the *initial set* for (3.1). We assume that the initial conditions are of the form

$$a_{\mathbf{p}} = f(\mathbf{p}), \quad \text{for } \mathbf{p} \in I, \tag{3.2}$$

where $f : I \to A$ is a given function.

The purpose of this chapter is to determine for which sets $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\} \subseteq \mathbb{Z}^d$ the equation (3.1) together with the initial conditions (3.2) has a unique solution. To be able to do so we need some properties of convex sets.

3.2 Some Properties of Convex Sets

Definition 4 Let $S \subseteq \mathbb{R}^d$. The *convex hull* of S is the set

$$\operatorname{conv} S = \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} = \sum_{i=0}^{k} \lambda_i \mathbf{s}_i,\ \lambda_i \in \mathbb{R},\ \lambda_i \ge 0,\ \sum_{i=0}^{k} \lambda_i = 1, \mathbf{s}_i \in S\}. \quad \square$$

Definition 5 Let $S \subseteq \mathbb{R}^d$. The *convex cone* generated by S is the set

$$\operatorname{cone} S = \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} = \sum_{i=0}^{k} \lambda_i \mathbf{s}_i,\ \lambda_i \in \mathbb{R},\ \lambda_i \geq 0,\ \mathbf{s}_i \in S\}\,. \quad \square$$

Definition 6 Let $S \subseteq \mathbb{R}^d$. The *integer cone* of S is the set

$$\operatorname{icon} S = \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} = \sum_{i=0}^{k} \lambda_i \mathbf{s}_i,\ \lambda_i \in \mathbb{N},\ \mathbf{s}_i \in S\}\,. \quad \square$$

Definition 7 Let $S \subseteq \mathbb{R}^d$. The *polar cone* of S is the set

$$S^p = \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \cdot \mathbf{s} \leq 0, \forall \mathbf{s} \in S\}\,. \quad \square$$

Definition 8 The *relative interior* of a set $S \subseteq \mathbb{R}^d$ is the interior of S in the relative topology of the affine subspace generated by S in $\mathbb{R}^d$. $\square$

Lemma 1 *If C is the convex cone generated by a finite set of points in $\mathbb{R}^d$ then $C^{pp} = C$.*

For a proof, see, for example, Lemma 2.7.9 in [SW70].

Theorem 3 *Any two nonempty disjoint convex polyhedra $P_1, P_2 \subseteq \mathbb{R}^d$ can be separated by a hyperplane H such that $H \cap P_1 = H \cap P_2 = \emptyset$.*

For a proof, see, for example, Theorem 2.12.9 in [SW70].

Theorem 4 *Two nonempty convex sets $K_1, K_2 \subseteq \mathbb{R}^d$ can be separated by a hyperplane H such that $K_1 \cup K_2 \not\subseteq H$ if and only if their relative interiors are disjoint.*

For a proof, see, for example, Theorem 3.3.9 in [SW70].

We shall also need some facts about rational points in convex sets in $\mathbb{R}^d$.

Definition 9 A point in $\mathbf{p} \in \mathbb{R}^d$ is *rational* if $p_i \in \mathbb{Q}$ for $i = 1, 2, \ldots, d$. A convex set $X \subseteq \mathbb{R}^d$ is *rational* if it is equal to the convex hull of its rational points. $\square$

Lemma 2 *Let $X \subseteq \mathbb{R}^d$ be a set of rational points, and $\mathbf{p} \in \operatorname{conv} X$. Then $\mathbf{p}$ is rational if and only if $\mathbf{p}$ can be expressed as a rational convex combination of points from X.*

Proof: If $\mathbf{p}$ is a rational convex combination of rational points then clearly $\mathbf{p}$ is rational.

Conversely, let $\mathbf{p} \in \operatorname{conv} X$ be rational. If $\mathbf{p} \in X$ the assertion is trivial. Otherwise let $S \subseteq X$ be a minimal subset with the property that $\mathbf{p} \in \operatorname{conv} S$. Such a subset exists because $\mathbf{p}$ is in the convex hull of some finite subset of X, and $\mathbf{p} \neq \mathbf{q}$ for all $\mathbf{q} \in X$. Let e be the affine dimension of S. By Carathéodory's theorem and by minimality of S, we have $|S| \leq e+1$. But $e+1$ equals the maximum number of affinely independent points in S, by definition of e. So $e+1 \leq |S|$, and S is affinely independent. Therefore conv S is an e-simplex containing $\mathbf{p}$.

Let $S = \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_e\}$ and $S_i = S - \{\mathbf{q}_i\} \cup \{\mathbf{p}\}$, for $i = 0, 1, \ldots, e$. Then

$$\mathbf{p} = \sum_{i=0}^{e} \lambda_i \mathbf{q}_i ,$$

where

$$\lambda_i = \frac{e\text{-vol}(\text{conv } S_i)}{e\text{-vol}(\text{conv } S)} > 0 , \qquad \text{for } i = 0, 1, \ldots, e .$$

Since the volume of a simplex can be expressed as a fraction of a determinant whose nonunit entries are the coordinates of its vertices (see, for example, [Ken61]), any simplex with rational vertices has rational volume. It follows that all the λ_i are rational. □

Lemma 3 *In a rational convex set rational points are dense.*

Proof: Let X be a rational convex set, $\mathbf{x} \in X$, and $\varepsilon > 0$. We claim that there is a rational point $\mathbf{p} \in X$ such that $\|\mathbf{x} - \mathbf{p}\| < \varepsilon$.

By Definition 9 there exist rational points $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_k \in X$ and nonnegative real numbers $\lambda_0, \lambda_1, \ldots, \lambda_k$ such that $\sum_{i=0}^{k} \lambda_i = 1$ and $\mathbf{x} = \sum_{i=0}^{k} \lambda_i \mathbf{q}_i$. If all the λ_i are rational take $\mathbf{p} = \mathbf{x}$. Otherwise assume without loss of generality that λ_0 is irrational. Let $M = \max_{0 \leq i \leq k} \|\mathbf{q}_i\|$. For $i = 1, 2, \ldots, k$ choose $\delta_i > 0$ such that $\delta_i < \min\{\lambda_0/k, \varepsilon/(2kM)\}$ and $\lambda_i + \delta_i$ is rational. Then the point $\mathbf{p} = (\lambda_0 - \sum_{i=1}^{k} \delta_i)\, \mathbf{q}_0 + \sum_{i=1}^{k} (\lambda_i + \delta_i)\, \mathbf{q}_i$ is rational, belongs to X, and $\|\mathbf{x} - \mathbf{p}\| < \varepsilon$. □

3.3 Well-Founded Dependency Relations

Definition 10 For $Z \subseteq \mathbb{Z}^d$ and $\mathbf{p}, \mathbf{q} \in \mathbb{N}^d$, let

$$\mathbf{p} \prec_Z \mathbf{q} \qquad \text{if} \qquad \mathbf{p} - \mathbf{q} \in Z \text{ and } \mathbf{q} + Z \subseteq \mathbb{N}^d . \tag{3.3}$$

The transitive closure $\overset{*}{\prec}_Z$ of $\prec_Z$ in $\mathbb{N}^d$ is the *dependency relation* corresponding to Z. We shall say that $\mathbf{q}$ *depends on* $\mathbf{p}$ when $\mathbf{p} \overset{*}{\prec}_Z \mathbf{q}$. Also, we shall write $\mathbf{p} \overset{*}{\preceq}_Z \mathbf{q}$ when $\mathbf{p} \overset{*}{\prec}_Z \mathbf{q}$ or $\mathbf{p} = \mathbf{q}$. □

Proposition 1 *Let* $\mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s} \in \mathbb{N}^d$. *Then*

$$\mathbf{p} \overset{*}{\prec}_Z \mathbf{q},\ \mathbf{r} \overset{*}{\preceq}_Z \mathbf{s} \quad \textit{implies} \quad \mathbf{p} + \mathbf{r} \overset{*}{\prec}_Z \mathbf{q} + \mathbf{s} . \tag{3.4}$$

Proof: Obviously $\prec_Z$ is translation-invariant: if $\mathbf{p}, \mathbf{q}, \mathbf{r} \in \mathbb{N}^d$ and $\mathbf{p} \prec_Z \mathbf{q}$ then $\mathbf{p} + \mathbf{r} \prec_Z \mathbf{q} + \mathbf{r}$. Therefore $\overset{*}{\prec}_Z$ is translation-invariant as well. Then $\mathbf{p} \overset{*}{\prec}_Z \mathbf{q}$ implies $\mathbf{p} + \mathbf{r} \overset{*}{\prec}_Z \mathbf{q} + \mathbf{r}$. If $\mathbf{r} \overset{*}{\prec}_Z \mathbf{s}$ then $\mathbf{q} + \mathbf{r} \overset{*}{\prec}_Z \mathbf{q} + \mathbf{s}$, and the assertion follows by transitivity. □

Theorem 5 *Let $Z \subseteq \mathbb{Z}^d$ be a finite set, and $\overset{*}{\prec}_Z$ the corresponding dependency relation. The following assertions are equivalent:*

(i) *$\overset{*}{\prec}_Z$ is asymmetric,*

(ii) $\mathbf{0} \notin \operatorname{icon} Z$,

(iii) $\mathbf{0} \notin \operatorname{conv} Z$,

(iv) *there exists an $\mathbf{a} \in \mathbb{R}^d$ such that $\mathbf{a} \cdot \mathbf{z} < 0$ for all $\mathbf{z} \in Z$.*

Proof: If Z is empty then all four assertions are trivially true, and thus equivalent. Now assume that Z is nonempty.

(i) $\Rightarrow$ (ii) If $\mathbf{0} \in \operatorname{icon} Z$ then there exist positive integers $k_0, k_1, \ldots, k_d$ and $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_d \in Z$ such that

$$\sum_{i=0}^{d} k_i \mathbf{z}_i = \mathbf{0}. \tag{3.5}$$

Let $M := \sum_{i=0}^{d} k_i$. There exists some $\mathbf{p} \in \mathbb{N}^d$ such that $\mathbf{p} + \mathbf{z}_i \in \mathbb{N}^d$, for $i = 0, 1, \ldots, d$. Then $\mathbf{p} + \mathbf{z}_i \prec_Z \mathbf{p}$, for $i = 0, 1, \ldots, d$. Hence by repeated application of Proposition 1, $k_i \mathbf{p} + k_i \mathbf{z}_i \overset{*}{\prec}_Z k_i \mathbf{p}$ for $i = 0, 1, \ldots, d$, and $M\mathbf{p} + \sum_{i=0}^{d} k_i \mathbf{z}_i \overset{*}{\prec}_Z M\mathbf{p}$. By (3.5), $M\mathbf{p} \overset{*}{\prec}_Z M\mathbf{p}$, so $\overset{*}{\prec}_Z$ is not irreflexive, and hence not asymmetric.

(ii) $\Rightarrow$ (iii) Assume that $\mathbf{0} \in \operatorname{conv} Z$. By Lemma 2 there is a rational convex combination of the points in Z which is equal to $\mathbf{0}$. Multiplying this combination by the least common denominator of its coefficients we see that $\mathbf{0}$ can be expressed as a positive integral combination of the points in Z.

(iii) $\Rightarrow$ (iv) If $\mathbf{0} \notin \operatorname{conv} Z$ then $\{\mathbf{0}\}$ and $\operatorname{conv} Z$ are disjoint convex polyhedra. According to Theorem 3, they can be separated by a hyperplane which meets neither of them. Therefore there exists an $\mathbf{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $\mathbf{a} \cdot \mathbf{z} < b < 0$ for all $\mathbf{z} \in Z$.

(iv) $\Rightarrow$ (i) Assume that $\mathbf{a} \in \mathbb{R}^d$ is such that $\mathbf{a} \cdot \mathbf{z} < 0$ for all $\mathbf{z} \in Z$. Let $\mathbf{p} \prec_Z \mathbf{q}$. Then $\mathbf{p} - \mathbf{q} \in Z$, so $\mathbf{a} \cdot (\mathbf{p} - \mathbf{q}) < 0$ and $\mathbf{a} \cdot \mathbf{p} < \mathbf{a} \cdot \mathbf{q}$. Applying this successively we see that $\mathbf{p} \overset{*}{\prec}_Z \mathbf{q}$ implies $\mathbf{a} \cdot \mathbf{p} < \mathbf{a} \cdot \mathbf{q}$ as well. Since $<$ is asymmetric in $\mathbb{R}$ it follows that $\overset{*}{\prec}_Z$ is asymmetric in $\mathbb{N}^d$. $\square$

Theorem 6 *Let $Z \subseteq \mathbb{Z}^d$ be a finite set, and $\overset{*}{\prec}_Z$ the corresponding dependency relation. The following assertions are equivalent:*

(i) *$\overset{*}{\prec}_Z$ has no infinite descending chain,*

(ii) $\operatorname{icon} Z \cap \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \geq \mathbf{0}, \mathbf{x} \neq \mathbf{0}\} = \emptyset$,

(iii) $\operatorname{conv} Z \cap \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \geq \mathbf{0}, \mathbf{x} \neq \mathbf{0}\} = \emptyset$,

(iv) *there exists an* $\mathbf{a} \in \mathbb{R}^d$, $\mathbf{a} > \mathbf{0}$, *such that* $\mathbf{a} \cdot \mathbf{z} \leq 0$ *for all* $\mathbf{z} \in Z$.

Proof: If Z is empty then all four assertions are trivially true, and thus equivalent. Now assume that Z is nonempty.

(i) $\Rightarrow$ (ii) Let $\mathbf{z}$ be a non-zero point in icon Z such that $\mathbf{z} \geq \mathbf{0}$. Then $\mathbf{z} = \sum_{i=0}^{d} k_i \mathbf{z}_i$ for some $d \in \mathbb{N}$, $k_i \in \mathbb{N}^+$, and $\mathbf{z}_i \in Z$. There exists some $\mathbf{p} \in \mathbb{N}^d$ such that $\mathbf{p} + \mathbf{z}_i \in \mathbb{N}^d$, for $i = 0, 1, \ldots, d$. Then $\mathbf{p} + \mathbf{z}_i \prec_Z \mathbf{p}$, for $i = 0, 1, \ldots, d$. Let $M := \sum_{i=0}^{d} k_i$. By repeated application of Proposition 1, $k_i \mathbf{p} + k_i \mathbf{z}_i \overset{*}{\prec}_Z k_i \mathbf{p}$ for $i = 0, 1, \ldots, d$, and $M\mathbf{p} + \mathbf{z} \overset{*}{\prec}_Z M\mathbf{p}$. Let $\mathbf{z}_k := M\mathbf{p} + k\mathbf{z}$. Since $\mathbf{z} \in \mathbb{N}^d$ it follows by repeated application of Proposition 1 and by induction on k that $\mathbf{z}_{k+1} \overset{*}{\prec}_Z \mathbf{z}_k$ for $k \geq 0$. Since $\mathbf{z} \neq \mathbf{0}$, all the $\mathbf{z}_k$ are distinct and form an infinite descending chain in $\mathbb{N}^d$.

(ii) $\Rightarrow$ (iii) Let $K := (\operatorname{conv} Z) \cap \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \geq \mathbf{0}\}$. As K is a convex polytope with rational vertices, it is a rational convex set. By Lemma 3, rational points are dense in K. By assumption, K contains a non-zero point. Therefore it contains a non-zero rational point $\mathbf{q}$. By Lemma 2, $\mathbf{q}$ is a rational convex combination of points from Z. Let M be the least common denominator of the coefficients in this combination. Then $\mathbf{z} = M\mathbf{q}$ belongs to icon Z, is non-zero, and $\mathbf{z} \geq \mathbf{0}$.

(iii) $\Rightarrow$ (iv) Let $Q = \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \geq \mathbf{0}\}$ be the nonnegative orthant. Assume that $(\operatorname{conv} Z) \cap Q \subseteq \{\mathbf{0}\}$. Let C be the convex cone generated by Z, and C^p its polar cone. We claim that the relative interiors of C^p and Q have a common point.

If not, then by Theorem 4, C^p and Q can be separated by some hyperplane H. That is, there is an $\mathbf{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $\mathbf{a} \neq 0$, $\mathbf{a} \cdot \mathbf{x} \leq b$ for all $\mathbf{x} \in C^p$, and $\mathbf{a} \cdot \mathbf{x} \geq b$ for all $\mathbf{x} \in Q$. Since $\mathbf{0}$ belongs to both C^p and Q, $b = 0$. Since $\mathbf{e}_i \in Q$ for $i = 1, 2, \ldots, d$, $a_i = \mathbf{a} \cdot \mathbf{e}_i \geq 0$ for $i = 1, 2, \ldots, d$, and so $\mathbf{a} \in Q$. Let $D = \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{a} \cdot \mathbf{x} \leq 0\}$ be the halfspace defined by H and containing C^p. Then $D^p \subseteq C^{pp} = C$, by Lemma 1. Here we used the fact that Z is finite. By definition of the polar cone, $\mathbf{a} \in D^p$, and so $\mathbf{a} \in C$. Therefore $\mathbf{a} = \sum_{i=0}^{d} \lambda_i \mathbf{z}_i$ for some $d \in \mathbb{N}$, $\lambda_i \geq 0$, and $\mathbf{z}_i \in Z$. As $\mathbf{a} \neq 0$, $\sum_{i=0}^{d} \lambda_i > 0$. Let $\mathbf{a}' = \mathbf{a} / \sum_{i=0}^{d} \lambda_i$. Then $\mathbf{a}' \neq 0$ and $\mathbf{a}' \in (\operatorname{conv} Z) \cap Q$, contrary to the assumption. This proves the claim.

Therefore there exists an $\mathbf{a} \in \mathbb{R}^d$ which is an inner point of both C^p and Q. Hence $\mathbf{a} > \mathbf{0}$ and $\mathbf{a} \in C^p$. By definition of the polar cone, this implies that $\mathbf{a} \cdot \mathbf{x} \leq 0$ for all $\mathbf{x} \in C$, and so $\mathbf{a} \cdot \mathbf{z} \leq 0$ for all $\mathbf{z} \in Z$.

(iv) $\Rightarrow$ (i) Assume that $\mathbf{a} \in \mathbb{R}^d$ is such that $\mathbf{a} > \mathbf{0}$, and $\mathbf{a} \cdot \mathbf{z} \leq 0$ for all $\mathbf{z} \in Z$. We claim that for any $\mathbf{p} \in \mathbb{N}^d$ there are only finitely many $\mathbf{q} \in \mathbb{N}^d$ such that $\mathbf{q} \overset{*}{\prec}_Z \mathbf{p}$. From this it follows immediately that $\overset{*}{\prec}_Z$ has no infinite descending chain in $\mathbb{N}^d$.

To prove the claim, pick $\mathbf{p} \in \mathbb{N}^d$ and let $c := \mathbf{a} \cdot \mathbf{p}$. If $\mathbf{q} \prec_Z \mathbf{p}$ then $\mathbf{q} - \mathbf{p} \in Z$, so $\mathbf{a} \cdot (\mathbf{q} - \mathbf{p}) \leq 0$ and $\mathbf{a} \cdot \mathbf{q} \leq c$. Applying this successively we see that $\mathbf{q} \overset{*}{\prec}_Z \mathbf{p}$ implies $\mathbf{a} \cdot \mathbf{q} \leq c$ as well. As $q_i \geq 0$ and $a_i > 0$ it follows that $0 \leq q_i \leq c/a_i$, for $i = 1, 2, \ldots, d$. Hence there are only finitely many $\mathbf{q} \in \mathbb{N}^d$ such that $\mathbf{q} \overset{*}{\prec}_Z \mathbf{p}$ as claimed. □

Remark The implication (ii) $\Rightarrow$ (i) can be proved directly by using Dickson's Lemma [Dic13] which says: Given a sequence $(\mathbf{p}_k)_{k=0}^{\infty}$ in $\mathbb{N}^d$, there exists an integer $K \geq 0$ such that for every $k > K$, there is a $j \leq K$ with $\mathbf{p}_j \leq \mathbf{p}_k$.

Assume that (i) is false. Then there are points $\mathbf{p}_0, \mathbf{p}_1, \ldots$ such that $\mathbf{p}_0 \overset{*}{\succ}_Z \mathbf{p}_1 \overset{*}{\succ}_Z \ldots$ and

$\mathbf{p}_j \neq \mathbf{p}_k$ when $j \neq k$. By Dickson's Lemma, there exist j and k such that $j < k$ and $\mathbf{p}_j \leq \mathbf{p}_k$. Then $\mathbf{p}_k - \mathbf{p}_j$ belongs to $\{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \geq \mathbf{0}, \mathbf{x} \neq \mathbf{0}\}$. Also, there are $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m$ such that $\mathbf{p}_j = \mathbf{q}_0 \succ_Z \mathbf{q}_1 \succ_Z \ldots \succ_Z \mathbf{q}_m = \mathbf{p}_k$. It follows that $\mathbf{q}_{i+1} - \mathbf{q}_i \in Z$, for $i = 0, 1, \ldots, m-1$. Then $\mathbf{p}_k - \mathbf{p}_j = \mathbf{q}_m - \mathbf{q}_0 = \sum_{i=0}^{m-1}(\mathbf{q}_{i+1} - \mathbf{q}_i) \in \operatorname{icon} Z$, in violation of (ii).

Corollary 2 *Let $Z \subseteq \mathbb{Z}^d$ be a finite set, and $\overset{*}{\prec}_Z$ the corresponding dependency relation. The following assertions are equivalent:*

(i) *$\overset{*}{\prec}_Z$ is well-founded in $\mathbb{N}^d$,*

(ii) $\operatorname{icon} Z \cap \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \geq \mathbf{0}\} = \emptyset$,

(iii) $\operatorname{conv} Z \cap \{\mathbf{x} \in \mathbb{R}^d;\ \mathbf{x} \geq \mathbf{0}\} = \emptyset$,

(iv) *there exists an $\mathbf{a} \in \mathbb{R}^d$, $\mathbf{a} > \mathbf{0}$, such that $\mathbf{a} \cdot \mathbf{z} < 0$ for all $\mathbf{z} \in Z$,*

(v) *there exists an $\mathbf{a} \in \mathbb{N}^d$, $\mathbf{a} > \mathbf{0}$, such that $\mathbf{a} \cdot \mathbf{z} < 0$ for all $\mathbf{z} \in Z$,*

(vi) *$\overset{*}{\prec}_Z$ can be embedded into a linear ordering of $\mathbb{N}^d$ of order type ω.*

Proof: Each of the assertions (i) – (iv) can be seen to be equivalent to the conjunction of the corresponding assertions in Theorems 5 and 6, therefore they are equivalent among themselves.

(iv) $\Rightarrow$ (v) Let $\mathbf{a} \in \mathbb{R}^d$, $\mathbf{a} > \mathbf{0}$, be such that $\mathbf{a} \cdot \mathbf{z} < 0$ for all $\mathbf{z} \in Z$. Pick $\varepsilon > 0$ so that $\varepsilon < \min_{1 \leq i \leq d} a_i$ and $\varepsilon < \min_{\mathbf{z} \in Z}(-\mathbf{a} \cdot \mathbf{z} / \sum_{i=1}^d |z_i|)$. This is possible because $\mathbf{0} \notin Z$ and Z is finite. Let $\mathbf{b} \in \mathbb{Q}^d$ be such that $|a_i - b_i| < \varepsilon$ for $i = 1, 2, \ldots, d$. Then

$$b_i = a_i - (a_i - b_i) \geq a_i - |a_i - b_i| > a_i - \varepsilon > 0$$

and

$$\begin{aligned} \mathbf{b} \cdot \mathbf{z} &= \sum_{i=1}^d b_i z_i = \sum_{i=1}^d [a_i - (a_i - b_i)] z_i = \mathbf{a} \cdot \mathbf{z} - \sum_{i=1}^d (a_i - b_i) z_i \\ &\leq \mathbf{a} \cdot \mathbf{z} + \sum_{i=1}^d |a_i - b_i||z_i| < \mathbf{a} \cdot \mathbf{z} + \varepsilon \sum_{i=1}^d |z_i| < \mathbf{a} \cdot \mathbf{z} - \mathbf{a} \cdot \mathbf{z} = 0 . \end{aligned}$$

Let K be a positive common denominator for the b_i, and $\mathbf{c} := K\mathbf{b}$. Then $\mathbf{c} \in \mathbb{N}^d$, $\mathbf{c} > 0$, and $\mathbf{c} \cdot \mathbf{z} < 0$ for all $\mathbf{z} \in Z$.

(v) $\Rightarrow$ (vi) Let $\mathbf{a} \in \mathbb{N}^d$, $\mathbf{a} > \mathbf{0}$, be such that $\mathbf{a} \cdot \mathbf{z} < 0$ for all $\mathbf{z} \in Z$. Then $\mathbf{a} \cdot \mathbf{p} \in \mathbb{N}$ for all $\mathbf{p} \in \mathbb{N}^d$. The equation $\mathbf{a} \cdot \mathbf{x} = k$ has only finitely many solutions $\mathbf{x} \in \mathbb{N}^d$ for any $k \in \mathbb{N}$, because it implies that $0 \leq x_i \leq k/a_i$ for all i.

Let L be any linear ordering of $\mathbb{N}^d$ with the property that $\mathbf{a} \cdot \mathbf{p} < \mathbf{a} \cdot \mathbf{q}$ implies $\mathbf{p} \, L \, \mathbf{q}$. Then by the preceding paragraph L is of order type ω, and since $\mathbf{p} \overset{*}{\prec}_Z \mathbf{q}$ implies $\mathbf{a} \cdot \mathbf{p} < \mathbf{a} \cdot \mathbf{q}$, dependency relation $\overset{*}{\prec}_Z$ can be embedded into L.

(vi) $\Rightarrow$ (i) Any ordering of type ω is a well-ordering, therefore any transitive relation embeddable into it is well-founded. □

3.4 Existence and Uniqueness of Solutions

Theorem 7 *Let $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\} \subseteq \mathbb{Z}^d$ satisfy any of the equivalent conditions of Theorem 2. Then there exists a unique sequence $a : \mathbb{N}^d \to A$ which satisfies* (3.1) *and* (3.2).

Proof: 1. EXISTENCE PART. Let $\mathbf{p}_1 L \mathbf{p}_2 L \ldots$ be a linear ordering of $\mathbb{N}^d$ of order type ω for which $\mathbf{p} \overset{*}{\prec}_Z \mathbf{q}$ implies $\mathbf{p} L \mathbf{q}$. Such an ordering exists by Theorem 2 (vi). Define a sequence of partial functions $\varphi_i : \mathbb{N}^d \to A$ for $i \geq 0$ by setting

$$\varphi_0 := \perp,$$

and inductively for $i \geq 1$ by

$$\begin{aligned} \varphi_i(\mathbf{p}_j) &:= \varphi_{i-1}(\mathbf{p}_j), \quad \text{for } j < i \\ \varphi_i(\mathbf{p}_i) &:= \begin{cases} f(\mathbf{p}_i), & \text{if } \mathbf{p}_i \in I \\ F(\varphi_{i-1}(\mathbf{p}_i + \mathbf{z}_1), \varphi_{i-1}(\mathbf{p}_i + \mathbf{z}_2), \ldots, \varphi_{i-1}(\mathbf{p}_i + \mathbf{z}_k)), & \text{otherwise} \end{cases} \\ \varphi_i(\mathbf{p}_j) &:= \perp, \quad \text{for } j > i. \end{aligned}$$

We claim that φ_i is defined on $\mathbf{p}_j$ with $j \leq i$, and undefined elsewhere. We prove this by induction on i.

The claim is certainly true for $i = 0$. Now assume that $i \geq 1$, and that the claim holds for $i - 1$. The only thing to check is whether φ_{i-1} is defined on $\mathbf{p}_i + \mathbf{z}_1, \mathbf{p}_i + \mathbf{z}_2, \ldots, \mathbf{p}_i + \mathbf{z}_k$ when $\mathbf{p}_i \notin I$. Then $\mathbf{p}_i \geq \mathbf{s}$, hence $\mathbf{p}_i + \mathbf{z}_j \geq \mathbf{s} + \mathbf{z}_j \in \mathbb{N}^d$ for $j = 1, 2, \ldots, k$. This implies that $\mathbf{p}_i + \mathbf{z}_j \prec_Z \mathbf{p}_i$, which in turn implies that $\mathbf{p}_i + \mathbf{z}_j L \mathbf{p}_i$, for $j = 1, 2, \ldots, k$. By induction hypothesis, it follows that φ_{i-1} is indeed defined on these points, proving the claim.

Since by definition, φ_i agrees with φ_{i-1} wherever the latter is defined, $a := \cup_{i=0}^{\infty} \varphi_i$ is a total function mapping $\mathbb{N}^d$ into A. It is clear from the definition of φ_i that a satisfies (3.1) and (3.2).

2. UNIQUENESS PART. Assume that a and b are two sequences satisfying (3.1) and (3.2). We will use structural induction on well-founded sets (cf. [Man74], Theorem 5-10) to prove that $a = b$.

By Theorem 2 (i), $\overset{*}{\prec}_Z$ is well-founded on $\mathbb{N}^d$. Let $S := \{\mathbf{p} \in \mathbb{N}^d; a_{\mathbf{p}} = b_{\mathbf{p}}\}$. We will have shown that $S = \mathbb{N}^d$ if we show that $\mathbf{p} \in S$, for all $\mathbf{p} \overset{*}{\prec}_Z \mathbf{q}$, implies $\mathbf{q} \in S$.

Assume that $\mathbf{p} \in S$, for all $\mathbf{p} \overset{*}{\prec}_Z \mathbf{q}$. We distinguish two cases. If $\mathbf{q} \in I$ then $a_{\mathbf{q}} = b_{\mathbf{q}} = f(\mathbf{q})$, hence $\mathbf{q} \in S$. If $\mathbf{q} \notin I$ then $a_{\mathbf{q}} = F(a_{\mathbf{q}+\mathbf{z}_1}, a_{\mathbf{q}+\mathbf{z}_2}, \ldots, a_{\mathbf{q}+\mathbf{z}_k})$ and $b_{\mathbf{q}} = F(b_{\mathbf{q}+\mathbf{z}_1}, b_{\mathbf{q}+\mathbf{z}_2}, \ldots, b_{\mathbf{q}+\mathbf{z}_k})$, by (3.1). But as $\mathbf{q} \notin I$, we have $\mathbf{q} + \mathbf{z}_j \prec_Z \mathbf{q}$, for $j = 1, 2, \ldots, k$. Then by the assumption, $\mathbf{q} + \mathbf{z}_j \in S$ and $a_{\mathbf{q}+\mathbf{z}_j} = b_{\mathbf{q}+\mathbf{z}_j}$, for $j = 1, 2, \ldots, k$. It follows that $a_{\mathbf{q}} = b_{\mathbf{q}}$ and so $\mathbf{q} \in S$.

This shows that $S = \mathbb{N}^d$, and hence that $a = b$. □

3.5 Equations with Constant Coefficients

First we restate the existence and uniqueness theorem for the case of constant coefficients, and show that the generating function of the solution is analytic at the origin, provided that the initial conditions do not grow faster than exponentially.

Theorem 8 *Let $Z \subseteq \mathbb{Z}^d$ be a nonempty finite set which satisfies the conditions of Theorem 2. Let $\mathbf{s} \in \mathbb{N}^d$ be such that $\mathbf{s} + \mathbf{z} \in \mathbb{N}^d$ for all $\mathbf{z} \in Z$. Let $I := \{\mathbf{p} \in \mathbb{N}^d;\ \mathbf{p} \not\geq \mathbf{s}\}$. Let $c_{\mathbf{z}} \in \mathbb{C}$, $c_{\mathbf{z}} \neq 0$ for $\mathbf{z} \in Z$ be given constants, and $f : I \to \mathbb{C}$ a given function. Then there exists a unique sequence $a : \mathbb{N}^d \to \mathbb{C}$ which satisfies*

$$a_{\mathbf{p}} = \sum_{\mathbf{z}\in Z} c_{\mathbf{z}} a_{\mathbf{p}+\mathbf{z}}, \quad \text{for } \mathbf{p} \geq \mathbf{s}, \tag{3.6}$$

and

$$a_{\mathbf{p}} = f(\mathbf{p}), \quad \text{for } \mathbf{p} \in I. \tag{3.7}$$

Furthermore, if

$$|f(\mathbf{p})| \leq k^{\mathbf{u}\cdot\mathbf{p}},$$

for some $k > 0$, $\mathbf{u} \in \mathbb{R}^d$, and for all $\mathbf{p} \in I$, then the generating function

$$G(x_1, x_2, \ldots, x_d) := \sum_{\mathbf{p}\in\mathbb{N}^d} a_{\mathbf{p}} x_1^{p_1} x_2^{p_2} \cdots x_d^{p_d}$$

is analytic in a neighborhood of the origin.

Proof: Existence and uniqueness follows immediately from Theorem 7. Analyticity will follow if we show that there is a $K > 0$ and a vector $\mathbf{v} \in \mathbb{R}^d$ such that $|a_{\mathbf{p}}| \leq K^{\mathbf{v}\cdot\mathbf{p}}$ for all $\mathbf{p} \in \mathbb{N}^d$.

By Theorem 2 (iv), there is a $\mathbf{v} \in \mathbb{R}^d$ such that $\mathbf{v} > \mathbf{0}$, and $\mathbf{v} \cdot \mathbf{z} < 0$ for all $\mathbf{z} \in Z$. Since Z is finite there exists an $\varepsilon > 0$ such that $\mathbf{v} \cdot \mathbf{z} \leq -\varepsilon$ for all $\mathbf{z} \in Z$. Let

$$K := \max\{1, \left(\sum_{\mathbf{z}\in Z} |c_{\mathbf{z}}|\right)^{1/\varepsilon}, \max_{1\leq i\leq d} k^{u_i/v_i}\}.$$

We now prove by structural induction on well-founded sets that

$$|a_{\mathbf{p}}| \leq K^{\mathbf{v}\cdot\mathbf{p}}$$

for all $\mathbf{p} \in \mathbb{N}^d$.

If $\mathbf{p} \in I$ then

$$\begin{aligned} |a_{\mathbf{p}}| &= |f(\mathbf{p})| \leq k^{\mathbf{u}\cdot\mathbf{p}} = k^{u_1p_1} k^{u_2p_2} \cdots k^{u_dp_d} \\ &= \left(k^{\frac{u_1}{v_1}}\right)^{v_1p_1} \left(k^{\frac{u_2}{v_2}}\right)^{v_2p_2} \cdots \left(k^{\frac{u_d}{v_d}}\right)^{v_dp_d} \leq K^{\mathbf{v}\cdot\mathbf{p}}. \end{aligned}$$

Otherwise we can assume that $|a_{\mathbf{p}+\mathbf{z}}| \leq K^{\mathbf{v}\cdot(\mathbf{p}+\mathbf{z})}$ for all $\mathbf{z} \in Z$. Then

$$\begin{aligned} \left|\sum_{\mathbf{z}\in Z} c_{\mathbf{z}} a_{\mathbf{p}+\mathbf{z}}\right| &\leq \sum_{\mathbf{z}\in Z} |c_{\mathbf{z}}| K^{\mathbf{v}\cdot(\mathbf{p}+\mathbf{z})} \leq \sum_{\mathbf{z}\in Z} |c_{\mathbf{z}}| K^{\mathbf{v}\cdot\mathbf{p}-\varepsilon} \\ &= K^{\mathbf{v}\cdot\mathbf{p}-\varepsilon} \sum_{\mathbf{z}\in Z} |c_{\mathbf{z}}| < K^{\mathbf{v}\cdot\mathbf{p}}, \end{aligned}$$

proving the claim. □

In the case of constant coefficients we can select any point with a non-zero coefficient to be expressed explicitly from the recurrence. For example, if $c_0 \neq 0$ then we can rewrite

$$\sum_{i=0}^{k} c_i a_{\mathbf{p}+\mathbf{w}_i} = 0, \quad \text{for } \mathbf{p} \geq \mathbf{s} \tag{3.8}$$

as

$$a_{\mathbf{p}} = \sum_{i=1}^{k} \left(-\frac{c_i}{c_0}\right) a_{\mathbf{p}+\mathbf{z}_i}, \quad \text{for } \mathbf{p} \geq \mathbf{s} + \mathbf{w}_0, \tag{3.9}$$

where $\mathbf{z}_i = \mathbf{w}_i - \mathbf{w}_0$, for $i = 1, 2, \ldots, k$. We now show that there is always at least one "good" point.

Theorem 9 *Let $W \subseteq \mathbb{Z}^d$ be a nonempty finite set. Then there exists a point $\mathbf{w}_0 \in W$ such that the set $Z := \{\mathbf{w} - \mathbf{w}_0; \mathbf{w} \in W, \mathbf{w} \neq \mathbf{w}_0\}$ satisfies the conditions of Theorem 2.*

Proof: Let $\mathbf{w}_0$ be the last point in W with respect to the lexicographic ordering of $\mathbb{Z}^d$. We claim that then the relation $\overset{*}{\prec}_Z$ is well-founded. To prove this claim we need to show that $\overset{*}{\prec}_Z$ is asymmetric and has no infinite descending chains.

Assume that $\overset{*}{\prec}_Z$ is not asymmetric. Then, because it is transitive, there is a $\mathbf{p} \in \mathbb{N}^d$ such that $\mathbf{p} \overset{*}{\prec}_Z \mathbf{p}$. Therefore $\prec_Z$ has a chain of the form $\mathbf{p} = \mathbf{p}_0 \prec_Z \mathbf{p}_1 \prec_Z \ldots \prec_Z \mathbf{p}_k = \mathbf{p}$. We assert that all the points in this chain are equal to $\mathbf{p}$. This will imply that $\mathbf{p} \prec_Z \mathbf{p}$ which is impossible since by construction, Z does not contain $\mathbf{0}$.

We prove the assertion by showing that the points in the chain agree in their first k coordinates, for $k = 1, 2, \ldots, d$. We use induction on k.

For the base case, take $k = 0$, and for the induction step, take $k > 0$ in the following proof.

Assume that $1 \leq k \leq d$ and that the points in the chain have their first $k-1$ coordinates equal. By definition of $\prec_Z$, the points $\mathbf{z}_i := \mathbf{p}_i - \mathbf{p}_{i+1}$ belong to Z, for $i = 0, 1, \ldots, k-1$. Their first $k-1$ coordinates are equal to zero, so by maximality of $\mathbf{w}_0$, we have $z_{ik} \leq 0$, for $i = 0, 1, \ldots, k-1$. But then $p_k = p_{0k} \leq p_{1k} \leq \ldots \leq p_{kk} = p_k$, so all the k-th coordinates are equal. This completes the proof that $\overset{*}{\prec}_Z$ is asymmetric.

If $\overset{*}{\prec}_Z$ has an infinite descending chain then so does $\prec_Z$ because $\overset{*}{\prec}_Z$ is asymmetric. Let $\mathbf{p}_0 \succ_Z \mathbf{p}_1 \succ_Z \ldots$ be such a chain. We assert that there is an N such that from N on, the points in this chain do not change any more. This will imply that the chain is finite.

We prove the assertion by showing that for every k between 1 and d, there is an N_k such that from N_k on, the first k coordinates of the points in the chain do not change any more. We use induction on k.

For the base case, take $k = 0$, and for the induction step, take $k > 0$ in the following proof.

Assume that $1 \leq k \leq d$ and that there is an N_{k-1} such that from N_{k-1} on, the first $k-1$ coordinates of the points in the chain do not change any more. By definition of $\prec_Z$, the points $\mathbf{z}_i := \mathbf{p}_{i+1} - \mathbf{p}_i$ belong to Z, for $i = N_{k-1}, N_{k-1}+1, \ldots$. Their first $k-1$ coordinates are equal to zero, so by maximality of $\mathbf{w}_0$, we have $z_{ik} \leq 0$, for $i = N_{k-1}, N_{k-1}+1, \ldots$. But then $p_{N_{k-1},k} \geq p_{N_{k-1}+1,k} \leq \ldots$. Since these coordinates are nonnegative integers, there exists an $N_k \geq N_{k-1}$ such that from N_k on they do not change any more. This completes the proof that $\overset{*}{\prec}_Z$ has no infinite descending chain. □

In certain cases the generating function can actually be computed. In the next section we give an example of a system of partial difference equations with constant coefficients and two unknown sequences, where a method apparently first used by Knuth in Exercises 2.2.1.–4 and 2.2.1.–11 of [Knu68] reveals the generating function.

3.6 An Example

This example is taken from [LP90]. The objective is to find an explicit solution of the following system of partial difference equations:

$$\begin{aligned} a_{i,0} &= 4^i \quad (i \geq 0), \\ a_{i,1} &= 4^{i+1} \quad (i \geq 0), \\ a_{i,n} &= a_{i-1,n} + 2a_{i,n-1} + a_{i+1,n-2} \quad (i \geq 1, n \geq 2), \end{aligned} \tag{3.10}$$

$$\begin{aligned} b_{i,0} &= 1 \quad (i \geq 0), \\ b_{i,1} &= 2i+4 \quad (i \geq 0), \\ b_{i,n} &= b_{i-1,n} + 2b_{i,n-1} + b_{i+1,n-2} \quad (i \geq 1, n \geq 2), \end{aligned} \tag{3.11}$$

$$\begin{aligned} a_{0,n} &= 2a_{1,n-2} + 2a_{0,n-1} + b_{1,n-2} \quad (n \geq 2), \\ b_{0,n} &= a_{0,n} \quad (n \geq 0). \end{aligned} \tag{3.12}$$

We refer to [LP90], [Luk87], and Problem 44 in [Bir67] for explanations of the meaning of $b_{1,n}$, $b_{i,n}$, and $a_{i,n}$ in lattice theory.

Let

$$\begin{aligned} A(x,y) &= \sum_{i=0}^{\infty}\sum_{n=0}^{\infty} a_{i,n}x^i y^n, \\ B(x,y) &= \sum_{i=0}^{\infty}\sum_{n=0}^{\infty} b_{i,n}x^i y^n, \\ A_0(y) &= \sum_{n=0}^{\infty} a_{0,n}y^n = A(0,y), \\ A_1(y) &= \sum_{n=0}^{\infty} a_{1,n}y^n = \frac{\partial A(x,y)}{\partial x}\Big|_{x=0}, \\ B_1(y) &= \sum_{n=0}^{\infty} b_{1,n}y^n = \frac{\partial B(x,y)}{\partial x}\Big|_{x=0}. \end{aligned}$$

From (3.10) – (3.12) we can estimate by induction on $i+n$ that $|a_{i,n}|, |b_{i,n}| \leq 5^{i+n}$ for all $i, n \geq 0$. Therefore these series converge at least for $|x|, |y| < 1/5$.

In terms of generating functions, (3.10) – (3.12) translate into

$$(x-(x+y)^2)A(x,y) + (y^2+2xy-x)A_0(y) + xy^2A_1(y) = \frac{x^2(4y+3)}{1-4x}, \tag{3.13}$$

$$(x-(x+y)^2)B(x,y) + (y^2+2xy-x)A_0(y) + xy^2B_1(y) = 0, \tag{3.14}$$

$$(1-2y)A_0(y) - 2y^2A_1(y) - y^2B_1(y) = 1+2y. \tag{3.15}$$

We have three equations for five unknown functions, hence we need two more. Both $A(x,y)$ and $B(x,y)$ are analytic in a neighborhood $\mathcal{N}$ of the origin. Therefore $(x-(x+y)^2)A(x,y) = (x-(x+y)^2)B(x,y) = 0$ on that segment of parabola $x-(x+y)^2 = 0$ which lies inside $\mathcal{N}$, that is, when $|y|$ is sufficiently small and $x = (1-\sqrt{1-4y})/2 - y$. Substituting this for x in (3.13) and (3.14) gives the two additional equations

$$\begin{aligned}(y - \frac{1-\sqrt{1-4y}}{2})A_0(y) + y^2A_1(y) &= \frac{\frac{1}{\sqrt{1-4y}} - (1+2y)}{2}, \quad (3.16)\\ (y - \frac{1-\sqrt{1-4y}}{2})A_0(y) + y^2B_1(y) &= 0. \quad (3.17)\end{aligned}$$

Now we can solve the linear equations (3.15), (3.16), and (3.17) to obtain

$$\begin{aligned}A_0(y) &= (\frac{2y-1}{\sqrt{1-4y}} - 3)/(2(y^2+8y-2)),\\ A_1(y) &= -(\frac{y^2-4y+1}{\sqrt{1-4y}} + 2y^3 + 17y^2 + 2y - 1)/(2y^2(y^2+8y-2)),\\ B_1(y) &= (2y - 1 - \frac{2y^2+4y-1}{\sqrt{1-4y}})/(2y^2(y^2+8y-2)). \quad (3.18)\end{aligned}$$

After substituting these back into (3.13) and (3.14), we introduce a new variable $z = \sqrt{1-4y}$, cancel out the seeming singularity at the origin, and find that

$$\begin{aligned}B(x,y) &= \frac{4(z+1)^2}{z(z^2-6z+1)(4x-(z+1)^2)}, \quad (3.19)\\ A(x,y) - B(x,y) &= \frac{4x(z+2)}{(4x-1)z(4x-(z+1)^2)}. \quad (3.20)\end{aligned}$$

By expanding (3.18) into a power series, we determine that

$$b_{1,n} = \frac{1}{6\sqrt{2}}\sum_{k=0}^{n}\left[2\binom{2k}{k} - \binom{2k}{k-2}\right](\lambda_1^{n-k+1} - \lambda_2^{n-k+1}) \qquad (n \geq 0), \quad (3.21)$$

where λ_1 and λ_2 are the roots of the polynomial $2\lambda^2 - 8\lambda - 1$,

$$\begin{aligned}\lambda_1 &= \frac{4+3\sqrt{2}}{2} \approx 4.12132, \quad (3.22)\\ \lambda_2 &= \frac{4-3\sqrt{2}}{2} \approx -0.12132. \quad (3.23)\end{aligned}$$

From (3.18) we can obtain several recurrence relations for $b_{1,n}$. For example,

$$2b_{1,n} - 8b_{1,n-1} - b_{1,n-2} = 2\binom{2n}{n} - \binom{2n}{n-2} \qquad (n \geq 2),$$

with initial conditions $b_{1,0} = 1$ and $b_{1,1} = 6$. One can also show that, asymptotically,

$$b_{1,n} = \frac{1}{4}\lambda_1^{n+2} - \frac{4^{n+2}}{\sqrt{\pi n}}(1 + O(n^{-1})). \quad (3.24)$$

From (3.16) and (3.17) it follows that

$$A_1(y) - B_1(y) = \frac{(1-4y)^{-1/2} - (1+2y)}{2y^2} = \frac{1}{2}\sum_{n=2}^{\infty}\binom{2n}{n} y^{n-2},$$

therefore

$$a_{1,n} = b_{1,n} + \frac{1}{2}\binom{2n+4}{n+2} \qquad (n \geq 0).$$

To compute $b_{i,n}$, we let

$$B_i(y) = \sum_{n=0}^{\infty} b_{i,n} y^n = \frac{1}{i!}\frac{\partial^i B(x,y)}{\partial x^i}\Big|_{x=0} \quad (i \geq 0).$$

Since x enters only once into the right-hand side of (3.19), it is easy to find that

$$B_i(y) = -\frac{4^{i+1}}{z(z+1)^{2i}(z^2-6z+1)} \qquad (z = \sqrt{1-4y},\ \ i \geq 0). \tag{3.25}$$

Let

$$d(y) = \frac{1}{\sqrt{1-4y}}, \qquad c(y) = \frac{1-\sqrt{1-4y}}{2y}$$

be the generating functions of the middle binomial coefficients, $d_n = \binom{2n}{n}$, and of the Catalan numbers, $c_n = d_n/(n+1)$, respectively. Then we can rewrite (3.25) as

$$B_i(y) = \frac{(2y-1)d(y) - 3}{2(y^2+8y-2)}\, c^{2i}(y) \quad (i \geq 0).$$

Expanding this function into a power series by means of the formulas in [Rio68], p. 154, we find that for $i \geq 0$ and $n \geq 0$,

$$b_{i,n} = \frac{1}{6\sqrt{2}}\sum_{k=0}^{n}\left[2\binom{2k+2i-2}{k} - \binom{2k+2i-2}{k-2}\right](\lambda_1^{n-k+1} - \lambda_2^{n-k+1}), \tag{3.26}$$

where λ_1 and λ_2 are defined in (3.22) and (3.23), respectively. For $i = 1$ this gives (3.21), while for $i = 0$ and $n \geq 0$ we have

$$a_{0,n} = b_{0,n} = \frac{1}{6\sqrt{2}}\left(\lambda_1^{n+1} - \lambda_2^{n+1} + \sum_{k=0}^{n}\binom{2k-2}{k}(\lambda_1^{n-k+1} - \lambda_2^{n-k+1})\right).$$

Applying the same approach to the difference $A(x,y) - E(x,y)$ in (3.20), one obtains

$$\begin{aligned} a_{i,n} &= b_{i,n} + \sum_{k=1}^{2i}\binom{2n+k}{n}2^{2i-k} \\ &= b_{i,n} + 4^{i+n} - \sum_{k=0}^{n}\binom{2n+2i+1}{k} \quad (i \geq 0,\ n \geq 0). \end{aligned} \tag{3.27}$$

It is straightforward to check that (3.26) and (3.27) satisfy Eqns. (3.10) and (3.11). Insertion of (3.26) and (3.27) into Eqn. (3.12) leads to

$$\frac{1}{6\sqrt{2}}\sum_{k=0}^{n}\left[4\binom{2k-4}{k}-5\binom{2k-4}{k-2}\right](\lambda_1^{n-k+1}-\lambda_2^{n-k+1}) = \frac{1}{2}(\lambda_1^n+\lambda_2^n)+\binom{2n}{n}, \tag{3.28}$$

where λ_1 and λ_2 are defined in (3.22) and (3.23), respectively, and $n \geq 2$. (As it turns out, (3.28) holds for $n \geq 0$.) To prove this identity, note that its left-hand side, being the convolution of $4\binom{2n-4}{n} - 5\binom{2n-4}{n-2}$ with the coefficients of the power series expansion of $1/(2-8z-z^2)$ around $z = 0$, solves the recurrence

$$2x_n - 8x_{n-1} - x_{n-2} = 4\binom{2n-4}{n} - 5\binom{2n-4}{n-2} \quad (n \geq 2). \tag{3.29}$$

Therefore it suffices to verify that (3.28) holds for $n = 0, 1$, and that $x_n = \binom{2n}{n}$ solves (3.29), too. — Alternatively, one can prove (3.28) by using Gosper's summation algorithm [Gos78] separately on the two terms involving λ_1 and λ_2.

Chapter 4

Difference equations with polynomial coefficients

A voda se odpre
in močna zvezda vzide
in nova ladja pride
in južni otok je.
— KAJETAN KOVIČ, *Južni otok (1976)*

4.1 Introduction

Following Cohn [Coh65] we shall call a field F *algorithmic* if it is computable (meaning that there exist algorithms for carrying out the field operations), and there is an algorithm for factoring polynomials in $F[x]$.

Let F be an algorithmic field of characteristic zero, and $p_0(x), p_1(x), \ldots, p_d(x)$ polynomials in $F[x]$ such that $p_0, p_d \not\equiv 0$. Then

$$\sum_{i=0}^{d} p_i(n)\, a_{n+i} = 0 \tag{4.1}$$

is a *homogeneous linear difference equation with polynomial coefficients* (an HLP, for short) for the unknown sequence (a_n) in F. The *order* of (4.1) is d, and the *degree* of (4.1) is $m = \max_{0 \le i \le d} \deg p_i(n)$. Following [Sta80] we identify two sequences if they agree from some point on. Hence we consider (a_n) as a solution of (4.1) if (4.1) holds for all large enough n. Similarly, by a non-zero sequence we mean a sequence with infinitely many non-zero terms.

We note that any equation with rational coefficients can be turned into an equivalent one with polynomial coefficients by multiplying it with a common denominator of its coefficients.

In this chapter we present an algorithm (called HYPER) for deciding existence of hypergeometric solutions of HLP's. To give some motivation, we describe first an application of algorithm HYPER to definite hypergeometric summation.

A non-zero sequence (a_n) is *hypergeometric* over F if it satisfies a first-order HLP, or equivalently, an equation of the form

$$a_{n+1} = r(n) a_n$$

where $r(x)$ is a rational function over F. The set H of all hypergeometric sequences over F forms a group under multiplication.

The problem of indefinite hypergeometric summation was solved by Gosper [Gos77], [Gos78] who discovered an algorithm for deciding existence of hypergeometric solutions of the nonhomogeneous first-order equation

discovered an algorithm for deciding existence of hypergeometric solutions of the nonhomogeneous first-order equation

um

$$a_{n+1} - a_n = h_n$$

of a hypergeometric sequence is hypergeometric (apart from an additive constant).

One can ask the same question about definite sums of the form

$$a_n = \sum_k F(n,k) \tag{4.2}$$

where summation ranges over all integers, and both $F(x+1,y)/F(x,y)$ and $F(x,y+1)/F(x,y)$ are rational functions of x and y. Using Bernstein's theory of holonomic functions, Zeilberger [Zeia] proved that every (a_n) of the form (4.2) satisfies an HLP. In [Zeib] he gave an algorithm which constructs such an equation. In [Wil91], Wilf gives an elementary proof of this important fact. His proof can also serve as an algorithm for finding the equation. Since algorithm HYPER can be used to obtain a maximal set of linearly independent hypergeometric solutions of an HLP, the combination of Zeilberger's (or Wilf's) algorithm with HYPER gives an algorithm for deciding whether (a_n) as defined in (4.2) is hypergeometric.

In 1974 Abramov [Abr74] developed an algorithm for finding rational solutions of nonhomogeneous equations with constant coefficients. In 1989 he gave algorithms for finding polynomial [Abr89b] and rational [Abr89a] solutions of nonhomogeneous linear difference equations with polynomial coefficients (NLP's). His algorithms work for homogeneous equations as well. With a simple modification, Abramov's algorithm for finding rational solutions can be used to find hypergeometric solutions of NLP's. This is described in Section 5. However, it is not clear how one could use Abramov's algorithm to find hypergeometric solutions of HLP's.

Although [Abr89b] contains an algorithm for finding polynomial solutions of NLP's and HLP's we devote Section 2 to derivation of algorithm POLY which finds a basis for the space of polynomial solutions of an HLP. There are three reasons for this: 1. to make the description of HYPER self-contained, 2. because the algorithm in [Abr89b] expects the equation to be given in terms of the difference operator whereas HYPER and POLY work with the shift operator, and 3. because POLY has been developed without knowledge of Abramov's work.

Section 3 describes a decomposition of rational functions which plays an important role both in Gosper's algorithm and in algorithm HYPER.

In Section 4 we develop algorithm HYPER which returns a hypergeometric solution of a given HLP if it exists, and prove its correctness. The algorithm works by constructing a finite set of auxiliary HLP's (of the same order as the original one) such that the original HLP has a hypergeometric solution if and only if an auxiliary HLP has a non-zero polynomial solution. Algorithm POLY is then used on each auxiliary HLP to determine if it has any non-zero polynomial solutions. A hypergeometric solution of the original HLP is easily obtained from any non-zero polynomial (or, for that matter, hypergeometric) solution of an auxiliary HLP. The auxiliary HLP's are generated in a loop which runs through all monic factors of the least and most significant coefficients of the given HLP, as well as through the roots of a polynomial whose degree does not exceed the order of the given HLP.

In Section 5 we show how to find a maximal set of linearly independent hypergeometric solutions of an HLP, and how to decide existence of hypergeometric solutions of NLP's. We conclude with some practical observations about the implementation of algorithm HYPER.

Throughout the chapter 0^0 is defined to be 1.

4.2 Polynomial solutions

In this section it suffices that F is a computable field of characteristic zero which has an algorithm for finding integer roots of polynomials in $F[x]$. Such a field is called "suitable" in [Abr89a].

The question of existence of non-zero polynomial solutions of an HLP reduces to the problem of finding an upper bound N for the possible degrees of such solutions. Once we have N, we can insert a polynomial of degree N with undetermined coefficients into the equation, and solve the resulting system of linear algebraic equations. It is not possible to bound the degree of polynomial solutions of (4.1) in terms of the degree of the equation alone. For example, the solutions of

$$na_{n+1} - (n+100)a_n = 0$$

are constant multiples of $n(n+1)\cdots(n+99)$ while the equation has linear coefficients. Similar examples can be constructed for higher-order equations.

We consider first the second-order equation

$$p(n)a_{n+2} + q(n)a_{n+1} + r(n)a_n = 0 \tag{4.3}$$

where p, q, r are polynomials. Assume that (a_n) is a non-zero polynomial solution of (4.3), of unknown degree N. Let m be the degree of (4.3) and

$$\begin{aligned} p(n) &= u_0n^m + u_1n^{m-1} + u_2n^{m-2} + O(n^{m-3}), \\ q(n) &= v_0n^m + v_1n^{m-1} + v_2n^{m-2} + O(n^{m-3}), \\ r(n) &= w_0n^m + w_1n^{m-1} + w_2n^{m-2} + O(n^{m-3}). \end{aligned}$$

By the definition of m, at least one of u_0, v_0, w_0 is not zero. Further, let

$$a_n = \alpha_0n^N + \alpha_1n^{N-1} + \alpha_2n^{N-2} + O(n^{N-3})$$

with $\alpha_0 \neq 0$. Then, by the Binomial Theorem,

$$\begin{aligned} a_{n+1} &= \alpha_0\left(n^N + Nn^{N-1} + \binom{N}{2}n^{N-2}\right) \\ &+ \alpha_1\left(n^{N-1} + (N-1)n^{N-2}\right) + \alpha_2n^{N-2} + O(n^{N-3}) \end{aligned}$$

and

$$\begin{aligned} a_{n+2} &= \alpha_0\left(n^N + 2Nn^{N-1} + 4\binom{N}{2}n^{N-2}\right) \\ &+ \alpha_1\left(n^{N-1} + 2(N-1)n^{N-2}\right) + \alpha_2n^{N-2} + O(n^{N-3}). \end{aligned}$$

We plug all these expansions into (4.3) and equate coefficients of like powers of n. The coefficient of n^{N+m} yields, after canceling α_0,

$$u_0 + v_0 + w_0 = 0. \tag{4.4}$$

If this condition is not fulfilled then (4.3) has no non-zero polynomial solution. Otherwise we go on to the next lower power n^{N+m-1}, use (4.4), cancel α_0, and find that

$$(2u_0 + v_0)N + u_1 + v_1 + w_1 = 0 . \tag{4.5}$$

If $2u_0 + v_0 \neq 0$ then (4.5) determines a single possible value for N. If

$$2u_0 + v_0 = 0 \tag{4.6}$$

then we must also have

$$u_1 + v_1 + w_1 = 0 \tag{4.7}$$

or else (4.3) has no non-zero polynomial solution. If both (4.6) and (4.7) are true we continue with the coefficient of n^{N+m-2}. Here we find after using (4.4), (4.6), and (4.7), and after canceling α_0, that

$$u_0N^2 + (2u_1 - u_0 + v_1)N + u_2 + v_2 + w_2 = 0 . \tag{4.8}$$

Now $u_0 \neq 0$ because otherwise it would follow from (4.4) and (4.6) that $u_0 = v_0 = w_0 = 0$. Therefore (4.8) determines at most two possible values for N, and we are done.

Algorithm POLY for $d = 2$

INPUT: Polynomials

$$p(n) = \sum_{j=0}^{m} u_j n^{m-j}, \; q(n) = \sum_{j=0}^{m} v_j n^{m-j}, \; r(n) = \sum_{j=0}^{m} w_j n^{m-j}$$

such that at least one of u_0, v_0, w_0 is non-zero.

OUTPUT: A basis $\mathcal{B}$ for the space of polynomial solutions of

$$p(n)a_{n+2} + q(n)a_{n+1} + r(n)a_n = 0 . \tag{4.9}$$

```
[1] If $u_0 + v_0 + w_0 \neq 0$ then $\mathcal{D} := \emptyset$
    else if $u_0 \neq w_0$ then $\mathcal{D} := \{N;\ (u_0 - w_0)N + u_1 + v_1 + w_1 = 0\} \cap \mathbb{N}$
    else if $u_1 + v_1 + w_1 \neq 0$ then $\mathcal{D} := \emptyset$
    else $\mathcal{D} := \{N;\ u_0N^2 + (u_1 - u_0 - w_1)N + u_2 + v_2 + w_2 = 0\} \cap \mathbb{N}$.

[2] If $\mathcal{D} = \emptyset$ then $\mathcal{B} := \emptyset$
    else
        $k := \max \mathcal{D}$;
        let $\mathcal{B}$ be a basis for the space of polynomial solutions of (4.9)
        of degree at most $k$.

[3] Return $\mathcal{B}$.
```

Example 1 The equation

$$n(n+1)a_{n+2} - 2n(n+100)a_{n+1} + (n+99)(n+100)a_n = 0$$

has $m = 2$, $u_0 = w_0 = 1$, $v_0 = -2$, $u_1 = 1$, $v_1 = -200$, $w_1 = 199$, $u_2 = v_2 = 0$, and $w_2 = 9900$. Therefore $u_0 + v_0 + w_0 = 0$, $u_0 - w_0 = 0$, $u_1 + v_1 + w_1 = 0$, and the possible degrees are among the roots of

$$N^2 - 199N + 9900 = 0\,,$$

which are $N = 99$ and $N = 100$. In fact, both $n(n+1)\cdots(n+98)$ and $n(n+1)\cdots(n+99)$ are solutions of this equation.

Algorithm POLY for arbitrary d

```
INPUT: Polynomials
        $$p_i(n) = \sum_{j=0}^{m} c_{i,j} n^{m-j}, \quad \text{for } i = 0, 1, \ldots, d, \qquad (4.10)$$
    such that at least one of $c_{i,0}$, $0 \le i \le d$, is non-zero.
    We assume that $c_{i,j} = 0$ when $j < 0$ or $j > m$.

OUTPUT: A basis $\mathcal{B}$ for the space of polynomial solutions of (4.1).

  [1] Initialize $s := -1$.
  [2] Repeat
          increment $s$ by 1;
          for $j = 0, 1, \ldots, s$ compute
              $$b_j^{(s)} = \sum_{i=0}^{d} i^j c_{i,s-j}$$
      until $\exists j \in \{0, 1, \ldots, s\}$ such that $b_j^{(s)} \neq 0$.
  [3] Let $\mathcal{D}$ be the set of nonnegative integer roots $N$ of the polynomial
              $$\sum_{j=0}^{s} b_j^{(s)} \binom{N}{j}\,.$$
  [4] If $\mathcal{D} = \emptyset$ then $\mathcal{B} := \emptyset$
      else
          $k := \max \mathcal{D}$;
```

let $\mathcal{B}$ be a basis for the space of polynomial solutions of (4.1)

of degree at most k.

[5] Return $\mathcal{B}$.

To prove correctness of algorithm POLY we have to show that the loop on step [2] eventually terminates, that the set $\mathcal{D}$ is finite, and that the degree of any non-zero polynomial solution of equation (4.1) belongs to $\mathcal{D}$. These facts are established by the following three lemmas.

Lemma 4 *In algorithm* POLY, *$s \le d$ at all times.*

Proof: Assume that at some point $s = s_0$. Then $b_j^{(s)} = 0$ for $0 \le j \le s < s_0$. In particular, $b_s^{(s)} = 0$ for $0 \le s < s_0$. If $s_0 > d$ this implies that

$$b_s^{(s)} = \sum_{i=0}^{d} i^s c_{i,0} = 0\,, \quad \text{for } 0 \le s \le d\,.$$

Since $\det(i^s)_{i=0,\dots,d}^{s=0,\dots,d} = V(0,1,\dots,d)$ is a Vandermonde determinant it follows that $c_{i,0} = 0$ for $i = 0,1,\dots,d$. But these are the coefficients of n^m in $p_i(n)$, and by our definition of m at least one of the $p_i(n)$ is of degree m, a contradiction. □

Lemma 5 *In algorithm* POLY, *$\mathcal{D}$ is a finite set.*

Proof: By Lemma 4, the loop at step [2] terminates after at most $d+1$ iterations. When this happens at least one of the $b_j^{(s)}$ is not zero, hence the polynomial at step [3] does not vanish identically and has a finite set of roots. Being a subset, $\mathcal{D}$ is finite, too. □

Lemma 6 *Let s_0 be the final value of s in algorithm* POLY. *Let*

$$a_n = \sum_{k=0}^{N_0} \alpha_k\, n^{N_0-k}, \quad \textit{where } \alpha_0 \ne 0 \tag{4.11}$$

be a polynomial solution of equation (4.1). *Then*

$$\sum_{j=0}^{s_0} \binom{N_0}{j} \sum_{i=0}^{d} i^j c_{i,s_0-j} = 0\,. \tag{4.12}$$

Proof: By the Binomial Theorem,

$$a_{n+i} = \sum_{k=0}^{N_0} \alpha_k \sum_{l=0}^{N_0-k} \binom{N_0-k}{l} i^{N_0-k-l} n^l\,. \tag{4.13}$$

We adopt the convention that $c_{i,j} = 0$ and $\alpha_k = 0$ whenever any of i, j, k are out of bounds indicated in (4.10) and (4.11). Then, by inserting (4.10) and (4.13) into (4.1), we get

$$\sum_{i,j,k,l} c_{i,j}\alpha_k \binom{N_0-k}{l} i^{N_0-k-l} n^{m+l-j} = 0$$

where each summation index ranges over all integers, and the order of summation can be chosen at will. Replacing j by $r = m + l - j$ gives

$$\sum_r n^r \sum_{i,k,l} c_{i,m+l-r} \alpha_k \binom{N_0 - k}{l} i^{N_0-k-l} = 0 .$$

By construction, the left-hand side is a polynomial in n. Being identically zero, all its coefficients vanish:

$$\sum_k \alpha_k \sum_l \binom{N_0 - k}{l} \sum_i i^{N_0-k-l} c_{i,m+l-r} = 0 , \quad \text{for all } r . \tag{4.14}$$

By definition of s_0,

$$b_j^{(s)} = \sum_i i^j c_{i,s-j} = 0 , \quad \text{for } 0 \le j \le s < s_0 . \tag{4.15}$$

By our convention, (4.15) also holds when $j > s$, so it certainly holds when $j \ge 0$ and $s < s_0$.

The innermost sum in (4.14) has the form of the sum in (4.15) with $j = N_0 - k - l$ and $s = N_0 + m - k - r$. When $j < 0$ the binomial in (4.14) is zero; when $j \ge 0$ and $s < s_0$ the innermost sum in (4.14) is zero, by (4.15). Replace r by $t = N_0 + m - s_0 - r$. Then $s < s_0$ is equivalent to $k > t$; hence all terms of the middle sum in (4.14) vanish provided that $k > t$, and we can rewrite (4.14) as

$$\sum_{k=0}^{t} \alpha_k \sum_l \binom{N_0 - k}{l} \sum_i i^{N_0-k-l} c_{i,\,l+t+s_0-N_0} = 0 , \quad \text{for all } t . \tag{4.16}$$

In particular, for $t = 0$ we have

$$\alpha_0 \sum_l \binom{N_0}{l} \sum_i i^{N_0-l} c_{i,l+s_0-N_0} = 0 .$$

After canceling α_0 and replacing l by $j = N_0 - l$, this turns into

$$\sum_j \binom{N_0}{j} \sum_i i^j c_{i,s_0-j} = 0$$

which is equivalent to (4.12). □

We can also use POLY to find polynomial solutions of NLP's. Let a_n be a polynomial sequence satisfying $L_n a_n = p_n$ where L_n is a linear difference operator with polynomial coefficients. Then p_n is a polynomial sequence, and a_n satisfies $L_n a_n p_{n+1} - L_{n+1} a_{n+1} p_n = 0$ which is an HLP of one-higher order. We apply POLY to this HLP and find a general solution which we then substitute for a_n into the original NLP. This yields a system of linear algebraic equations for the unknown coefficients.

4.3 A decomposition for rational functions

The existence part of the following lemma is stated in [Gos78] (without mentioning properties 3 and 4). The algorithm HYPER given in Section 4 below relies likewise on the existence part, while the uniqueness part can be used to decide whether a hypergeometric sequence is in fact polynomial.

Lemma 7 *Let F be a field of characteristic zero. Every non-zero rational function $r(x)$ over F has a unique decomposition of the form*

$$r(x) = Z\frac{A(x)}{B(x)}\frac{C(x+1)}{C(x)} \tag{4.17}$$

where

1. $Z \in F$, $Z \neq 0$,
2. $A(x), B(x), C(x)$ *are monic polynomials over* F,
3. $A(x), C(x)$ *are relatively prime,*
4. $B(x), C(x+1)$ *are relatively prime,*
5. $A(x), B(x+k)$ *are relatively prime for every nonnegative integer* k.

Proof of existence: Let $r(x) = p(x)/q(x)$ where p, q are polynomials over F with leading coefficients α and β, respectively. Let $Z := \alpha/\beta$. Then $r(x) \neq 0$ implies $p(x) \neq 0$ which implies $\alpha \neq 0$ which implies $Z \neq 0$.

If the polynomial $R(k) = \text{Resultant}(p(x), q(x+k); x)$ has no nonnegative integer roots, then Z, $A(x) := p(x)/\alpha$, $B(x) := q(x)/\beta$, and $C(x) := 1$ satisfy 1 – 5.

Otherwise, let N be the largest nonnegative integer root of $R(k)$. Construct polynomials $p_i(x)$, $q_i(x)$ with $-1 \le i \le N$ inductively by

$$p_{-1}(x) = \frac{p(x)}{\alpha}, \quad q_{-1}(x) = \frac{q(x)}{\beta}$$

and for $0 \le i \le N$,

$$\begin{aligned} s_i(x) &= \gcd(p_{i-1}(x), q_{i-1}(x+i)), \\ p_i(x) &= \frac{p_{i-1}(x)}{s_i(x)}, \\ q_i(x) &= \frac{q_{i-1}(x)}{s_i(x-i)}. \end{aligned}$$

Let

$$\begin{aligned} A(x) &:= p_N(x), \\ B(x) &:= q_N(x), \\ C(x) &:= \prod_{i=1}^{N}\prod_{k=1}^{i} s_i(x-k). \end{aligned}$$

Then

$$\begin{aligned} Z\frac{A(x)}{B(x)}\frac{C(x+1)}{C(x)} &= \frac{\alpha}{\beta}\frac{p_N(x)}{q_N(x)}\prod_{i=1}^{N}\prod_{k=1}^{i}\frac{s_i(x-k+1)}{s_i(x-k)} \\ &= \frac{\alpha}{\beta}\frac{p_{-1}(x)}{\prod_{i=0}^{N} s_i(x)}\frac{\prod_{i=0}^{N} s_i(x-i)}{q_{-1}(x)}\prod_{i=1}^{N}\frac{s_i(x)}{s_i(x-i)} \\ &= \frac{\alpha p_{-1}(x)}{\beta q_{-1}(x)} = \frac{p(x)}{q(x)} = r(x). \end{aligned}$$

The polynomials $A(x), B(x), C(x)$ are monic by construction. Now we verify 3 – 5.

By definition of p_k, q_k, and s_k,

$$\gcd(p_k(x), q_k(x+k)) = \gcd(\frac{p_{k-1}(x)}{s_k(x)}, \frac{q_{k-1}(x+k)}{s_k(x)}) = 1$$

for all k such that $0 \le k \le N$. Let $0 \le k \le i, j \le N$. Then $p_i, p_j \mid p_k$ and $q_i, q_j \mid q_k$, therefore

$$\gcd(p_i(x), q_j(x+k)) \mid \gcd(p_k(x), q_k(x+k)) = 1\,. \tag{4.18}$$

3. If $A(x)$ and $C(x)$ have a common factor then so do $p_N(x)$ and $s_i(x-k)$, for some i and k such that $1 \le k \le i \le N$. Then $p_N(x)$ and $q_{i-1}(x+i-k)$ have a common factor, too, by definition of q_i. But as $0 \le i-k \le i-1 < N$, this contradicts (4.18).

4. If $B(x)$ and $C(x+1)$ have a common factor then so do $q_N(x)$ and $s_i(x-k)$, for some i and k such that $0 \le k \le i-1 \le N-1$. Then $q_N(x)$ and $p_{i-1}(x-k)$ have a common factor, too, by definition of p_i. Hence $q_N(x+k)$ and $p_{i-1}(x)$ have a common factor. But as $0 \le k \le i-1 < N$, this contradicts (4.18).

5. Setting $i = j = N$ in (4.18), we see that $A(x)$ and $B(x+k)$ are relatively prime for all k such that $0 \le k \le N$. By definition of N this implies that they are relatively prime for all $k \ge 0$.

Proof of uniqueness: Assume that

$$r(x) = Z\frac{A(x)}{B(x)}\frac{C(x+1)}{C(x)} = z\frac{a(x)}{b(x)}\frac{c(x+1)}{c(x)} \tag{4.19}$$

where Z, A, B, C as well as z, a, b, c satisfy 1 – 5. Since all the polynomials appearing in (4.19) are monic, we have $Z = z$, and

$$A(x)C(x+1)b(x)c(x) = a(x)c(x+1)B(x)C(x) \tag{4.20}$$

Therefore

$$\prod_{i=0}^{k} A(x+i)C(x+i+1)b(x+i)c(x+i) = \prod_{i=0}^{k} a(x+i)c(x+i+1)B(x+i)C(x+i) \tag{4.21}$$

for every $k \ge 0$. Canceling c's and C's, we obtain

$$c(x)C(x+k+1)\prod_{i=0}^{k} A(x+i)b(x+i) = c(x+k+1)C(x)\prod_{i=0}^{k} a(x+i)B(x+i)\,, \tag{4.22}$$

whence

$$A(x) \mid c(x+k+1)C(x)\prod_{i=0}^{k} a(x+i)B(x+i)$$

and, by 3 and 5,

$$A(x) \mid c(x+k+1)\prod_{i=0}^{k} a(x+i) \quad \text{for every } k \ge 0\,. \tag{4.23}$$

If a polynomial $d(x)$ divides $c(x+j+1)\prod_{i=1}^{j} a(x+i)$ for all j such that $0 \le j \le k$, then $d(x)$ divides $c(x+j+1)$ for all j such that $0 \le j \le k$. We prove this by induction on k.

If $k = 0$ the assertion is a tautology.

For the induction step, assume that $d(x)$ divides $c(x+j+1)\prod_{i=1}^{j} a(x+i)$ for all j such that $0 \le j \le k$, and that the assertion holds for all smaller k. Then $d(x)$ divides $c(x+j+1)$ for all j such that $0 \le j \le k-1$. By 3, this implies that $d(x)$ is relatively prime with $\prod_{i=1}^{k} a(x+i)$. Therefore $d(x)$ divides $c(x+k+1)$. This completes the proof by induction on k.

From what we have just proved it follows that

$$\begin{aligned}
\gcd_{0\le j\le k} c(x+j+1)\prod_{i=0}^{j} a(x+i) &= a(x)\gcd_{0\le j\le k} c(x+j+1)\prod_{i=1}^{j} a(x+i) \\
&= a(x)\gcd_{0\le j\le k} c(x+j+1) \\
&= a(x)\gcd_{1\le j\le k+1} c(x+j)\,.
\end{aligned}$$

Hence we can conclude from (4.23) that $A(x)$ divides $a(x)\gcd_{1\le j\le k} c(x+j)$ for all $k \ge 1$.

We claim that for large enough k, $\gcd_{1\le j\le k} c(x+j) = 1$. Assume not. Then there is an element $\alpha \in \overline{F}$ which is a root of $c(x+j)$ for all $j \ge 1$, where $\overline{F}$ is the algebraic closure of F. Hence $\alpha + j$ is a root of $c(x)$ for all $j \ge 1$. Because $\overline{F}$ has characteristic zero, this means that $c(x)$ has infinitely many distinct roots in $\overline{F}$. Therefore $c(x)$ is the zero polynomial. Since $c(x)$ is monic, this is a contradiction, and the claim is proved.

It follows that $A(x)$ divides $a(x)$. In the same way, we can prove that $a(x)$ divides $A(x)$. Since they are monic, this implies that they are equal.

In a similar way, we can prove from (4.22) that $B(x)$ and $b(x)$ are equal. Then it follows from (4.20) that

$$C(x+1)c(x) = c(x+1)C(x)\,. \tag{4.24}$$

Let N be an integer such that $C(n), c(n) \ne 0$ for all integers $n \ge N$. Such an N exists because F has characteristic zero. Then it follows from (4.24) that

$$C(N)c(n) = c(N)C(n)\,, \quad \text{for all } n \ge N\,, \tag{4.25}$$

by induction on n. Thus the polynomials $C(N)c(x)$ and $c(N)C(x)$ agree for infinitely many $x \in F$ and are consequently equal. Both $c(x)$ and $C(x)$ are monic, therefore $C(N) = c(N)$, by comparison of leading coefficients, and $C(x)$ is equal to $c(x)$. □

4.4 Hypergeometric solutions

As with polynomial solutions, we consider first the second-order equation (4.3). Assume that (a_n) is a hypergeometric solution of (4.3). Then there is a rational sequence S_n such that $a_{n+1} = S_n a_n$. Plugging this into (4.3) and canceling a_n gives

$$p(n)S_{n+1}S_n + q(n)S_n + r(n) = 0\,.$$

Let

$$S_n = Z\frac{A_n}{B_n}\frac{C_{n+1}}{C_n}$$

be the decomposition of S_n described in Lemma 7. Then

$$Z^2p(n)A_{n+1}A_nC_{n+2} + Zq(n)B_{n+1}A_nC_{n+1} + r(n)B_{n+1}B_nC_n = 0\,. \tag{4.26}$$

The first two terms contain A_n as a factor, therefore $A_n \mid r(n)B_{n+1}B_nC_n$. By properties 3 and 5 of the decomposition, A_n is relatively prime with C_n, B_n, and B_{n+1}, hence $A_n \mid r(n)$. Similarly we find that $B_{n+1} \mid Z^2p(n)A_{n+1}A_nC_{n+2}$, hence by properties 4 and 5 of the decomposition $B_{n+1} \mid p(n)$ and $B_n \mid p(n-1)$. So the choice of A_n and B_n has been narrowed down to a finite set – the set of monic factors of $r(n)$ and $p(n-1)$, respectively. Also, (4.26) can be rewritten with lower degree as

$$Z^2\frac{p(n)}{B_{n+1}}A_{n+1}C_{n+2} + Zq(n)C_{n+1} + \frac{r(n)}{A_n}B_nC_n = 0\,. \tag{4.27}$$

To determine the constant Z, we look at the leading coefficient of the left-hand side (which is a polynomial in n) and see that Z satisfies an algebraic equation of degree 2 or less. So given the choice of A_n and B_n, there are at most two choices for Z.

For a fixed choice of A_n, B_n, and Z, (4.27) is an HLP, and we can use algorithm POLY to determine if it has any non-zero polynomial solution C_n. If yes, we have found a hypergeometric solution of (4.3). If (4.27) has no non-zero polynomial solution whatever the choice of A_n, B_n, and Z, then (4.3) has no hypergeometric solution.

Algorithm HYPER for $d = 2$

```
INPUT: Polynomials $p(n)$, $q(n)$, and $r(n)$.

OUTPUT: A hypergeometric solution $(a_n)$ of (4.3) if it exists;

        0 otherwise.

  [1] For all monic factors $A_n$ of $r(n)$ and $B_n$ of $p(n-1)$ do:

        $P_n := (p(n)/B_{n+1})A_{n+1}$;  $Q_n := q(n)$;  $R_n := (r(n)/A_n)B_n$;

        $m := \max\deg\{P_n, Q_n, R_n\}$;

        let $\alpha$, $\beta$, $\gamma$ be the coefficients of $n^m$ in $P_n$, $Q_n$, $R_n$, respectively;

        for all non-zero $Z$ such that $\alpha Z^2 + \beta Z + \gamma = 0$ do:

          If the equation $Z^2P_nC_{n+2} + ZQ_nC_{n+1} + R_nC_n = 0$ has

          a non-zero polynomial solution $C_n$ then

            $S_n := Z(A_n/B_n)(C_{n+1}/C_n)$;

            return a non-zero solution of $a_{n+1} = S_na_n$ and exit.

  [2] Return 0.
```

Example 2 The equation

$$(n+4)a_{n+2} + a_{n+1} - (n+1)a_n = 0$$

appears in Bender and Orszag ([BO78], p. 43, Example 2.3.5). The monic factors of $r(n)$ are 1 and $n+1$, and those of $p(n-1)$ are 1 and $n+3$. Taking $A_n = B_n = 1$ leads to $Z = \pm 1$; however, in algorithm POLY both values produce negative degrees for potential polynomial solutions of the auxiliary equation. If exactly one of A_n, B_n is equal to 1, the equation for Z is either $Z = 0$ or contradictory. Finally, the choice $A_n = n+1$, $B_n = n+3$ yields $Z = \pm 1$ again. For $Z = 1$, the auxiliary equation is

$$(n+2)C_{n+2} + C_{n+1} - (n+3)C_n = 0$$

with unique (up to a constant factor) polynomial solution $C_n = 1$. This gives $S_n = (n+1)/(n+3)$ and

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

For $Z = -1$, the auxiliary equation is

$$(n+2)C_{n+2} - C_{n+1} - (n+3)C_n = 0$$

with unique (up to a constant factor) polynomial solution $C_n = 2n+3$. This gives $S_n = -((n+1)/(n+3))((2n+5)/(2n+3))$ and

$$a_n = \frac{(-1)^n(2n+3)}{(n+1)(n+2)}.$$

Example 3 The number d_n of derangements of a set with n elements satisfies the equation

$$a_{n+2} - (n+1)a_{n+1} - (n+1)a_n = 0. \tag{4.28}$$

Taking $A_n = B_n = 1$ yields $Z = -1$, but the auxiliary equation has no non-zero polynomial solution. The only other choice ($A_n = n+1$, $B_n = 1$) yields $Z^2 - Z = 0$, so $Z = 1$. The auxiliary equation

$$(n+2)C_{n+2} - (n+1)C_{n+1} - C_n = 0$$

has the unique (up to a constant factor) polynomial solution $C_n = 1$. This gives $S_n = n+1$ and

$$a_n = n!.$$

This is the unique (up to a constant factor) hypergeometric solution of (4.28). Using the well-known method of reduction of order, we find another solution

$$a_n = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!}. \tag{4.29}$$

Since this is not a constant multiple of $n!$ (although it comes very close, being equal to the integer nearest to $n!/e$), we have proved that (4.29) is not a hypergeometric sequence. As the summand in (4.29) does not depend on n this could have also been shown by Gosper's algorithm.

Example 4 The number i_n of involutions of a set with n elements satisfies the equation

$$a_{n+2} - a_{n+1} - (n+1)a_n = 0. \tag{4.30}$$

Here either $A_n = B_n = 1$ or $A_n = n+1$ and $B_n = 1$. The equation for Z is contradictory in the first case and $Z = 0$ in the second, so (4.30) has no hypergeometric solution. This proves that the well-known expression for i_n (see, for example, [Com74]),

$$i_n = \sum_k \frac{n!}{(n-2k)!\,2^k\,k!} \tag{4.31}$$

is not a hypergeometric sequence.

Example 5 In [vdP79], it is shown that the numbers

$$a_n = \sum_{k=0}^{n} \binom{n}{k}^2 \binom{n+k}{k}^2 \tag{4.32}$$

satisfy the equation

$$(n+2)^3 a_{n+2} - (2n+3)(17n^2+51n+39)a_{n+1} + (n+1)^3 a_n = 0\,. \tag{4.33}$$

Here all the coefficients are of the same degree, therefore the equation for Z will have no non-zero solution unless A_n and B_n are of the same degree as well. But they are both monic factors of $(n+1)^3$, so they must be equal. Then $P_n = p(n)$, $Q_n = q(n)$, $R_n = r(n)$, and the equation for Z is $Z^2 - 34Z + 1 = 0$ with solutions $Z = 17 \pm 12\sqrt{2}$. In both cases the only potential degree for non-zero polynomial solutions of the auxiliary equation turns out to be $-2/3$, proving that (4.33) has no hypergeometric solution. As a consequence, the sequence (4.32) is not hypergeometric.

We remark that as the summands in (4.31) and (4.32) depend on n, those sums cannot be shown to be nonhypergeometric by Gosper's algorithm.

Algorithm HYPER for arbitrary d

```
INPUT: Polynomials $p_i(n)$ for $i = 0, 1, \ldots, d$.

OUTPUT: A hypergeometric solution $(a_n)$ of (4.1) if it exists;
        0 otherwise.

  [1] For all monic factors $A_n$ of $p_0(n)$ and $B_n$ of $p_d(n-d+1)$ do:

        $P_i(n) := p_i(n) \prod_{j=0}^{i-1} A_{n+j} \prod_{j=i}^{d-1} B_{n+j}$, for $i = 0, 1, \ldots, d$;

        $m := \max_{0 \le i \le d} \deg P_i(n)$;

        let $\alpha_i$ be the coefficient of $n^m$ in $P_i(n)$, for $i = 0, 1, \ldots, d$;

        for all non-zero $Z$ such that $\sum_{i=0}^{d} \alpha_i Z^i = 0$ do:

          If the equation
```

$$\sum_{i=0}^{d} Z^i P_i(n) C_{n+i} = 0 \tag{4.34}$$

has a non-zero polynomial solution C_n then

$S_n := Z(A_n/B_n)(C_{n+1}/C_n)$;

return a non-zero solution of $a_{n+1} = S_n a_n$ and exit.

[2] Return 0.

We can reduce the degree of equation (4.34) by canceling the common factor $A_n B_{n+d-1}$ (as we did in (4.27) for the case $d = 2$).

Lemma 8 *Let (a_n) be a non-zero solution of* (4.1) *such that $a_{n+1} = S_n a_n$ where S_n is a rational sequence. Let*

$$S_n = Z \frac{A_n}{B_n} \frac{C_{n+1}}{C_n} \tag{4.35}$$

be the decomposition described in Lemma 7. Let $P_i(n)$ and α_i, for $i = 0, 1, \ldots, d$, be defined as in algorithm HYPER. *Then $\sum_{i=0}^{d} \alpha_i Z^i = 0$, A_n divides $p_0(n)$, B_n divides $p_d(n-d+1)$, and C_n satisfies* (4.34).

Conversely, if Z is an arbitrary constant, A_n and B_n arbitrary sequences, C_n satisfies (4.34) *where $P_i(n)$, for $i = 0, 1, \ldots, d$, is defined as in algorithm* HYPER, *and $a_{n+1} = S_n a_n$ where S_n is as in* (4.35), *then (a_n) satisfies* (4.1).

Proof: From (4.1) and $a_{n+1} = S_n a_n$ it follows that

$$\sum_{i=0}^{d} p_i(n) \prod_{j=0}^{i-1} S_{n+j} a_n = 0 \tag{4.36}$$

hence after canceling a_n and using (4.35) we have

$$\sum_{i=0}^{d} p_i(n) Z^i \prod_{j=0}^{i-1} \frac{A_{n+j}}{B_{n+j}} \frac{C_{n+1}}{C_n} = 0\,. \tag{4.37}$$

Multiplication by $C_n \prod_{j=0}^{d-1} B_{n+j}$ now gives (4.34). All terms in the sum in (4.34) with $i > 0$ contain the factor A_n, hence A_n divides the term with $i = 0$ which is $p_0(n) C_n \prod_{j=0}^{d-1} B_{n+j}$. By properties 3 and 5 of decomposition (4.35), it follows that A_n divides $p_0(n)$. Similarly, B_{n+d-1} divides $Z^d p_d(n) C_{n+d} \prod_{j=0}^{d-1} A_{n+j}$, hence by properties 3 and 5 of decomposition (4.35), B_{n+d-1} divides $p_d(n)$, and B_n divides $p_d(n-d+1)$. Finally, a look at the leading coefficient of the left-hand side of (4.34) shows that $\sum_{i=0}^{d} \alpha_i Z^i = 0$.

To prove the converse, we retrace the steps which led from (4.1) through (4.36) and (4.37) to (4.34), in reverse order. □

4.5 Extensions and simplifications

1. Given an HLP, the algorithm HYPER will decide existence of hypergeometric solutions. But it can also be used to find a basis for the space of solutions which belong to $\mathcal{L}(H)$, the space of all F-linear combinations of hypergeometric sequences.

Call two hypergeometric sequences *similar* if they have rational ratio. It is easy to see the following: Similarity is a congruence relation in the multiplicative group H of hypergeometric sequences. If the sum of two similar hypergeometric sequences is non-zero then the sum is hypergeometric, too. If L is a linear difference operator with polynomial coefficients, a is hypergeometric, and $La \neq 0$ then La is similar to a. Also, one can show that a set S of hypergeometric sequences is linearly dependent over the field of rational sequences if and only if S contains two similar sequences. It follows that if a linear combination of pairwise dissimilar hypergeometric sequences solves an HLP then so does each individual term of this combination.

Let r be a fixed rational sequence. Consider the mapping $f : C \mapsto h$ where C is a non-zero polynomial sequence and h is any hypergeometric sequence satisfying

$$h_{n+1} = r_n \frac{C_{n+1}}{C_n} h_n .$$

It is not hard to see that for any non-zero polynomial sequences p, q and for any $\lambda, \mu \in F$ there are $\lambda', \mu' \in F$ such that $f(\lambda p + \mu q) = \lambda' f(p) + \mu' f(q)$.

Therefore running through all possible choices of A_n, B_n, and Z in algorithm HYPER, and using POLY to construct a basis for the space of polynomial solutions of each auxiliary equation, will produce a generating set for the space of solutions which belong to $\mathcal{L}(H)$. (In general, this will not be a basis because solutions coming from different auxiliary equations can be linearly dependent). One can prune this generating set testing linear independence by means of the Casoratian determinant until it becomes a basis.

Another way to obtain such a basis is to find one solution with HYPER, then reduce the order of the equation, recursively find the corresponding basis for the reduced equation, and then use Gosper's algorithm to put the antidifferences of these solutions into closed form. If there is at least one hypergeometric solution h, this method will also yield those solutions a for which $\Delta(a/h)$ is hypergeometric (cf. Example 3).

2. Noting that the right-hand side of an NLP with a hypergeometric solution is hypergeometric, we can also use HYPER to find hypergeometric solutions of NLP's in a similar way as we can use POLY on NLP's (see the last paragraph of Section 4.2). However, this means that we are throwing away the information about the similarity class of potential hypergeometric solutions which is determined by the right-hand side, and then rediscovering it by factoring. A much better method is to seek solution as an unknown rational multiple of the right-hand side and then use Abramov's algorithm to find all rational solutions of the resulting equation. Like Gosper's algorithm, Abramov's algorithm requires computation of polynomial gcd's and resultants but no factoring.

3. Given an HLP for (a_n) we can construct an HLP of the same (or possibly lower) order for (Δa_n), by first applying Δ to the original equation to produce an auxiliary equation, then expressing a_{n+k} in both equations in terms of Δa_{n+j} and a_n, and finally eliminating a_n among them. Repeating this several times we see that algorithm HYPER can be used to find all solutions of a given HLP whose k-th differences are hypergeometric, for any given k.

We conclude the chapter by pointing out several possible shortcuts in the implementation of algorithm HYPER:

1. Consider only those pairs A_n, B_n for which the polynomial $R(k) = \text{Resultant}(A_n, B_{n+k}; n)$ has no nonnegative integer roots.

2. If $\deg p_0 + \deg p_d = 2m + 1$ for some integer m, and $\deg p_i \leq m$ for $1 \leq i \leq d - 1$, then the equation has no hypergeometric solutions (cf. Example 4).

3. If there is a k such that $1 \leq k \leq d-1$, $\deg p_k > \deg p_i$ for all $i \neq k$, and $\deg p_k > \deg p_0 + \deg p_d$, then the equation has no hypergeometric solutions.

4. Discard any auxiliary equation where one coefficient dominates all others in degree (cf. Example 5).

Chapter 5

Galois theory of difference equations

There are more things in heaven and earth, Horatio,
Than are dreamt of in your philosophy.
— HAMLET

5.1 Introduction

There are many similarities and analogies between the theories of difference and differential equations. The algebraic theory of differential equations which is one of the tools for investigating existence of closed-form solutions was developed under the name of *differential algebra*, principally by Ritt and Kolchin (see, for example, Ritt [Rit32], Kaplansky [Kap57], and Kolchin [Kol73]). Within this framework an analogue of Galois theory of algebraic equations has been constructed using the notion of differential fields. A *differential field* is a field together with a mapping ' (the *derivation*) such that $(a+b)' = a' + b'$ and $(ab)' = a'b + ab'$. An element a of a differential field is called *constant* if $a' = 0$. A *Picard–Vessiot extension* K of a differential field k is obtained by adjoining to k a basis of the space of solutions of a homogeneous linear differential equation with coefficients in k, provided that the fields of constants of k and K coincide. A differential field K is a *Liouvillian extension* of k if there is a finite chain of differential fields $k = F_0 \subseteq F_1 \subseteq \ldots \subseteq F_k = K$ such that F_i is either finite algebraic over F_{i-1}, or is obtained from F_{i-1} by adjunction of some y which satisfies an equation of the form $y' = ay$ or $y' = b$, where a and b are elements of F_{i-1}. One of the main results in this theory is that the Galois group of a Picard–Vessiot extension is an algebraic group. This is the basis for construction of decision algorithms for the problem of existence of closed-form solutions of a given linear differential equation with coefficients in k. Here a solution is defined to be in closed form if it lies in a Liouvillian extension of k.

On the algorithmic side, Baldassari and Dwork [BD79] as well as Singer [Sin80] gave an algorithm for deciding whether a homogeneous linear differential equation with rational coefficients has algebraic solutions, and Baldassari [Bal80] extended this to equations with algebraic coefficients. Kovacic [Kov86] was the first to give an algorithm for deciding whether a second-order homogeneous linear differential equation with rational coefficients has Liouvillian solutions, and if so, to find them, although his results weren't published until a few years later. His algorithm has been implemented in several symbolic computation systems. Singer [Sin81] extended this to n-th degree equations. Later, Singer [Sin85] attacked the question of solvability by means of adjoining solutions of second-order equations, while Davenport and Singer [DS86] gave an algorithm for finding Liouvillian solutions of nonhomogeneous linear equations.

An overview of Galois theory of differential equations can be found in [Sin90] and [Sin].

The development of an analogue of differential algebra in the theory of difference equations was started by Ritt, and carried on by Cohn [Coh65] under the name of *difference algebra*. An *inversive difference field* is a field together with an automorphism τ (the *transform* or the *shift operator*). For difference equations, Galois theory was developed in a series of papers by Franke: [Fra63], [Fra66], [Fra67], [Fra69], [Fra71], [Fra73], [Fra74]. A difference field K is a *Liouvillian extension* of k if there is a finite chain of difference fields $k = F_0 \subseteq F_1 \subseteq \ldots \subseteq F_k = K$ such that F_i is either finite algebraic over F_{i-1}, or is obtained from F_{i-1} by adjunction of some y which satisfies an equation of the form $\tau y = ay$ or $\tau y - y = b$ where a and b are elements of F_{i-1}. In order to obtain a more satisfactory theory, Franke [Fra66] expanded the definition of closed-form solutions to *q–Liouvillian extensions* which allow for adjunctions of solutions of $\tau^q y = ay$ and $\tau^q y - y = b$. Here q is a positive integer.

Still, this development did not lead to a complete resolution of the problem of existence of Liouvillian solutions as did the analogous line of research in the case of differential equations. The main obstacle was the existence of difference equations without Picard-Vessiot extensions (i.e., such that the adjunction of any fundamental set of solutions introduces new constants). An example is given in Proposition 2 and Corollary 3 below.

Instead of with fields of functions, we work with rings of sequences. In order to get a difference ring the shift operator is converted into an automorphism by identifying sequences which agree from some point on. This works well for rational sequences since we don't have to worry about their poles. The units in this quotient ring are the sequences which are non-zero from some point on. The advantage of this setting is that we have a universal structure in which all solutions lie - the ring of all sequences over the field of coefficients, and adjoining solutions produces no new constants. The setback is the appearance of nonunits (which are also zero divisors).

We prove that in this setting the Galois group of a linear difference operator with rational coefficients is an algebraic matrix group, which is one of the major stepping stones in developing Galois theory. The others are establishing Galois correspondence, proving normality of extensions, and determining periodicity properties of nonunit solutions. This constitutes a major research program in itself.

5.2 Difference rings and fields

Definition 11 [Coh65] A *difference ring* is a commutative ring with unit, K, together with an injective endomorphism, τ, called the *transform* of K. A *difference field* is a difference ring which is a field. A difference ring or field K with transform τ is *inversive* if τ is an automorphism of K.

The set $C(K) = \{x \in K; \tau x = x\}$ is a subring of K called the *ring of constants* of K. If K is a field then so is $C(K)$. □

All difference rings and fields considered will be inversive.

Example 6 Any ring K together with the identity automorphism id_K is an inversive difference ring. In this case, $C(K) = K$.

Example 7 The field of rational functions over the complex numbers $\mathbf{C}(z)$ together with the shift operator $\sigma f(z) = f(z+1)$ is a difference field. It is not difficult to see that the field of constants is isomorphic to $\mathbb{C}$.

Definition 12 Let K be a difference ring with transform τ. A *difference automorphism* of K is an automorphism α of K such that $\alpha\tau = \tau\alpha$. If k, K are difference rings, $k \subseteq K$, and the transform of K agrees on k with the transform of k, then k is a *difference subring* of K, and K is a *difference extension ring* of k. If k is a difference subring of K, and $a_1, a_2, \ldots, a_r$ are elements of K, then $k\{a_1, a_2, \ldots, a_r\}$ denotes the smallest difference subring of K which contains the set $k \cup \{a_1, a_2, \ldots, a_r\}$. It is equal to the (algebraic) extension ring of k obtained by adjoining $\tau^{j-1}a_i$, for $1 \le i \le r, 1 \le j$. $\square$

The main difficulty in applications of Galois theory of difference fields lies in the fact that, unlike differential equations, there exist difference equations with coefficients in a difference field k such that any difference extension field of k in which the equation has a non-zero solution contains constants not in k.

Proposition 2 *[Fra63] Let K be a difference field with transform τ. If the field of constants $C(K)$ is algebraically closed then the equation $\tau x + x = 0$ has no non-zero solution in K.*

Proof: Let $\xi \in K$ be such that $\xi \neq 0$ and $\tau\xi + \xi = 0$. Then

$$\tau\xi^2 = (\tau\xi)^2 = (-\xi)^2 = \xi^2,$$

so $\xi^2 \in C(K)$. Since $C(K)$ is algebraically closed this implies that $\xi \in C(K)$, and therefore $\tau\xi = \xi$. From $\tau\xi + \xi = 0$ it follows that $\xi = 0$, a contradiction. Hence no such ξ exists. $\square$

Corollary 3 *Let k be the difference field $\mathbb{C}(z)$ with the shift operator σ as the transform. Let ξ be a non-zero solution of the equation $\sigma x + x = 0$ in some difference extension field K of k. Then $C(K)$ contains elements which are not in $\mathbb{C}$.*

Proof: By Proposition 2, $C(K)$ is not algebraically closed. Since $\mathbb{C} = C(k) \subseteq C(K)$ and $\mathbb{C}$ is algebraically closed, the assertion follows. $\square$

5.3 Difference algebra of sequences over a field

Let F be a field, $F^{\mathbb{N}}$ the ring of sequences over F with addition and multiplication defined componentwise, and $\sigma : F^{\mathbb{N}} \to F^{\mathbb{N}}$ the shift operator defined for $f \in F^{\mathbb{N}}$ by

$$\sigma f(n) = f(n+1), \quad \text{for all } n \in \mathbb{N}.$$

Note that σ is a ring endomorphism of $F^{\mathbb{N}}$ which is not injective, and so $F^{\mathbb{N}}$ together with σ is not a difference ring.

Let J be equal to $\bigcup_{k=0}^{\infty} \operatorname{Ker} \sigma^k$, the ideal of eventually zero sequences. Construct the quotient ring

$$C = F^{\mathbb{N}}/J \tag{5.1}$$

and let $\varphi : F^{\mathbb{N}} \to \mathcal{C}$ be the natural epimorphism. Since both σ and φ are epimorphisms, so is $\varphi\sigma : F^{\mathbb{N}} \to \mathcal{C}$. Furthermore,

$$\text{Ker } \varphi\sigma = \sigma^{-1}(J) = \bigcup_{k=0}^{\infty} \sigma^{-k-1}(0) = \bigcup_{k=1}^{\infty} \text{Ker } \sigma^k = J.$$

By the Isomorphism Theorem of universal algebra, there exists a unique automorphism τ of $\mathcal{C}$ such that

$$\tau\varphi = \varphi\sigma \tag{5.2}$$

and the diagram

$$\begin{tikzcd} F^{\mathbb{N}} \arrow[r, "\varphi"] \arrow[d, "\sigma"'] & \mathcal{C} \arrow[d, "\tau"] \\ F^{\mathbb{N}} \arrow[r, "\varphi"] & \mathcal{C} \end{tikzcd}$$

commutes. Thus $\mathcal{C}$ together with τ is an inversive difference ring. The elements of $\mathcal{C}$ are equivalence classes of sequences over F, two sequences being equivalent if they agree from some point on. In [Sta80] such equivalence classes are called "germs at ∞ of functions $f : \mathbb{N} \to F$". One can show easily that $K(\mathcal{C}) = F$, provided that we identify an eventually constant sequence over F with the appropriate element of F. Since $\mathcal{C}$ is in fact an F-algebra and τ is F-linear, we shall call $\mathcal{C}$ together with τ the *difference algebra of sequences over F*.

The non-zero elements of $\mathcal{C}$ either have no zero terms (from some point on), or they have infinitely many zero terms. The former are the units of $\mathcal{C}$, and the latter are zero-divisors. Since τ is an automorphism it transforms units into units and zero-divisors into zero-divisors.

Let $\mathcal{R}$ denote the difference subring of those $a \in \mathcal{C}$ for which there exists a rational function $r(z) \in F(z)$, a sequence $f \in F^{\mathbb{N}}$, and $N \in \mathbb{N}$ such that $f(n) = r(n)$ for $n \geq N$, and $a = \varphi f$. The elements of $\mathcal{R}$ are units, and in fact $\mathcal{R}$ together with τ is a difference field. We shall call it the *difference field of rational sequences over F*. Similarly we can define the *difference ring $\mathcal{P}$ of polynomial sequences over F*. We have

$$K(\mathcal{C}) = F \subseteq \mathcal{P} \subseteq \mathcal{R} \subseteq \mathcal{C}.$$

Since two distinct rational functions over a field can only agree at a finite number of points, $\mathcal{R}$ with τ, and $F(z)$ with the shift operator σ are isomorphic as difference fields. Similarly, $\mathcal{P}$ with τ, and $F[z]$ with σ are isomorphic as difference rings.

In the rest of the chapter, F will denote a field, $\mathcal{C}$ the difference algebra of sequences over F, and $\mathcal{R}$ the difference field of rational sequences over F.

5.4 Linear difference operators on $\mathcal{C}$

Definition 13 A mapping $L : \mathcal{C} \to \mathcal{C}$ of the form

$$Lx = \sum_{k=0}^{d} a_k \tau^k x\,, \quad a_k \in \mathcal{C} \text{ for } k = 0, 1, \ldots, d \tag{5.3}$$

with $a_d \neq 0$ is a *linear difference operator* on $\mathcal{C}$. The *degree* of L is d, and the *effective degree* of L is $e = d - \min\{k;\ a_k \neq 0\}$. L is called *rational* if $a_k \in \mathcal{R}$ for $k = 0, 1, \ldots, d$. □

A linear difference operator is a linear map of the F-vector space C into itself, so its kernel is a linear subspace of C. If L is such an operator a basis for Ker$L$ over F is called a set of *fundamental solutions* for L.

Lemma 9 *Let L be a linear difference operator of the form* (5.3) *of effective degree e, and such that either a_d or a_{d-e} is a unit of C. Then* $\dim \mathrm{Ker} L \leq e$.

Proof: Let $x_1, x_2, \ldots, x_{e+1}$ be solutions of the equation $Lx = 0$. Let $y_1, y_2, \ldots, y_{e+1} \in F^{\mathbb{N}}$ be such that $x_i = \varphi y_i$, for $i = 1, 2, \ldots, e+1$. Let $b_0, b_1, \ldots, b_d \in F^{\mathbb{N}}$ be such that $a_i = \varphi b_i$, for $i = 0, 1, \ldots, d$. Then by (5.2) and (5.3),

$$\varphi\left(\sum_{j=d-e}^{d} b_j\, \sigma^j y_i\right) = \sum_{j=0}^{d} a_j\, \tau^j x_i = 0\,, \qquad \text{for } i = 1, 2, \ldots, e+1.$$

Therefore there exists an $N \in \mathbb{N}$ such that for all $n \geq N$,

$$\sum_{j=d-e}^{d} b_j(n) y_i(n+j) = 0\,, \qquad \text{for } i = 1, 2, \ldots, e+1. \tag{5.4}$$

Let $\mathbf{y}(n) \in F^{e+1}$ be the vector with components $y_1(n), y_2(n), \ldots, y_{e+1}(n)$, for all $n \geq 0$. Denote by $\mathcal{L}_n$ the linear span of $\mathbf{y}(n), \mathbf{y}(n+1), \ldots, \mathbf{y}(n+e-1)$, and let $\mathcal{O}_n := \{\mathbf{u};\ \sum_{i=1}^{e+1} u_i v_i = 0, \mathrm{for all} \mathbf{v} \in \mathcal{L}_n\}$. Since $\dim \mathcal{L}_n \leq e$, it follows that $e + 1 \geq \dim \mathcal{O}_n \geq 1$, for all $n \geq 0$.

Now we split the proof into two cases:

Case 1: a_d is a unit. Then there exists an $M \in \mathbb{N}$ such that $b_d(n) \neq 0$ for all $n \geq M$. Let $k = \max\{N, M\}$. Then by (5.4), for all $n \geq k$,

$$y_i(n+d) = -\sum_{j=d-e}^{d-1} b_j(n)/b_d(n) y_i(n+j)\,, \qquad \text{for } i = 1, 2, \ldots, e+1.$$

Hence $\mathbf{y}(n)$ belongs to $\mathcal{L}_{n-e}$ when $n \geq k + d$. It follows that $\mathcal{O}_{n-e} \subseteq \mathcal{O}_{n-e+1}$ for $n \geq k + d$, and so $\mathcal{O}_{k+d-e}$ is a subspace of $\mathcal{O}_n$ for every $n \geq k + d - e$. Since $\dim \mathcal{O}_n \geq 1$ for all $n \geq 0$, there is a non-zero vector $\mathbf{c} \in \mathcal{O}_{k+d-e} \subseteq \cap_{n \geq k+d-e} \mathcal{O}_n$. Then

$$\sum_{i=1}^{e+1} c_i y_i(n) = 0\,, \qquad \text{for } n \geq k + d - e\,. \tag{5.5}$$

Applying φ to (5.5) gives

$$\sum_{i=1}^{e+1} c_i x_i = 0\,.$$

Since $\mathbf{c} \neq \mathbf{0}$ this means that $x_1, x_2, \ldots, x_{e+1}$ are linearly dependent over F.

Case 2: a_{d-e} is a unit. Then there exists an $M \in \mathbb{N}$ such that $b_{d-e}(n) \neq 0$ for all $n \geq M$. Let $k = \max\{N, M\}$. Then by (5.4), for all $n \geq k$,

$$y_i(n+d-e) = -\sum_{j=d-e+1}^{d} b_j(n)/b_{d-e}(n) y_i(n+j)\,, \qquad \text{for } i = 1, 2, \ldots, e+1.$$

Hence $\mathbf{y}(n)$ belongs to $\mathcal{L}_{n+1}$ when $n \geq k+d-e$. It follows that $\mathcal{O}_{n+1} \subseteq \mathcal{O}_n$ for $n \geq k+d-e$, so that $\mathcal{O}_{k+d-e} \supseteq \mathcal{O}_{k+d-e+1} \supseteq \ldots$ is a decreasing chain of finite-dimensional linear subspaces. Every proper inclusion corresponds to a decrease in dimension, so there are only finitely many proper inclusions in the chain. Therefore there is an $m \in \mathbb{N}$ such that $\mathcal{O}_n = \mathcal{O}_m$ for all $n \geq m$, hence $\mathcal{O}_m$ is a subspace of $\mathcal{O}_n$ for every $n \geq k+d-e$. Since $\dim \mathcal{O}_n \geq 1$ for all $n \geq 0$, there is a non-zero vector $\mathbf{c} \in \mathcal{O}_m$. Then (5.5) holds again, and we finish the proof as in Case 1. □

Example 8 To show that the condition in Lemma 9 cannot in general be weakened consider the equation

$$a_1 \tau x + a_0 x = 0 \tag{5.6}$$

Let $a_0 = \varphi(a)$ and $a_1 = \tau a_0$ where $a(n) = 1$ for n even, and $a(n) = 0$ for n odd. Clearly, a_0 and a_1 are nonunits. Let $x = \varphi y$. Then (5.6) is equivalent to the requirement that from some point on, the even-numbered terms of y are equal to zero. It is not difficult to see that the linear subspace of $\mathcal{C}$ consisting of all φy such that y satisfies this requirement has infinite dimension although the effective degree of the operator is 1.

Lemma 10 *Let L be a linear difference operator of the form* (5.3) *of effective degree e, and such that both a_d and a_{d-e} are units of $\mathcal{C}$. Then* $\dim \operatorname{Ker} L \geq e$.

Proof: Let $b_0, b_1, \ldots, b_d \in F^{\mathbb{N}}$ be such that $a_i = \varphi b_i$, for $i = 0, 1, \ldots, d$. Since both a_d and a_{d-e} are units of $\mathcal{C}$ there exists an $N \in \mathbb{N}$ such that $b_d(n), b_{d-e}(n) \neq 0$ for all $n \geq N$. Let $K = N + d - e$, and let $\mathbf{v}(0), \mathbf{v}(1), \ldots, \mathbf{v}(e-1)$ be linearly independent vectors from F^e. Then there exist sequences $y_1, y_2, \ldots, y_e \in F^{\mathbb{N}}$ such that

$$1.\quad y_i(K+k) = v_i(k), \qquad \text{for } k = 0, 1, \ldots, e-1, \tag{5.7}$$

$$2.\quad y_i(k) = -\sum_{j=d-e}^{d-1} \frac{b_j(k-d)}{b_d(k-d)} y_i(k-d+j), \qquad \text{for } k \geq K+e, \tag{5.8}$$

for $i = 1, 2, \ldots, e$. Multiplying (5.8) by $b_d(k-d)$ and setting $n = k-d$ gives

$$\sum_{j=d-e}^{d} b_j(n) \sigma^j y_i(n) = 0, \qquad \text{for } i = 1, 2, \ldots, e, \tag{5.9}$$

when $n \geq N$. Let $x_i = \varphi y_i$, for $i = 1, 2, \ldots, e$. Then an application of φ to (5.9) shows that $L x_i = 0$ for $i = 1, 2, \ldots, e$. We claim that $x_1, x_2, \ldots, x_e$ are linearly independent.

Assume not. For all $n \geq 0$, let $\mathbf{y}(n) \in F^e$ be the vector with components $y_1(n), y_2(n), \ldots, y_e(n)$. Denote by $\mathcal{L}_n$ the linear span of $\mathbf{y}(n), \mathbf{y}(n+1), \ldots, \mathbf{y}(n+e-1)$, and let $\mathcal{O}_n := \{\mathbf{u};\ \sum_{i=1}^{e+1} u_i v_i = 0, \text{forall} \mathbf{v} \in \mathcal{L}_n\}$. By the assumption there exists a non-zero vector $\mathbf{c} \in F^e$ and an $M \in \mathbb{N}$ such that $\mathbf{c} \in \mathcal{O}_n$ for all $n \geq M$. Dividing (5.9) by $b_{d-e}(n)$ gives

$$y_i(n+d-e) = -\sum_{j=d-e+1}^{d} \frac{b_j(n)}{b_{d-e}(n)} y_i(n+j), \qquad \text{for } i = 1, 2, \ldots, e,$$

when $n \geq N$. Hence $\mathbf{y}(n) \in \mathcal{L}_{n+1}$ when $n \geq K$. Therefore $\mathcal{O}_{n+1} \subseteq \mathcal{O}_n$ for $n \geq K$, and so

$$\mathbf{c} \in \mathcal{O}_n, \qquad \text{for all } n \geq K. \tag{5.10}$$

But by (5.7), $\mathbf{y}(K+k) = \mathbf{v}(k)$ for $k = 0, 1, \ldots, e-1$, therefore $\dim \mathcal{L}_K = e$ and $\dim \mathcal{O}_K = 0$, in contradiction with (5.10). This proves the claim. □

Example 9 To show that the condition in Lemma 10 cannot in general be weakened consider the equations

$$a_0 \tau x + x = 0$$

and

$$\tau x + a_0 x = 0$$

where a_0 is as in Example 8. It is not difficult to see that if $x \in C$ satisfies either of these equations then $x = 0$, and so in both cases the dimension of the solution space is 0 while the effective degree of the operator is 1.

Definition 14 A linear difference operator of the form (5.3) of degree d and effective degree e is called *regular* if both a_d and a_{d-e} are units of C. □

Theorem 10 *Let L be a regular linear difference operator on C of effective degree e. Then* Ker$L$ *has dimension e.*

Proof: This follows immediately from Lemmas 9 and 10. □

5.5 Hypergeometric sequences

Proposition 3 *Let $x \in C$ be a non-zero solution of the first-order equation $\tau x = ax$. Then x is a unit.*

Proof: Let $x = \varphi(y)$ and $a = \varphi(b)$. If x is not a unit then y contains infinitely many zero terms. Since x is not zero y also contains infinitely many non-zero terms. This implies that for every N we can find k and m such that $N < m < k$, $y(m) = 0$, and $y(k) \neq 0$. But from the equation which x satisfies it follows that, for large enough N, $y(k) = b(k-1)b(k-2)\cdots b(m)y(m) = 0$, a contradiction. □

Definition 15 A non-zero sequence $x \in C$ is *hypergeometric over F* if it satisfies an equation of the form $\tau x = ax$ where $a \in \mathcal{R}$ is a rational sequence.

Note that a in this definition has to be non-zero.

By Proposition 3, hypergeometric sequences are units. We shall denote the set of all hypergeometric sequences over F by H. Obviously $\mathcal{R} \setminus \{0\} \subseteq H$. If $\tau x = ax$ and $\tau y = by$, then $\tau(xy) = (ab)(xy)$ and $\tau x^{-1} = a^{-1}x^{-1}$, so H is a group under multiplication. Finite sums of hypergeometric sequences form an algebra over the field of rational sequences $\mathcal{R}$.

Theorem 11 *Let S be a set of hypergeometric sequences. If S is linearly dependent over $\mathcal{R}$ then S contains two elements which are linearly dependent over $\mathcal{R}$.*

Proof: Let $S \subseteq H$ be linearly dependent over $\mathcal{R}$. Let $S' = \{a_0, a_1, \ldots, a_m\}$ be a minimal subset of S which is linearly dependent over $\mathcal{R}$. Note that a hypergeometric sequence is non-zero by definition, therefore $m \geq 1$. Then there exist non-zero elements $r_0, r_1, \ldots, r_m \in \mathcal{R}$ such that

$$\sum_{i=0}^{m} r_i a_i = 0. \qquad (5.11)$$

Let $b_i := r_i a_i$, for $i = 0,1,\ldots,m$. For $j = 0,1,\ldots,m-1$ apply τ^j to (5.11) and divide the resulting equations by b_m. This gives

$$\sum_{i=0}^{m} \frac{\tau^j b_i}{b_m} = 0, \quad \text{for } j = 0,1,\ldots,m-1$$

which can be rewritten as

$$\sum_{i=0}^{m-1} \frac{\tau^j b_i}{b_i}\frac{b_i}{b_m} = -\frac{\tau^j b_m}{b_m}, \quad \text{for } j = 0,1,\ldots,m-1. \tag{5.12}$$

This is a system of m linear algebraic equations for the m unknown ratios b_i/b_m, $0 \le i \le m$, with coefficients and right-hand sides in $\mathcal{R}$. The determinant of this system is

$$D := \det\left[\frac{\tau^j b_i}{b_i}\right]_{i,j=0}^{m-1} = \frac{\det\left[\tau^j b_i\right]_{i,j=0}^{m-1}}{\prod_{i=0}^{m-1} b_i}.$$

If $D = 0$ then one can show, by developing a suitable subdeterminant with respect to the first column, that $b_0, b_1, \ldots, b_{m-1}$ are linearly dependent over $\mathcal{R}$. But this contradicts the minimality of S'. Therefore $D \neq 0$ and (5.12) has a unique non-zero solution in $\mathcal{R}$. Hence there is an i, $0 \le i \le m-1$, and an $r \in \mathcal{R}$ such that $b_i = r b_m$, proving the theorem. □

Theorem 12 *Let $\{\xi_1, \xi_2, \ldots, \xi_d\}$ be a set of hypergeometric sequences which are linearly independent over F. Then there exists a rational linear difference operator on $\mathcal{C}$ of degree and effective degree d such that $\{\xi_1, \xi_2, \ldots, \xi_d\}$ are its fundamental solutions.*

Proof: Consider the equation $\mathrm{Cas}(a, \xi_1, \xi_2, \ldots, \xi_d) = 0$ where $\mathrm{Cas}(x_0, x_1, \ldots, x_n) = \det(\tau^j x_i)_{i,j=0}^{n}$ is the Casoratian determinant. After dividing it by the product $\xi_1 \xi_2 \cdots \xi_d$ we obtain an equation for a with rational coefficients satisfied by $\{\xi_1, \xi_2, \ldots, \xi_d\}$. The left-hand side of this equation is therefore a rational difference operator L of effective degree $e \le d$. Since $\{\xi_1, \xi_2, \ldots, \xi_d\} \in \mathrm{Ker} L$ it follows from Lemma 9 that in fact $e = d$. □

5.6 The Galois group of a rational operator

Let L be a rational linear difference operator on $\mathcal{C}$ of effective degree e. By Theorem 10, $\mathrm{Ker}\, L$ has dimension e. Let $\xi_1, \xi_2, \ldots, \xi_e$ be a basis for $\mathrm{Ker}\, L$. Denote by $\mathcal{G}(\mathcal{R}\{\xi\}/\mathcal{R})$ the group of those difference automorphisms of $\mathcal{R}\{\xi_1, \xi_2, \ldots, \xi_e\}$ which fix $\mathcal{R}$. If $Lx = 0$ and $\alpha \in \mathcal{G}(\mathcal{R}\{\xi\}/\mathcal{R})$ then $L\alpha x = 0$ as well. Therefore for every $\alpha \in \mathcal{G}(\mathcal{R}\{\xi\}/\mathcal{R})$ there exists a unique matrix $A \in \mathbf{GL}_e(F)$, with elements a_{ij}, such that

$$\alpha \xi_i = \sum_{j=1}^{e} a_{ij} \xi_j, \quad \text{for } i = 1,2,\ldots,e. \tag{5.13}$$

This defines a group monomorphism $M : \mathcal{G}(\mathcal{R}\{\xi\}/\mathcal{R}) \to \mathbf{GL}_e(F)$. The matrix group $M(\mathcal{G}(\mathcal{R}\{\xi\}/\mathcal{R}))$ is called the *Galois group* of $\mathcal{R}\{\xi_1, \xi_2, \ldots, \xi_e\}$ over $\mathcal{R}$, and will be denoted by $\mathrm{G}(\mathcal{R}\{\xi\}/\mathcal{R})$.

The Galois group depends on the choice of basis for $\mathrm{Ker}\, L$. But it is clear that if $\eta_1, \eta_2, \ldots, \eta_e$ is another basis for $\mathrm{Ker}\, L$ then there exists a nonsingular matrix $P \in \mathbf{GL}_e(F)$ such that $\mathrm{G}(\mathcal{R}\{\eta\}/\mathcal{R}) = P^{-1}\, \mathrm{G}(\mathcal{R}\{\xi\}/\mathcal{R})\, P$. So the Galois group of an operator is determined up to conjugacy.

Theorem 13 *Let L be a rational linear difference operator on C of degree and effective degree d. Let $\xi_1, \xi_2, \ldots, \xi_d$ be a basis for* Ker L*. The Galois group $\mathrm{G}(\mathcal{R}\{\xi\}/\mathcal{R})$ is an algebraic matrix group.*

The following proof is based on the proof of the analogous theorem for differential operators given in [Kov86], which in turn is based on the proof given in [Kap57]. However, there seems to be a flaw in the second part of the proof in [Kov86]. Since we also wanted to avoid using transcendency degrees as is done in [Kap57] this part of the proof follows a different path.

Proof: Let $G = \mathrm{G}(\mathcal{R}\{\xi\}/\mathcal{R})$, and $\mathcal{S} = \mathcal{R}\{\xi_1, \xi_2, \ldots, \xi_d\}$. Because the ξ_i satisfy the equation $Lx = 0$ it is clear that $\mathcal{S}$ equals the extension ring of $\mathcal{R}$ obtained by adjoining $\tau^{j-1}\xi_i$, for $1 \le i, j \le d$. Let $F[z_{ij}]$ be the ring of polynomials over F in the indeterminates z_{ij}, $1 \le i, j \le d$. We need to exhibit a set of polynomials $\mathcal{E} \subseteq F[z_{ij}]$ such that $A \in G$ if and only if $p[a_{ij}] = 0$ for every $p \in \mathcal{E}$. Let $\mathcal{R}[x_{ij}]$ be the ring of polynomials over $\mathcal{R}$ in the indeterminates x_{ij}, $1 \le i, j \le d$. Let $\mathcal{S}[z_{ij}]$ be the ring of polynomials over $\mathcal{S}$ in the indeterminates z_{ij}, $1 \le i, j \le d$. By the universal property of polynomial rings, there exists a unique ring homomorphism

$$f : \mathcal{R}[x_{ij}] \to \mathcal{S}[z_{ij}]$$

such that $f|_{\mathcal{R}} = \mathrm{id}_{\mathcal{R}}$ and $f(x_{ij}) = \sum_{k=1}^{d} z_{ik}\tau^{j-1}\xi_k$, for $1 \le i, j \le d$. Let $(w_\lambda)_{\lambda\in\mathcal{I}}$ be a vector space basis for $\mathcal{S}$ over F. Then the elements of $\mathcal{S}[z_{ij}]$ can be written as finite linear combinations of w_λ's with coefficients in $F[z_{ij}]$. In particular, for every $q \in \mathcal{R}[x_{ij}]$ and $\lambda \in \mathcal{I}$ there is a unique polynomial $p_{q\lambda} \in F[z_{ij}]$ such that

$$f(q) = \sum_{\lambda\in\mathcal{I}} p_{q\lambda}[z_{ij}]\, w_\lambda\,. \tag{5.14}$$

The sum on the right-hand side is finite for every $q \in \mathcal{R}[x_{ij}]$.

Let Z be the matrix of indeterminates $[z_{ij}]_{i,j=1}^{d}$. To each polynomial $p \in F[z_{ij}]$ we assign a polynomial $\overline{p} \in F[z_{ij}]$ by

$$\overline{p}[z_{ij}] = D^k p[D_{ji}/D]$$

where $k = \deg p$, $D = \det Z$, and D_{ij} is the cofactor of the (i,j)-th element of the matrix Z.

By the universal property of polynomial rings, there exists a unique ring homomorphism

$$\psi : \mathcal{R}[x_{ij}] \to \mathcal{S}$$

such that $\psi|_{\mathcal{R}} = \mathrm{id}_{\mathcal{R}}$ and $\psi(x_{ij}) = \tau^{j-1}\xi_i$, for $1 \le i, j \le d$. The kernel of ψ is the *ideal of algebraic relations* among the solutions of $Lx = 0$ and their transforms. Define

$$\begin{aligned} \mathcal{E} &= \{p_{q\lambda};\ q \in \mathrm{Ker}\ \psi, \lambda \in \mathcal{I}\} \\ \overline{\mathcal{E}} &= \{\overline{p};\ p \in \mathcal{E}\} \end{aligned}$$

Let $A \in \mathbf{GL}_d(F)$. We claim that $A \in G$ if and only if $p[a_{ij}] = 0$ for every $p \in \mathcal{E} \cup \overline{\mathcal{E}}$.

To prove this claim, let

$$\chi_A : \mathcal{S}[z_{ij}] \to \mathcal{S}$$

be the unique ring homomorphism such that $\chi_A|_{\mathcal{S}} = \mathrm{id}_{\mathcal{S}}$ and $\chi_A(z_{ij}) = a_{ij}$, for $1 \le i, j \le d$. It exists, once again, by the universal property of polynomial rings.

Assume first that $A \in G$. Let α be the corresponding difference automorphism of $\mathcal{S}$ such that $A = M\alpha$, as in (5.13). Both $\alpha\psi$ and $\chi_A f$ are ring homomorphisms from $\mathcal{R}[x_{ij}]$ into $\mathcal{S}$. Since they agree on $\mathcal{R}$ and

$$\alpha\psi(x_{ij}) = \alpha\tau^{j-1}\xi_i = \sum_{k=1}^{d} a_{ik}\tau^{j-1}\xi_k = \chi_A(\sum_{k=1}^{d} z_{ik}\tau^{j-1}\xi_k) = \chi_A f(x_{ij}),$$

for $1 \le i, j \le d$, it follows that

$$\alpha\psi = \chi_A f \tag{5.15}$$

and the diagram

$$\begin{tikzcd} \mathcal{R}[x_{ij}] \arrow[r, "\psi"] \arrow[d, "f"'] & \mathcal{S} \arrow[d, "\alpha"] \\ \mathcal{S}[z_{ij}] \arrow[r, "\chi_A"] & \mathcal{S} \end{tikzcd}$$

commutes. If $q \in \operatorname{Ker} \psi$ then by (5.14) and (5.15),

$$\sum_{\lambda \in \mathcal{I}} p_{q\lambda}[a_{ij}]\, w_\lambda = \chi_A(\sum_{\lambda \in \mathcal{I}} p_{q\lambda}[z_{ij}]\, w_\lambda) = \chi_A f(q) = \alpha\psi(q) = 0,$$

and so $p_{q\lambda}[a_{ij}] = 0$ for every $q \in \operatorname{Ker} \psi$ and $\lambda \in \mathcal{I}$. Hence

$$p[a_{ij}] = 0, \qquad \text{for every } p \in \mathcal{E}.$$

Let $A^{-1} = [\overline{a}_{ij}]_{i,j=1}^{d}$. Repeating the same argument with A^{-1} in place of A we obtain $p[\overline{a}_{ij}] = 0$, for every $p \in \mathcal{E}$. By definition of $\overline{p}$,

$$p[\overline{a}_{ij}] = p[A_{ji}/\det A] = \overline{p}[a_{ij}]/(\det A)^{\deg p}, \tag{5.16}$$

where A_{ij} is the cofactor of the (i, j)-th element of A. Since $\det A \neq 0$, this implies that

$$\overline{p}[a_{ij}] = 0, \qquad \text{for every } \overline{p} \in \overline{\mathcal{E}}.$$

Conversely, assume that $p[a_{ij}] = 0$ for every $p \in \mathcal{E} \cup \overline{\mathcal{E}}$. Let $q \in \operatorname{Ker} \psi$. Then $\chi_A f(q) = \sum_{\lambda \in \mathcal{I}} p_{q\lambda}[a_{ij}]\, w_\lambda = 0$, so $\operatorname{Ker} \psi \subseteq \operatorname{Ker} \chi_A f$. Using (5.16), we obtain $\operatorname{Ker} \psi \subseteq \operatorname{Ker} \chi_{A^{-1}} f$ in the same way. By the Isomorphism Theorem of universal algebra, there exist unique ring endomorphisms α and β of $\mathcal{S}$ such that

$$\alpha\psi = \chi_A f, \tag{5.17}$$
$$\beta\psi = \chi_{A^{-1}} f. \tag{5.18}$$

Then

$$\alpha\,\tau^{j-1}\xi_i = \alpha\psi(x_{ij}) = \chi_A f(x_{ij}) = \sum_{k=1}^{d} a_{ik}\tau^{j-1}\xi_k, \tag{5.19}$$

and similarly,

$$\beta\,\tau^{j-1}\xi_i = \sum_{k=1}^{d} \overline{a}_{ik}\tau^{j-1}\xi_k, \tag{5.20}$$

for $1 \le i, j \le d$. We want to show that α is a difference automorphism of $\mathcal{S}$ which fixes $\mathcal{R}$ and such that $M\alpha = A$.

Using (5.19) and (5.20) we find that

$$\begin{aligned}\alpha\beta\,\tau^{j-1}\xi_i &= \alpha\sum_{k=1}^{d}\overline{a}_{ik}\tau^{j-1}\xi_k = \sum_{k=1}^{d}\overline{a}_{ik}\sum_{r=1}^{d}a_{kr}\tau^{j-1}\xi_r \\ &= \sum_{r=1}^{d}\tau^{j-1}\xi_r\sum_{k=1}^{d}\overline{a}_{ik}a_{kr} = \tau^{j-1}\xi_i\,, \end{aligned} \tag{5.21}$$

and similarly, that

$$\beta\alpha\,\tau^{j-1}\xi_i = \tau^{j-1}\xi_i\,, \tag{5.22}$$

for $1 \le i,j \le d$. Since ψ, f, χ_A, and χ_A^{-1} all fix $\mathcal{R}$, so do $\alpha,\beta,\alpha\beta$, and $\beta\alpha$. This together with (5.21) and (5.22) implies that

$$\alpha\beta = \beta\alpha = \mathrm{id}_{\mathcal{S}}\,.$$

Therefore α is an automorphism of $\mathcal{S}$ which fixes $\mathcal{R}$. If $r \in \mathcal{R}$, then $\tau r \in \mathcal{R}$, so $\alpha\tau r = \tau r = \tau\alpha r$ since α fixes $\mathcal{R}$. Furthermore, by (5.19)

$$\alpha\tau(\tau^{j-1}\xi_i) = \sum_{k=1}^{d}a_{ik}\,\tau^{j}\xi_k = \tau\alpha(\tau^{j-1}\xi_i)\,,$$

for $1 \le i,j \le d$. This proves that α commutes with τ, and is thus a difference automorphism of $\mathcal{S}$. From (5.19) it follows that $M\alpha = A$. So $A \in G$. □

Example 10 Let $\{\xi_1,\xi_2,\ldots,\xi_d\}$ be a set of fundamental solutions, $\alpha \in \mathrm{G}(\mathcal{R}\{\xi\}/\mathcal{R})$ an automorphism of $\mathcal{R}\{\xi\}$ which fixes $\mathcal{R}$, and $(a_{ij})_{i,j=1}^{d}$ the corresponding matrix in the Galois group $\mathrm{G}(\mathcal{R}\{\xi\}/\mathcal{R})$.

If $\xi_k \in \mathcal{R}$ then $\alpha\xi_k = \xi_k$, therefore $a_{kj} = \delta_{kj}$, for $j = 1,2,\ldots,d$.

If $\xi_k \in \mathcal{H}$ then there is an $r \in \mathcal{R}$ such that $\tau\xi_k = r\xi_k$. Applying α to this equation we see that $\tau\alpha\xi_k = r\alpha\xi_k$, hence $\alpha\xi_k = c\xi_k$ for some non-zero $c \in F$. Therefore $a_{kj} = 0$, for $j = 1,2,\ldots,d$, $j \neq k$.

If $\xi_k \in \mathcal{H}$ and $\xi_m = r\xi_k$ for some non-zero $r \in \mathcal{R}$ then by the previous paragraph, $\alpha\xi_k = c\xi_k$ for some $c \in F$, so $\alpha\xi_m = r\alpha\xi_k = cr\xi_k = c\xi_m$. Therefore $a_{kk} = a_{mm}$. More generally, if $\xi_{i_1},\xi_{i_2},\ldots,\xi_{i_k} \in \mathcal{H}$ and $r\xi_{i_1}^{n_1}\xi_{i_2}^{n_2}\cdots\xi_{i_k}^{n_k} = 1$ for some $n_1,n_2,\ldots n_k \in \mathbb{Z}$ and $r \in \mathcal{R}$ then $a_{i_1,i_1}^{n_1}a_{i_2,i_2}^{n_2}\cdots a_{i_k,i_k}^{n_k} = 1$ as well.

Chapter 6

Conclusions

This thesis has approached the problem of finding symbolic solutions of difference equations from four different directions.

First, it has shown that the method of generating functions is sufficiently versatile so that a comprehensive package for solving difference equations could be built around it. A prototype implementation in *Mathematica*$^{\text{TM}}$ has been carried out successfully. However, except for small classes of equations such as ordinary difference equations with constant coefficients, the method of generating functions should be considered as a heuristic, and should therefore be complemented by algorithms for other classes of equations. Procedures for related problems such as evaluation of sums and products in closed form, the operations of difference calculus, simplification of factorial expressions, and proving combinatorial identities should also be implemented.

Second, considering that ordinary difference equations with constant coefficients always give rise to rational generating functions and are completely understood it is surprising that it is not known if the generating functions defined by partial difference equations with constant coefficients are algebraic. The increase in complexity is due to the geometry of initial conditions. We have proved an existence and uniqueness theorem, and shown that under reasonable assumptons the generating function is analytic.

Turning to ordinary difference equations with polynomial coefficients, there is again a sharp increase in complexity of solutions in comparison to equations with constant coefficients. Explicit solutions for a general equation of this type - even for second-order equations - are not known. Therefore it s important to construct algorithms which decide existence of solutions of certain restricted types. We have developed algorithms for finding polynomial and hypergeometric solutions of such equations. In conjunction with an algorithm of Zeilberger [Zeib], the latter algorithm solves the long-standing problem of deciding whether a definite hypergeometric sum is again hypergeometric.

Finally, on a more theoretical level, there has been success in using differential algebra and Galois theory of differential equations to find Liouvillian and elementary solutions of differential equations. In the analogous approach to difference equations there are obstacles which do not appear with differential equations. We have tried to avoid these obstacles by using difference rings of sequences in place of difference fields of functions. In this setting we have proved that the Galois group of a linear difference operator with rational coefficients is an algebraic group.

This work suggests a number of problems for further research:

1. Are there partial difference equations with constant coefficients and such that the corresponding generating functions are not algebraic? If not, how can one compute the minimal polynomial defining the generating function? Can this be automated? In Chapter 3, we assumed that initial conditions are given on the boundary of the first orthant. What other geometric configurations could be handled?

2. Many recurrences with polynomial coefficients surfacing in combinatorial enumeration problems have solutions which have the form of a definite hypergeometric sum. Is there an algorithm for deciding existence of solutions that are single or multiple definite hypergeometric sums?

3. Are difference algebra extensions of $\mathcal{R}$ by adjoining solutions of a rational difference operator on $\mathcal{C}$ normal? Is there a Galois correspondence between the normal subgroups of the Galois group and the intermediate algebras in such an extension? What periodicity properties do the nonunit solutions of rational difference equations possess? Is there an algorithm for finding all Liouvillian solutions of such equations?

Appendix A

The program

```
(*** RSolve.m ***)

     (* alpha test version *)

     (* last modified: April 25, 1991 *)

     (* Copyright C 1990 by Marko Petkovsek *)


     BeginPackage["RSolve`"]


     RSolve::usage = "RSolve[{eqn1, eqn2, ..}, {a1[n], a2[n], ..}, n,
     opts] solves the recurrence equations eqn1, eqn2, ..
     for the sequences a1[n], a2[n], ... If there is a single sequence or
     equation it need not be given in a list. Equations can either be
     recurrences or initial/boundary conditions. An equation may have
     the form eqn /; cond where cond is an inequality specifying the range
     of values of n for which eqn is valid; when no condition is given
     it is assumed to be n >= 0. Alias: RS."

     PowerSum::usage = "PowerSum[expr, {z, n, n0:0}] computes
     Sum[expr z^n, {n, n0, Infinity}]. If a[n] is a sequence, the name
     Gf[a][z] is used by PowerSum to denote Sum[a[n] z^n, {n, 0, Infinity}].
     PowerSum is threaded over lists and equations. Alias: PS."

     ExponentialPowerSum::usage = "ExponentialPowerSum[expr, {z, n, n0:0}]
     computes Sum[expr z^(n-n0) / (n-n0)!, {n, n0, Infinity}]. If a[n] is a
     sequence, the name EGf[a][z] is used by ExponentialPowerSum to denote
     Sum[a[n] z^n / n!, {n, 0, Infinity}]. ExponentialPowerSum is threaded
     over lists and equations. Alias: EPS."

     GeneratingFunction::usage = "GeneratingFunction[{eqn1, eqn2, ..},
     {a1[n], a2[n], ..}, n, z, opts] computes the ordinary generating
     functions Sum[ai[n] z^n, {n, si, Infinity}] for the sequences a1[n],
     a2[n], .. defined by the equations eqn1, eqn2, ... Here si denotes the
     least value of n such that ai[n] appears in the equations. An equation
     may have the form eqn /; cond where cond is an inequality specifying
     the range of values of n for which eqn is valid; when no condition
     is given it is assumed to be n >= 0. Alias: GF."

     ExponentialGeneratingFunction::usage = "ExponentialGeneratingFunction[
     {eqn1, eqn2, ..}, {a1[n], a2[n], ..}, n, z] computes the exponential
     generating functions Sum[ai[n + si] z^n / n!, {n, 0, Infinity}] for the
     sequences a1[n], a2[n], .. defined by the equations eqn1, eqn2, ... Here
     si denotes the least value of n such that ai[n] appears in the equations.
     An equation may have the form eqn /; cond where cond is an inequality
     specifying the range of values of n for which eqn is valid; when no
     condition is given it is assumed to be n >= 0. Alias: EGF."

     Gf::usage = "Gf[a][z] is used by PowerSum to denote
     Sum[a[n] z^n, {n, 0, Infinity}]."

     EGf::usage = "EGf[a][z] is used by ExponentialPowerSum to denote
     Sum[a[n] z^n / n!, {n, 0, Infinity}]."

     HSolve::usage = "HSolve[eq, f, z] returns a hypergeometric formal
```

```
series solution to differential equation eq with unknown function
f[z], or {} when it determines that there is no series solution.
It returns Fail when it is unable to do either."

HypergeometricF::usage = "HypergeometricF[{a1, a2, ..}, {b1, b2, ..}, z]
is the generalized hypergeometric function with upper parameters
a1, a2,.., lower parameters b1, b2,.., and argument z."

K::usage = "K is the head used for summation indices by SeriesTerm."

SeriesTerm::usage = "SeriesTerm[expr, {z, a, n}, opts]
returns the n-th coefficient of the Laurent series expansion of expr
around z = a, for symbolic n. Alias: ST."

SymbolicMod::usage = "SymbolicMod[n, k] is equal to Mod[n, k] when n
is a number, and is left unevaluated otherwise."

RealQ::usage = "'RealQ[x] = True' serves to declare x to be real."

ReP::usage = "ReP[x] gives the real part of x."

ImP::usage = "ImP[x] gives the imaginary part of x."

ConjugateQ::usage = "ConjugateQ[x, y] tests whether x and y are complex
conjugates."

SimplifySum::usage = "SimplifySum is a list of rules for simplification
of sums."

SimplifyTrig::usage = "SimplifyTrig is a list of rules which put
trigonometric expressions into canonical form."

SimplifyComplexE1::usage = "SimplifyComplexE1 is one of lists of rules
used to replace sums of conjugate complex quantities by their double
real parts."

SimplifyComplexE2::usage = "SimplifyComplexE2 is one of lists of rules
used to replace sums of conjugate complex quantities by their double
real parts."

SimplifyComplex2::usage = "SimplifyComplex2 is one of lists of rules
used to replace sums of conjugate complex quantities by their double
real parts."

SimplifyComplex3::usage = "SimplifyComplex3 is one of lists of rules
used to replace sums of conjugate complex quantities by their double
real parts."

SimplifyComplex4::usage = "SimplifyComplex4 is one of lists of rules
used to replace sums of conjugate complex quantities by their double
real parts."

SimplifyComplexN::usage = "SimplifyComplexN is one of lists of rules
used to replace sums of conjugate complex quantities by their double
real parts."

FactorialSimplify::usage = "FactorialSimplify[expr] simplifies
occurrences of Factorial[] inside expr."

PartialFractions::usage = "PartialFractions[f, z, opts] gives a partial
fraction decomposition of f[z] over the complex functions."

ArgPi::usage = "When possible ArgPi[z] expresses the complex number z
in the form Abs[z] E^(r Pi I) where r is a rational number."

Entails::usage = "Entails[a, b] determines whether a implies b where
a and b are Boolean combinations of equalities and/or inequalities among
rational functions of a single integer variable."

FirstPos::usage = "FirstPos[l, pattern] returns the position of
the first argument of l which matches pattern, or Null if there is
no such element."

FreeL::usage = "FreeL[expr, list] returns True if none of the
elements of list appears in expr, False otherwise."

Info::usage = "Info[n] is information about n, expressed as a
Boolean combination of inequalities. It should be associated with n."

ISolve::usage = "ISolve[expr] solves a Boolean combination of equalities
and/or inequalities in a single integer variable. The answer is given
as a disjunction of disjoint inclusive inequalities."
```

```
LeadingCoef::usage = "LeadingCoef[poly, x] returns the leading
coefficient of the polynomial poly[x]."

ListComplement::usage = "ListComplement[l1_List, l2_List] returns
the multiset-difference of l1 and l2."

MakeList::usage = "MakeList[expr, head:List] returns the list of
arguments of expr if the head of expr matches the given one, or else
it returns {expr}."

MakeTrinomial::usage = "MakeTrinomial[expr_] tries to write expr
in the form Multinomial[a, b, c]."

PatternList::usage = "PatternList[expr, pattern] returns the list of
all subexpressions of expr which match the given pattern, provided
that expr does not contain Rule[]."

PoleMultiplicity::usage = "PoleMultiplicity[f, {z, a}] computes
the multiplicity of the pole of f at z = a."

Reset::usage = "Reset[recur, conds, unknowns, startValues, s0] returns
the list {recur, conds} with starting values of the index reset so
that it starts with s0 rather than with startValues, for all unknown
sequences."

SafeFirst::usage = "SafeFirst[l] returns the first argument of l,
or Null if the length of l is zero."

SafeSeries::usage = "SafeSeries[f, {z, a, n}] overcomes a bug in
Series[] which fails when n is less than the degree of zero of the
denominator of f at a."

TTQ::usage = "TTQ[cond] returns True if it determines that the
information given about the variables implies that cond is true,
and False otherwise."

UserSymbols::usage = "UserSymbols[expr] returns a list of maximal
subexpressions of expr with the property that their head contains
symbols defined in a context different from System`. "

When::usage = "When[cond, a, b] returns a if it determines that the
information given about the variables implies that cond is true,
b if it determines that the information given about the variables
implies that cond is false, and If[cond, a, b] otherwise."

Methods::usage = "Methods is an optional parameter for RSolve.
It specifies which methods and in what order are to be used by RSolve.
Default: Methods -> {MethodGF, MethodEGF}. If a single method is given
it need not be enclosed in a list."

MethodGF::usage = "MethodGF is a possible value for Methods. It denotes
the method of ordinary generating functions."

MethodEGF::usage = "MethodEGF is a possible value for Methods. It denotes
the method of exponential generating functions."

PrecisionHS::usage = "PrecisionHS is an optional parameter for RSolve,
GeneratingFunction, ExponentialGeneratingFunction, and HSolve. It
specifies the precision with which the roots of polynomials are to be
computed by HSolve. When successful, the result of HSolve is a
generalized hypergeometric function, and these roots are its parameters.
Possible values: PrecisionHS -> n, PrecisionHS -> Infinity,
PrecisionHS -> Automatic (default). Infinity means exact roots, but
those roots which cannot be found by Solve[] are computed numerically.
Automatic means that only those roots which Solve[] can find without
recourse to general formulas for solving cubics and quartics are found
exactly, while others are computed numerically."

PrecisionST::usage = "PrecisionST is an optional parameter for RSolve
and SeriesTerm. It specifies the precision with which the roots of
denominators of rational functions to be expanded into power series
are to be computed by SeriesTerm. Possible values: PrecisionST -> n,
PrecisionST -> Infinity, PrecisionST -> Automatic (default). Infinity
means exact roots, but those roots which cannot be found by Solve[]
are computed numerically. Automatic means that only those roots which
Solve[] can find without recourse to general formulas for solving cubics
and quartics are found exactly, while others are computed numerically."

UseApart::usage = "UseApart is an optional parameter for RSolve
and SeriesTerm. It specifies whether or not Apart should be used
before attempting expansion into power series, and in what form.
Possible values: UseApart -> None (don't use Apart at all),
UseApart -> Automatic (default; use Apart without second argument),
UseApart -> All (use Apart with second argument). By default, MethodGF
uses UseApart -> Automatic and MethodEGF uses UseApart -> None.
UseApart -> All is useful when the function to be expanded contains
parameters."

UseMod::usage = "UseMod is an optional parameter for RSolve and
SeriesTerm. Possible values: UseMod -> True (default; use EvenQ, OddQ
and/or SymbolicMod to express the result), UseMod -> False (the
opposite)."

MakeReal::usage = "MakeReal is an optional parameter for RSolve and
SeriesTerm. Possible values: MakeReal -> True (default; replace pairs
of conjugate complex quantities by their doble real parts),
MakeReal -> False (the opposite)."

SpecialFunctions::usage = "SpecialFunctions is an optional parameter
for RSolve and SeriesTerm. Possible values: SpecialFunctions -> True
(default; use special functions such as Legendre polynomials to express
the result), SpecialFunctions -> False (the opposite)."


Begin["'Private'"]


(** ALIASES **)


RS ::= RSolve

PS ::= PowerSum

EPS ::= ExponentialPowerSum

GF ::= GeneratingFunction

EGF ::= ExponentialGeneratingFunction

ST ::= SeriesTerm


(** ATTRIBUTES **)


Attributes[RSolve] = {HoldFirst}

Attributes[GeneratingFunction] = {HoldFirst}

Attributes[ExponentialGeneratingFunction] = {HoldFirst}

Attributes[Parse] = {HoldFirst}


(** OPTIONS **)


Options[RSolve] := Join[{Methods ->{MethodGF, MethodEGF}},
                        Options[HSolve],
                        Options[SeriesTerm] ]

Options[GeneratingFunction] := Options[HSolve]

Options[ExponentialGeneratingFunction] := Options[HSolve]

Options[HSolve] = {PrecisionHS -> Automatic}

Options[SeriesTerm] = {PrecisionST -> Automatic,
                       UseApart -> Automatic,
                       MakeReal -> True,
                       UseMod -> True,
                       SpecialFunctions -> True}

Options[PartialFractions] := Options[SeriesTerm]



(** MESSAGES **)
```

```
ST::badopt = "`` -> `` is not a valid option."

PF::badopt = "`` -> `` is not a valid option."

Parse::eqn = "`` is not an equation or a system of equations."



(** REDEFINITIONS OF SYSTEM FUNCTIONS **)


Unprotect[Binomial, Power]

    Binomial[n_, n_] := When[n >= 0, 1, 0]

    0^0 = 1

Protect[Binomial, Power]


(** (RSolve.m) **)

(* RSOLVE *)

RSolve[list1_, list2_, n_, opts___Rule] :=
    Block[{recur, conds, unknowns, startValues, methods, result, prs},
        prs = Parse[list1, list2, n];
        If[prs === Fail,
            Return[Fail],
          {recur, conds, unknowns, startValues} = prs ];
        methods = MakeList[Methods /.{opts} /.Options[RSolve]];
        result = Scan[Block[{temp},
             temp = # @@ {recur, conds, unknowns, startValues, n, opts};
             If[temp =!= Fail, Return[temp]]]&, methods];
        Which[result === Null,
                Return[Fail],
              True,
                Return[result] ]]


(* METHOD GF *)

MethodGF[recur_, conds_, unknowns_, startValues_, n_, opts___Rule] :=
    Block[{genf, mm, z, rec, con, i, temp, start, optsList = {opts}},
       {rec, con} = Reset[recur, conds, unknowns, startValues, 0];
        genf = FunctionSolve[rec, con, unknowns, n, PowerSum,
                              z, opts];
        If[genf =!= Fail,
           (Release[start /@ unknowns]) = startValues;
            If[FreeQ[optsList, UseApart],
                 AppendTo[optsList, UseApart -> Automatic] ];
            temp = (Table[mm /: Info[mm] = mm >= startValues[[i]];
                SeriesTerm[#[[i]],
               {z, 0, mm - startValues[[i]]}, Sequence @@ optsList],
               {i, Length[unknowns]} ]& /@ genf) /. mm -> n;
            Return[temp /. aa_?(MemberQ[unknowns, #]&)[kk_] :>
                              aa[Expand[kk + start[aa]]] ],
          Return[Fail] ]]


(* METHOD EGF *)

MethodEGF[recur_, conds_, unknowns_, startValues_, n_, opts___Rule] :=
    Block[{genf, mm, z, rec, con, i, temp, start, optsList = {opts}},
       {rec, con} = Reset[recur, conds, unknowns, startValues, 0];
        genf = FunctionSolve[rec, con, unknowns, n, ExponentialPowerSum,
                              z, opts];
        If[genf =!= Fail,
           (Release[start /@ unknowns]) = startValues;
            If[FreeQ[optsList, UseApart],
                 AppendTo[optsList, UseApart -> None] ];
            temp = (Table[mm /: Info[mm] = mm >= startValues[[i]];
                FactorialSimplify[
               (mm - startValues[[i]])! SeriesTerm[#[[i]],
               {z, 0, mm - startValues[[i]]}, Sequence @@ optsList] ],
               {i, Length[unknowns]} ]& /@ genf) /. mm -> n;
            Return[temp /. aa_?(MemberQ[unknowns, #]&)[kk_] :>
                              aa[Expand[kk + start[aa]]] ],
          Return[Fail] ]]
```

```
(** (GeneratingFunctions.m) **)

(* ROOTS OF UNITY *)

Omega[n_] := Exp[2 Pi I / n]


(* SIMPLIFICATION OF EXP[LOG[_]] AND I*SQRT[_] *)

SimplifyExpLog = {E^((a_. Log[b_] + c_.)d_.) -> b^(a d) E^(c d)}

SimplifyI = {Complex[0, a_] Power[b_, Rational[n_, 2]] :>
                              a (-1)^((n-1)/2) Expand[-b]^(n/2)}


(* SOLVING ROUTINE *)

FunctionSolve[recur_, conds_, unknowns_, n_, Transform_, z_,
              opts___Rule] :=
    Block[{eqns, extra, range, start, aa, kk, nm,
           lo, hi, functions, fnHeads, initials, constants, temp,
           series, order, xx, ii, bb, initvals, soln, times,
           arg, G, init, subst, pinitials},

(* transform recurrence equations into functional equations *)

        eqns = Expand[#[[1]] - #[[2]]]& /@ recur;
        eqns = Function[xx,
                   Block[{hom, inhom, yy},
                       yy = MakeList[xx, Plus];
                       inhom = Select[yy, FreeL[#, unknowns]&];
                       hom = Together[Plus @@ Complement[yy, inhom]];
                       Numerator[hom] + Expand[(Plus @@
                           inhom) Denominator[hom]] ]] /@ eqns;
        eqns = Transform[eqns, z, n];
        If[!FreeL[PatternList[eqns, Transform[__]], unknowns],
            Return[Fail]];
        Which[Transform === PowerSum,            G = Gf,
             Transform === ExponentialPowerSum, G = EGf];

(* substitute values which are explicitly given by initial
   conditions *)

        init = Select[conds, Function[xx, MatchQ[xx,
            a_?(MemberQ[unknowns, #]&)[_?NumberQ] == _?NumberQ]]];
        subst = Solve[init, Union @@ (PatternList[init, #[_]]& /@
            unknowns)];
        If[subst === {}, Return[Fail], eqns = eqns /. First[subst]];

(* remember the unknown functions *)

        functions = G[#][z]& /@ unknowns;
        fnHeads = G /@ unknowns;

(* solve the equations using Solve, DSolve, or HypergeometricSolve *)

        If[FreeQ[eqns, Derivative],

            temp = Solve[Thread[eqns == 0], functions],

    (* before using differential equation solvers, integrate
       equations of the form  lhs' == rhs'  *)

            times = Integrable[#, fnHeads, z]& /@ eqns;
            eqns = PreIntegrate[#[[1]], fnHeads, {z, #[[2]]}]& /@
                Thread[List[eqns, times]];
            eqns = Table[Sum[C[ii, jj] z^(jj - 1), {jj, times[[ii]]}] +
                eqns[[ii]], {ii, Length[eqns]} ];
            eqns = eqns /. Derivative[kk_][ff_] :> D[ff, {z, kk}];
            eqns = Thread[eqns == 0];

    (* use HSolve then DSolve *)

            temp = HSolve1[eqns, functions, z, opts];
            temp = temp /. C -> First[unknowns];
            If[!FreeL[temp, {HSolve, Fail}],
                temp = DSolve[eqns, functions, z] ]];
        If[temp === {}, Return[{}]];
        If[!FreeL[temp, {Solve, DSolve, Fail}], Return[Fail]];
        If[!FreeL[functions /. temp, fnHeads], Return[Fail]];
        temp = temp //. SimplifyExpLog;

(* find all initial terms occurring in conditions, equations
   or solutions *)

        initials = Union @@ (PatternList[{conds, eqns,
            functions /. temp}, #[_]]& /@ unknowns);

            (* sort initials by falling indices *)

        initials = Join @@ (Append[Cases[initials, _[#]]& /@
            Reverse[Union[PatternList[initials, _?NumberQ]]], {}]);
        pinitials = Select[initials, (#[[1]] >= 0)&];

            (* How many series terms will be needed to determine
               the initials? *)

       (order[#] = Max[Append[First /@ PatternList[initials,
            #[_]], 0 ]])& /@ unknowns;

(* determine values of initials and eliminate constants
   of integration *)

        constants = PatternList[functions /. temp, C[__]] // Union;
        series = temp /. (G[aa_?(MemberQ[unknowns, #]&)][z] -> bb_
            ) :> (aa -> Taylor[bb, z, 0, order[aa]]);
        If[!FreeL[series, {Series, Fail}], Return[Fail]];
        If[G === EGf,
            series = series /. (aa_?(MemberQ[unknowns, #]&) -> bb_
                ) :> (aa -> Table[nm!, {nm, 0, order[aa]}] bb) ];
        initvals = Function[xx, Solve[Union[conds,
           (# == (Head[#] /. xx)[[First[#] + 1]])& /@ pinitials,
            Thread[Join @@ Rest /@ CList[#[[2,1]]& /@ xx /.
                z -> z^(-1), z ] == 0 ] ],
            Join[constants, initials] ]] /@ series;

(* plug in initial values *)

        soln = Select[Union @@ Table[(Function[xx,
            G[xx][z] /. temp[[ii]] ] /@
            unknowns) /. initvals[[ii]],
           {ii, Length[temp]} ], (Head[#] === List)& ];
        Return[soln] ]


(* TAYLOR and CLIST *)

Taylor[f_, z_, a_, n_] :=
    Block[{tmp, kk},
        tmp = SafeSeries[f, {z, a, n}];
        If[tmp === Fail, Return[Fail]];
        tmp = CoefficientList[tmp, z];
        If[Length[tmp] > n + 1, tmp = Take[tmp, n + 1]];
        Return[Join[tmp, Table[0,
            {kk, n + 1 - Length[tmp]} ]]]]

CList[l_List, x_] := CList[#, x]& /@ l

CList[0, x_] := {0}

CList[a_, x_] := CoefficientList[a, x] /; a =!= 0


(* INTEGRABLE and PREINTEGRATE *)

Integrable[a_Plus, b___] := Min[Integrable[#, b]& /@ (List @@ a)]

Integrable[a_, functions_, z_] := Infinity /; FreeL[a, functions]

Integrable[a_ b_, functions_, z_] :=
    Integrable[b, functions, z] /; FreeQ[a, z]

Integrable[a_, functions_, z_] :=
    FirstArg[a] /; FirstHead[a] === Derivative

Integrable[a___] := 0


PreIntegrate[a_, functions_, {z_, 0}] := a

PreIntegrate[a_Plus, b___] := PreIntegrate[#, b]& /@ a
```

```
PreIntegrate[a_, functions_, {z_, n_}] :=
    Nest[Integrate[#, z]&, a, n] /; FreeL[a, functions]

PreIntegrate[a_ b_, functions_, {z_, n_}] :=
    a PreIntegrate[b, functions, {z, n}] /; FreeQ[a, z]

PreIntegrate[a_, functions_, {z_, n_}] :=
     MapAt[(# - n)&, a,
         Append[First[Position[a, Derivative[_]]], 1]] /;
                             FirstHead[a] === Derivative


(* FIRSTHEAD, FIRSTARG and HEADCOUNT *)

FirstHead[a_] := Nest[Head, a, HeadCount[a]]


FirstArg[a_] := Which[Length[a] == 0,
                        Null,
                     True,
                         First[Nest[Head, a, HeadCount[a] - 1]] ]


HeadCount[a_] := Which[Length[a] == 0,
                          0,
                      True,
                          HeadCount[Head[a]] + 1 ]



(* POWER SUM *)

    (* lists and equations *)

PowerSum[expr_List, rest__] := PowerSum[#, rest]& /@ expr

PowerSum[expr_Equal, rest__] := PowerSum[#, rest]& /@ expr


    (* revert to inner syntax *)

PowerSum[expr_, {z_Symbol, n_, n0_:0}] :=
    Block[{e = expr //. {Sum[a_, {k_, m_}] :> Sum[a, {k, 1, m}],
                         EvenQ[a_] :> SymbolicMod[a, 2] == 0,
                         OddQ[a_]  :> SymbolicMod[a, 2] == 1,
                         Mod -> SymbolicMod} },
        PowerSum[e, z, n, n0] /.
            Derivative[kk_][ff_] :> D[ff, {z, kk}] ];


    (* nonzero starting point *)

PowerSum[expr_, z_, n_, n0_] :=
    z^n0 PowerSum[Expand[expr /. n -> n + n0] /.
        a_Symbol?(Context[#] =!= "System`" && # =!= K &)[m_] :>
        a[Expand[m]], z, n] /; NumberQ[N[n0]]


    (* the geometric series *)

PowerSum[1, z_, n_] := 1/(1 - z)


    (* linearity *)

PowerSum[c_ expr_., z_, n_] :=
    c PowerSum[expr, z, n] /; FreeL[c, {n, Blank}]

PowerSum[c_.(expr1_ + expr2_), rest__] :=
                      PowerSum[c expr1, rest] + PowerSum[c expr2, rest]

PowerSum[Sum[a_, {k_, lo_, hi_}] b_., z_, n_] :=
    Sum[PowerSum[Expand[a b], z, n], {k, lo, hi}] /;
                                     FreeQ[{lo, hi}, n] && FreeQ[b, k]


    (* powers *)
```

```
PowerSum[c_^((a_ + b_)d_) e_., rest__] :=
    PowerSum[c^(a d + b d) e, rest]

PowerSum[c_^(a_ + b_) e_., z_, n_] :=
    c^a PowerSum[c^b e, z, n] /;
                          FreeL[c, {n, Blank}] && FreeL[a, {n, Blank}]

PowerSum[(a_ + b_)^c_Integer?Positive d_., rest__] :=
    PowerSum[Expand[(a + b)^c d], rest]


    (* exponentials *)

PowerSum[e_. c_^((a_. n_ + b_.)d_.), z_, n_] :=
    (c^(b d) PowerSum[e, z, n] /. z -> c^(a d) z) /;
  (FreeL[#, {n, Blank}]& /@ And[a, b, c, d]) && FreeL[e, {z, Blank}]


    (* binomials *)

PowerSum[Binomial[a_, n_ + k_.], z_, n_] :=
    z^(-k)((1 + z)^a - Sum[Binomial[a, m] z^m, {m, 0, k - 1}]) /;
                                            FreeQ[a, n] && IntegerQ[k]

PowerSum[Binomial[a_, b_], z_, n_] :=
    Block[{e = Expand[n - b], r = Expand[a - 2 b], m},
        z^e (((1 - Sqrt[1 - 4 z])/(2 z))^r / Sqrt[1 - 4 z] -
            Sum[Binomial[2 m + r, m] z^m, {m, 0, -(e + 1)}] ) /;
                                           IntegerQ[e] && IntegerQ[r]]


    (* trinomials *)

PowerSum[Sum[a_. b_Binomial c_Binomial, {k_, 0, n_}],
    z_, n_] :=
    Block[{aa = a^(1/k)},
    1/Sqrt[1 - 2 (2 aa + 1) z + z^2] /;
        FreeL[aa, {k, n}] &&
        MakeTrinomial[b c] == Sort[Multinomial[k, k, n - k]] ]

PowerSum[Sum[a_. b_Binomial c_Binomial, {k_, 0, n_}],
    z_, n_] :=
    Block[{aa = a^(1/(n - k))},
    1/Sqrt[1 - 2 (2 aa + 1) z + z^2] /;
        FreeL[aa, {k, n}] &&
        MakeTrinomial[b c] == Sort[Multinomial[k, n - k, n - k]] ]

PowerSum[Sum[a_. b_Binomial c_Binomial, {k_, 0, n_}],
    z_, n_] :=
    Block[{aa = a^(1/k)},
    1/Sqrt[1 - 2 z + (1 - 4 aa) z^2] /;
        FreeL[aa, {k, n}] &&
        MakeTrinomial[b c] == Sort[Multinomial[k, k, n - 2 k]] ]

PowerSum[Sum[a_. b_Binomial c_Binomial, {k_, 0, n_}],
    z_, n_] :=
    Block[{aa = a^(1/(n - k))},
    1/Sqrt[1 - 2 z + (1 - 4 aa) z^2] /;
        FreeL[aa, {k, n}] &&
        MakeTrinomial[b c] == Sort[Multinomial[n - k, n - k, 2 k - n]] ]


    (* PowerSum is the inverse of SeriesTerm *)

PowerSum[Literal[SeriesTerm[f_, z_, n_ + m_.]],
    z_, n_] :=
    Block[{k},
        z^(-m) (f - Sum[SeriesTerm[f, {z, 0, k}] z^k,
            {k, 0, m - 1}]) /;
                PoleMultiplicity[f, {z, 0}] == 0 && IntegerQ[m] ]


    (* trigonometric inhomogeneity *)

PowerSum[Cos[n_], rest__] :=
    PowerSum[(E^(I n) + E^(-I n))/2, rest]


PowerSum[Sin[n_], rest__] :=
    PowerSum[(E^(I n) - E^(-I n))/2/I, rest]
    (* factorial inhomogeneity *)

PowerSum[expr_./(n_ + a_.)!, z_, n_] :=
    Block[{k},
       (ExponentialPowerSum[Expand[expr /. n -> n - a], {z, n}] -
       Sum[(expr /. n -> k - a) z^k / k!, {k, 0, a - 1}]) / z^a] /;
                                                          IntegerQ[a]


    (* generating functions of sequences

      (here we assume that a[n] is defined for n >= 0) *)


PowerSum[n_^m_. a_[n_ + d_.], z_, n_] :=
    Block[{j, k},
         Sum[StirlingS2[m, j] z^j
            Derivative[j][(PowerSum[a[k], z, k] -
            Sum[a[k] z^k, {k, 0, d - 1}])/z^d ],
           {j, 0, m}] // Factor] /;
     IntegerQ[d] && TTQ[d >= 0] && IntegerQ[m] && TTQ[m >= 0]

PowerSum[a_[k_. n_ + d_.], z_, n_] :=
    Block[{j, m},
        (z^(-d/k)/k Sum[Omega[k]^(-d j) (Gf[a][Omega[k]^j z^(1/k)] -
            Sum[Omega[k]^(m j) a[m] z^(m/k), {m, 0, d-1}]),
            {j, 0, k - 1} ] ) // Factor] /;
         IntegerQ[d] && TTQ[d >= 0] && IntegerQ[k] && TTQ[k > 0]



    (* rational inhomogeneity *)

PowerSum[n_^m_., z_, n_] :=
    Block[{j},
        Factor[Sum[(StirlingS2[m, j] j! z^j)/(1 - z)^(j + 1),
                  {j, 0, m}] ] ] /; IntegerQ[m] && TTQ[m >= 0]


PowerSum[(n_ + a_)^m_Integer?Negative, z_, n_] :=
    Block[{j},
        z^(-a) PolyLog[-m, z] - Sum[z^(j-a) j^m, {j, 1, a-1}] ] /;
                                          IntegerQ[a] && TTQ[a > 0]


    (* convolutions *)

PowerSum[Sum[expr_, {k_, lo_, hi_}], z_, n_] :=
    Block[{nfree, nfull, s, aa = Expand[lo], bb = Expand[hi - n]},
       {nfree, nfull} = If[Head[expr] === Times,
           {Select[expr,  FreeQ[#, n]&], Select[expr, !FreeQ[#, n]&]},
                If[FreeQ[expr, n], {expr, 1}, {1, expr}] ];
        Sum[z^k nfree PowerSum[Expand[nfull /. n -> s + k],
                z, s, -k], {k, aa, bb - 1}] +
            PowerSum[nfree PowerSum[Expand[nfull /. n -> s + k],
                    z, s, -bb], z, k, Max[aa, bb] ] /;
                                          FreeQ[aa, n] && FreeQ[bb, n] ]


    (* multisection of series *)

PowerSum[If[SymbolicMod[n_ + d_., m_] == k_, a_, b_] c_., z_, n_] :=
    Block[{l = Mod[k - d, m], jj,
          fps = PowerSum[Expand[(a - b)c], z, n]},
        PowerSum[Expand[b c], z, n] +
        1/m Sum[Omega[m]^(jj(m - 1)) fps /. z -> z Omega[m]^jj,
        {jj, 0, m - 1}] // Expand ]


    (* other conditionals *)

PowerSum[If[cond_, a_, b_] c_., z_, n_] :=
    Block[{t, tt, s, k},
        t = IneqSolve[cond && n >= 0, n];
        tt = IneqSolve[!cond && n >= 0, n];
        Plus @@ (Sum[Expand[a c] z^n, {n, First[#], Last[#]}]& /@ t) +
            Plus @@ (Sum[Expand[b c] z^n, {n, First[#], Last[#]}]& /@
            tt) /.
               {Sum[s_. z^n, {n, k_, Infinity}] :>
                    PowerSum[s, {z, n, k}],
                Sum[0, {__}] -> 0} ]
    (* PowerSum under Series *)

PowerSum /:
    Series[PowerSum[a_, z_, n_], {z_, 0, m_}] :=
        Block[{k},
        Sum[(a /. n -> k) z^k, {k, 0, m}] + O[z]^(m+1)]


(* EXPONENTIAL POWER SUM *)


    (* lists and equations *)

ExponentialPowerSum[expr_List, rest__] :=
    ExponentialPowerSum[#, rest]& /@ expr

ExponentialPowerSum[expr_Equal, rest__] :=
    ExponentialPowerSum[#, rest]& /@ expr


    (* revert to inner syntax *)

ExponentialPowerSum[expr_, {z_Symbol, n_, n0_:0}] :=
    Block[{e = expr //. {Sum[a_, {k_, m_}] :> Sum[a, {k, 1, m}],
                         EvenQ[a_] :> SymbolicMod[a, 2] == 0,
                         OddQ[a_]  :> SymbolicMod[a, 2] == 1,
                         Mod -> SymbolicMod} },
        ExponentialPowerSum[e, z, n, n0] /.
            Derivative[kk_][ff_] :> D[ff, {z, kk}] ];


    (* nonzero starting point *)

ExponentialPowerSum[expr_, z_, n_, n0_] :=
    ExponentialPowerSum[expr /. n -> n + n0, z, n] /; n0 >= 0


    (* the exponential series *)

ExponentialPowerSum[1, z_, n_] := E^z


    (* linearity *)

ExponentialPowerSum[c_ expr_., z_, n_] :=
    c ExponentialPowerSum[expr, z, n] /; FreeL[c, {n, Blank}]

ExponentialPowerSum[c_.(expr1_ + expr2_), rest__] :=
    ExponentialPowerSum[c expr1, rest] +
    ExponentialPowerSum[c expr2, rest]

ExponentialPowerSum[Sum[a_, {k_, lo_, hi_}] b_., z_, n_] :=
    Sum[ExponentialPowerSum[Expand[a b], z, n], {k, lo, hi}] /;
                                        FreeQ[{lo, hi}, n] && FreeQ[b, k]


    (* powers *)

ExponentialPowerSum[c_^((a_ + b_)d_), rest__] :=
    ExponentialPowerSum[c^(a d + b d), rest]

ExponentialPowerSum[c_^(a_ + b_), z_, n_] :=
    c^a ExponentialPowerSum[c^b, z, n] /;
                        FreeL[c, {n, Blank}] && FreeL[a, {n, Blank}]


    (* exponentials *)

ExponentialPowerSum[e_. c_^((a_. n_ + b_.)d_.), z_, n_] :=
    (c^(b d) ExponentialPowerSum[e, z, n] /. z -> c^(a d) z) /;
  (FreeL[#, {n, Blank}]& /@ And[a, b, c, d]) && FreeL[e, {z, Blank}]


    (* polynomial inhomogeneity *)

ExponentialPowerSum[n_^m_., z_, n_] :=
    Block[{j},
        E^z Factor[Sum[StirlingS2[m, j] z^j,
                  {j, 0, m}] ] ]  /; IntegerQ[m] && TTQ[m >= 0]
    (* factorial inhomogeneity *)

ExponentialPowerSum[(n_ + a_.)!, z_, n_] :=
    PowerSum[Pochhammer[n + 1, a], z, n] /; IntegerQ[a]


    (* convolutions *)

ExponentialPowerSum[Sum[expr_ Binomial[n_ + cc_, k_], {k_, lo_, hi_}],
    z_, n_] :=
    Block[{aa = Expand[lo], bb = Expand[hi - n]},
        Derivative[cc][ExponentialPowerSum[Sum[(expr /. n -> n - cc)
            Binomial[n, k], {k, aa, n + bb - cc} ], z, n ] ] /;
      IntegerQ[cc] && TTQ[cc > 0] && FreeQ[aa, n] && FreeQ[bb, n] ]



ExponentialPowerSum[Sum[expr_ Binomial[n_, k_], {k_, lo_, hi_}],
    z_, n_] :=
    Block[{nfree, nfull, s, aa = Expand[lo], bb = Expand[hi - n]},
       {nfree, nfull} = If[Head[expr] === Times,
          {Select[expr,  FreeQ[#, n]&], Select[expr, !FreeQ[#, n]&]},
               If[FreeQ[expr, n], {expr, 1}, {1, expr}] ];
       Sum[z^k nfree/k! ExponentialPowerSum[Expand[nfull /. n -> s + k],
               z, s, -k], {k, aa, bb - 1}] +
           ExponentialPowerSum[nfree ExponentialPowerSum[Expand[nfull /.
               n -> s + k], z, s, -bb], z, k, Max[aa, bb] ] /;
                                         FreeQ[aa, n] && FreeQ[bb, n] ]


    (* multisection of series *)

ExponentialPowerSum[If[SymbolicMod[n_ + d_., m_] == k_, a_, b_] c_.,
z_, n_] :=
    Block[{l = Mod[k - d, m], jj,
          fps = ExponentialPowerSum[Expand[(a - b)c], z, n]},
        ExponentialPowerSum[Expand[b c], z, n] +
        1/m Sum[Omega[m]^(jj(m - 1)) fps /. z -> z Omega[m]^jj,
        {jj, 0, m - 1}] // Expand ]


    (* other conditionals *)

ExponentialPowerSum[If[cond_, a_, b_] c_., z_, n_] :=
    Block[{t, tt, s, k},
        t = IneqSolve[cond && n >= 0, n];
        tt = IneqSolve[!cond && n >= 0, n];
        Plus @@ (Sum[Expand[a c] z^n / n!, {n, First[#], Last[#]}]& /@
            t) +
        Plus @@ (Sum[Expand[b c] z^n / n!, {n, First[#], Last[#]}]& /@
            tt) /.
              {Sum[s_. z^n / n!, {n, k_, Infinity}] :>
                  ExponentialPowerSum[s, {z, n, k}],
               Sum[0, {__}] -> 0} ]


    (* generating functions of sequences

      (here we assume that a[n] is defined for n >= 0) *)

ExponentialPowerSum[a_[n_ + d_.], z_, n_] :=
    Derivative[d][EGf[a][z]]  /; IntegerQ[d] && TTQ[d >= 0]

ExponentialPowerSum[n_^m_. a_[n_ + d_.], z_, n_] :=
    Block[{j, k},
        Sum[StirlingS2[m, j] z^j
            Derivative[d + j][EGf[a][z]], {j, 0, m}] ] /;
     IntegerQ[d] && TTQ[d >= 0] && IntegerQ[m] && TTQ[m >= 0]

ExponentialPowerSum[n_^m_. a_[n_ - 1], z_, n_] :=
    Block[{j, k},
        Sum[StirlingS2[m, j] z^j
            Derivative[j - 1][EGf[a][z]], {j, m}] ] /;
                                    IntegerQ[m] && TTQ[m > 0]




(* GENERATING FUNCTION *)
```

```
GeneratingFunction[list1_, list2_, n_, z_, opts___Rule] :=
    Block[{temp, recur, conds, unknowns, startValues, start,
          rec, con, i, prs},
        prs = Parse[list1, list2, n];
        If[prs === Fail,
            Return[Fail],
          {recur, conds, unknowns, startValues} = prs ];
       {rec, con} = Reset[recur, conds, unknowns, startValues, 0];
       (Release[start /@ unknowns]) = startValues;
        temp = FunctionSolve[rec, con, unknowns, n, PowerSum,
                             z, opts];
        If[temp === Fail,
            Return[Fail],
          temp = Table[#[[i]] z^startValues[[i]],
              {i, Length[unknowns]} ]& /@ temp;
          Return[temp/.aa_?(MemberQ[unknowns,#]&)[kk_] :>
                             aa[Expand[kk + start[aa]]] ]]]


(* EXPONENTIAL GENERATING FUNCTION *)

ExponentialGeneratingFunction[list1_, list2_, n_, z_, opts___Rule] :=
    Block[{temp, recur, conds, unknowns, startValues, start,
          rec, con, prs},
        prs = Parse[list1, list2, n];
        If[prs === Fail,
            Return[Fail],
          {recur, conds, unknowns, startValues} = prs ];
       {rec, con} = Reset[recur, conds, unknowns, startValues, 0];
       (Release[start /@ unknowns]) = startValues;
        temp = FunctionSolve[rec, con, unknowns, n, ExponentialPowerSum,
                             z, opts];
        If[temp === Fail,
            Return[Fail],
            Return[temp/.aa_?(MemberQ[unknowns,#]&)[kk_] :>
                           aa[Expand[kk + start[aa]]] ]]]



(** (Hypergeometric.m) **)


(* HSOLVE *)

HSolve[eq_, f_, x_Symbol, opts___Rule] :=
    Block[{hsol = HSolve1[eq, f, x, opts], ll},
        ll = PList[hsol, C[_]];
        Return[hsol /. Thread[ll -> Table[C[k], {k, Length[ll]}]]] ]

HSolve1[eq_, f_, x_Symbol, opts___Rule] :=

    Block[{lhs, rhs, terms, weights, inhom, minw, theta, ll, llc, rlc,
          aList, bList, badB, nnHS, n0, subst, eqn, y, k, pr, temp,
          pm, nneg, n1, badB1},

        pr = PrecisionHS /.{opts} /.Options[HSolve];
        If[Head[eq] === List,
            If[Length[eq] > 1, Return[Fail]];
            eqn = First[eq],
            eqn = eq ];
        If[Head[f] === List,
            If[Length[f] > 1, Return[Fail]];
            y = Head[First[f]],
            y = Head[f] ];
        lhs = Expand[Numerator[Together[First[eqn] - Last[eqn]]]];
        terms = Select[MakeList[lhs, Plus], !FreeQ[#, y]&];
        weights = Union[Weight[#, y[x], x]& /@ terms];
        minw = Min[weights];
        If[!MemberQ[{{0},{0,1}}, weights - minw], Return[Fail]];
        lhs = Expand[lhs/x^minw] /. x^k_.Derivative[n_][y][x] ->
            x^(k - n) Sum[StirlingS1[n, j] theta^j y[x], {j,0,n}]
            // Expand;
        inhom = Plus @@ Select[MakeList[lhs, Plus], FreeQ[#, y]&]
            // Expand;
        pm = PoleMultiplicity[inhom, {x, 0}];
        If[!(IntegerQ[pm] && pm >= 0), Return[Fail]];
        If[!PolynomialQ[x^pm inhom, x], Return[Fail]];
        lhs = (lhs - inhom) / y[x] // Expand;
        rhs = Plus @@ Select[MakeList[lhs, Plus], FreeQ[#, x]&];
        lhs = Expand[(rhs - lhs)/x];
        {llc, rlc} = {LeadingCoef[lhs, theta], LeadingCoef[rhs, theta]};
        aList = If[FreeQ[lhs, theta], {}, -#[[1,2]]& /@
    Which[pr === Automatic,
                {ToRules[Roots[lhs == 0, theta,
                 Cubics -> False, Quartics -> False] ]} /.
                 Literal[Roots[a_, b_, c___]] :> NRoots[a, b],
           pr === Infinity,
                 Solve[lhs == 0, theta] /.
                 Roots -> NRoots,
           NumberQ[pr],
                 Solve[lhs == 0, theta] /.
                 Literal[Roots[a_, b_]] :> NRoots[a, b, pr] ]];
bList = If[FreeQ[rhs, theta], {}, (1 - #[[1,2]])& /@
    Which[pr === Automatic,
                {ToRules[Roots[rhs == 0, theta,
                 Cubics -> False, Quartics -> False] ]} /.
                 Literal[Roots[a_, b_, c___]] :> NRoots[a, b],
           pr === Infinity,
                 Solve[rhs == 0, theta] /.
                 Roots -> NRoots,
           NumberQ[pr],
                 Solve[rhs == 0, theta] /.
                 Literal[Roots[a_, b_]] :> NRoots[a, b, pr] ]];
badB = Select[bList, (IntegerQ[#] && # <= 0)&];
nnHS = If[badB != {}, 1 - Min[badB], 0];
n0 = Max[nnHS, Exponent[inhom, x]];
badB1 = Select[bList, (IntegerQ[#] && # > 0)&];
nneg = If[badB1 != {}, Max[badB1] - 1, 0];
n1 = Max[nneg, pm];
Clear[c];
c[-n1 - 1] = 0;
subst = {ToRules[Reduce[Table[If[n + 1 == 0,
    Coefficient[inhom, x, 0] /. x -> 1/x /. x -> 0,
    Coefficient[inhom, x^(n+1)]] +
    rlc Times @@ (bList + n) c[n+1]
     == llc Times @@ (aList + n) c[n], {n, -n1 - 1, n0 - 1}],
    Table[c[n], {n, n0, -n1, -1}] ]]};
If[subst === {}, Return[{}] ];

If[Times @@ (Pochhammer[# + nnHS, n0 - nnHS]& /@
   aList) =!= 0 && llc =!= 0,

(* then *)

{aList, bList} = {aList, bList} + nnHS;
If[!MemberQ[bList, 1], AppendTo[aList, 1];
                         AppendTo[bList, 1] ];
bList = Drop[bList, {FirstPos[bList, 1]}];
If[MemberQ[aList, #], aList = Drop[aList,
                           {FirstPos[aList, #]}];
                        bList = Drop[bList,
                           {FirstPos[bList, #]}]
  ]& /@ bList;
aList = Sort[aList];
bList = Sort[bList];
If[NumberQ[pr],
    aList = N[aList, pr];
    bList = N[bList, pr] ];

temp = (Sum[c[k] x^k, {k, -n1, n0 - 1}] +
    c[n0] (llc/rlc)^(nnHS - n0) (Times @@
    (Pochhammer[#, n0 - nnHS]&
    /@ bList)) / (Times @@ (Pochhammer[#, n0 - nnHS]& /@
    aList)) (n0 - nnHS)!
    x^nnHS (HypergeometricF[aList, bList, llc/rlc x] -
    Sum[(Times @@ (Pochhammer[#, k]& /@ aList))/
        (Times @@ (Pochhammer[#, k]& /@ bList)) (llc/
        rlc x)^k / k!,
        {k, 0, n0 - nnHS - 1}] ) ) /. subst // Simplify,

(* else *)

{aList, bList} = {aList, bList} + n0;
If[!MemberQ[bList, 1], AppendTo[aList, 1];
                         AppendTo[bList, 1] ];
bList = Drop[bList, {FirstPos[bList, 1]}];
If[MemberQ[aList, #], aList = Drop[aList,
                           {FirstPos[aList, #]}];
                        bList = Drop[bList,
                           {FirstPos[bList, #]}]
  ]& /@ bList;
aList = Sort[aList];
bList = Sort[bList];
If[NumberQ[pr],
    aList = N[aList, pr];
```

```
            bList = N[bList, pr] ];

        temp = (Sum[c[k] x^k, {k, -n1, n0 - 1}] +
            c[n0] x^n0 HypergeometricF[aList, bList, llc/rlc x])
            /. subst // Expand ];

        temp = temp /. c -> C;
        Return[Thread[y[x] -> #]& /@ {temp}] ]


(* WEIGHT *)


Weight[a_. x_^k_. Derivative[n_][y_][x_], y_[x_], x_] := k - n /;
                                                   FreeL[a, {x, y}]

Weight[a_. Derivative[n_][y_][x_], y_[x_], x_] := -n /; FreeL[a, {x, y}]

Weight[a_. x_^k_. y_[x_], y_[x_], x_] := k  /; FreeL[a, {x, y}]

Weight[a_. y_[x_], y_[x_], x_] := 0  /; FreeL[a, {x, y}]

Weight[a_, y_[x_], x_] := Infinity  /; FreeQ[a, y]


(* HYPERGEOMETRIC F *)


HypergeometricF[aList_List, bList_List, c_.z_] :=
    Block[{max, kk, neg = Select[aList, (IntegerQ[#] && # <= 0)&]},
       (max = - Max[neg];
        Sum[Times @@ (Pochhammer[#, kk]& /@ aList) /
            Times @@ (Pochhammer[#, kk]& /@ bList) / kk! (c z)^kk,
                {kk, 0, max}]) /; Length[neg] > 0 ]

HypergeometricF[a_List, b_List, 0] := 1

HypergeometricF[{}, {}, z_] := E^z

HypergeometricF[{a_}, {}, z_] := 1/(1 - z)^a

HypergeometricF[{}, {c_}, z_] := Hypergeometric0F1[c, z]

HypergeometricF[{a_}, {c_}, z_] := Hypergeometric1F1[a, c, z]

HypergeometricF[{a_, b_}, {c_}, z_] := Hypergeometric2F1[a, b, c, z]

HypergeometricF /:
    Series[HypergeometricF[aList_List, bList_List, c_.z_],
    {z_, 0, n_}] :=
        Block[{kk},
            Sum[Times @@ (Pochhammer[#, kk]& /@ aList) /
                Times @@ (Pochhammer[#, kk]& /@ bList) / kk! (c z)^kk,
                {kk, 0, n}] + O[z]^(n + 1) ]

Derivative[0, 0, n_Integer?Positive][HypergeometricF][aList_List,
    bList_List, c_.z_] :=
        c^n Times @@ (Pochhammer[#, n]& /@ aList) /
            Times @@ (Pochhammer[#, n]& /@ bList) HypergeometricF[
            aList + n, bList + n, c z]




(** (SeriesExpansions.m) **)


(* SERIES TERM *)

SeriesTerm[expr_List, rest__] := SeriesTerm[#, rest]& /@ expr

SeriesTerm[f_, {z_, a_, n_}, opts___Rule] :=
    Block[{ff = f /. z -> z + a, nnST, useApart, n0, temp0, temp,
            precision, makeReal, useMod, specialFunctions},
        IntegerQ[nnST] ^= True;
        Info[nnST] ^= Info[n] /. Solve[n == nnST, PatternList[n,
                _Symbol?(Context[#] =!= "System`"&) ]][[1]];
        sumDepth = 1;
        useApart = UseApart /.{opts} /.Options[SeriesTerm];
        precision = PrecisionST /.{opts} /.Options[SeriesTerm];
        makeReal = MakeReal /.{opts} /.Options[SeriesTerm];
        useMod = UseMod /.{opts} /.Options[SeriesTerm];
        specialFunctions = SpecialFunctions /.{opts} /.
                                          Options[SeriesTerm];
        Which[useApart === Automatic,
                 ff = Apart[ff],
              useApart === All,
                 ff = Apart[ff, z],
              useApart =!= None,
                 Message[ST::badopt, UseApart, useApart ] ];
        temp = SeriesTerm[ff, z, nnST] /. {nnST -> n};
        temp = Chop[temp] /. SimplifyFloat;
        temp = temp //. SimplifyTrig;
        temp = temp /. SimplifyBinomial;
        temp = temp /.
                {Power[aa_, b_] :> Power[aa, Expand[b]],
                 Binomial[aa_, b_] :> Binomial[Expand[aa], Expand[b]],
                 If[aa_, b_, c_] :> If[ISolve[aa], b, c] /;
                     FreeL[aa, {OddQ, EvenQ, SymbolicMod}] };
        temp = temp /. aa_If :> Release //@ aa;
        temp = temp /.
                {If[OddQ[m_], If[m_ >= aa_, b_, c_] d_., e_] :>
                     If[OddQ[m], Release[Expand[b d]], e] /;
                     Entails[Info[m], m >= aa - 1] && OddQ[aa],
                 If[EvenQ[m_], If[m_ >= aa_, b_, c_] d_., e_] :>
                     If[EvenQ[m], Release[Expand[b d]], e] /;
                     Entails[Info[m], m >= aa - 1] && EvenQ[aa] };
        temp = temp //. SimplifySum;
        If[NumberQ[precision],
            temp = N[temp, precision] /. SimplifyFloat ];
        If[MatchQ[Info[n], n >= _],
            n0 = Info[n][[2]];
            temp0 = temp /. If[n >= aa_, b_, _] :> b /; aa == n0 + 1;
            If[(temp0 /. n -> n0) == (temp /. n -> n0), temp = temp0] ];
        Return[temp] ]


    (* finite n *)

SeriesTerm[f_, z_Symbol, 0] := SeriesTerm[z f, z, 1] /;
                                        FreeQ[f, HypergeometricF]

SeriesTerm[f_, z_Symbol, n_Integer] :=
    Coefficient[SafeSeries[f, {z, 0, n}], z^n] /;
                                  n != 0 && FreeQ[f, HypergeometricF]

    (* linearity *)

SeriesTerm[f_Plus, rest__] := SeriesTerm[#, rest]& /@ f

SeriesTerm[c_ f_, z_Symbol, rest__] := c SeriesTerm[f, z, rest] /;
                                                        FreeQ[c, z]

    (* SeriesTerm is inverse of Gf and of PowerSum *)

SeriesTerm[Gf[a_][z_], z_Symbol, n_] := a[n]

SeriesTerm[PowerSum[a_, z_, n_], z_Symbol, m_] :=
    a /. n -> m


    (* SeriesTerm is close to the inverse of EGf and of
       ExponentialPowerSum *)

SeriesTerm[EGf[a_][z_], z_Symbol, n_] := a[n] / n!

SeriesTerm[ExponentialPowerSum[a_, z_, n_], z_Symbol, m_] :=
    (a / n!) /. n -> m



    (* binomials *)

SeriesTerm[(a_.z_ + b_)^p_, z_Symbol, n_] :=
    Block[{k},
       (b^p When[p < 0,
                (-a/b)^n Binomial[n - p - 1, n],
                 (a/b)^n Binomial[p, n]]) /;
                                (FreeQ[#, z]&) /@ (a && b && p)]

SeriesTerm[(a_.z_^m_. + b_)^p_, z_Symbol, n_] :=
    Cancel[a^p SeriesTerm[z^(m p)(1 + b/a z^(-m))^p, z, n]] /;
              (FreeQ[#, z]&) /@ (a && b && m && p) && TTQ[m < 0]
```

```
SeriesTerm[(a_.z_^m_ + b_)^p_, z_Symbol, n_] :=
    If[SymbolicMod[n, m] == 0,
        Cancel[b^p SeriesTerm[(1 + a/b z)^p, z, n/m]],
      0] /; (FreeQ[#, z]&) /@ (a && b && m && p) && TTQ[m >= 0] &&
                                                 Head[p] =!= Integer

SeriesTerm[(a_.z_^m_. + b_.z_^k_.)^p_, z_Symbol, n_] :=
    Cancel[b^p SeriesTerm[z^(k p)(1 + a/b z^(m-k))^p, z, n]] /;
                        (FreeQ[#, z]&) /@ (a && b && m && k && p)


    (* hypergeometrics *)

SeriesTerm[HypergeometricF[aList_, bList_, c_.z_], z_Symbol, n_] :=
     When[n >= 0,
    (Times @@ (If[IntegerQ[#], (# + n - 1)!/(# - 1)!,
        Pochhammer[#, n]]& /@ aList))/
    (Times @@ (If[IntegerQ[#], (# + n - 1)!/(# - 1)!,
        Pochhammer[#, n]]& /@ bList))/n! c^n, 0] /; FreeQ[c, z]

SeriesTerm[Hypergeometric2F1[a_, b_, c_, d_.z_], z_Symbol, n_] :=
     When[n >= 0,
    (Times @@ (If[IntegerQ[#], (# + n - 1)!/(# - 1)!,
        Pochhammer[#, n]]& /@ {a, b}))/
    (Times @@ (If[IntegerQ[#], (# + n - 1)!/(# - 1)!,
        Pochhammer[#, n]]& /@ {c, 1})) d^n, 0] /; FreeQ[d, z]

SeriesTerm[Hypergeometric1F1[a_, b_, c_.z_], z_Symbol, n_] :=
     When[n >= 0,
     If[IntegerQ[a], (a + n - 1)!/(a - 1)!, Pochhammer[a, n]]/
    (Times @@ (If[IntegerQ[#], (# + n - 1)!/(# - 1)!,
        Pochhammer[#, n]]& /@ {b, 1})) c^n, 0] /; FreeQ[c, z]


SeriesTerm[Hypergeometric0F1[a_, b_.z_], z_Symbol, n_] :=
     When[n >= 0,
     1/(Times @@ (If[IntegerQ[#], (# + n - 1)!/(# - 1)!,
        Pochhammer[#, n]]& /@ {a, 1})) b^n, 0] /; FreeQ[b, z]


     (* exponentials *)

SeriesTerm[E^(k_.z_ + m_.), z_Symbol, n_] := E^m k^n/n! /;
                                 FreeQ[k, z] && FreeQ[m, z]

SeriesTerm[a_^b_, z_Symbol, n_] := SeriesTerm[E^(b Log[a]), z, n] /;
                                    a =!= E && FreeQ[a, z]

    (* logarithms *)

SeriesTerm[Log[(a_ + b_.z_)^k_.], z_Symbol, n_] :=
    When[n == 0, Release[k Log[a]], Release[-k(-b/a)^n / n]] /;
                   FreeQ[k, z] && FreeQ[a, z] && FreeQ[b, z]

SeriesTerm[Log[a_Times], rest___] :=
    Plus @@ (SeriesTerm[Log[#], rest]& /@ a)


    (* powers and constants *)

SeriesTerm[z_^k_.f_, z_Symbol, n_] := SeriesTerm[f, z, n-k] /;
                           IntegerQ[k] && FreeQ[k, z] && !FreeQ[f, z]
SeriesTerm[z_^k_., z_Symbol, n_] := When[n == k, 1, 0] /;
                                    IntegerQ[k] && FreeQ[k, z]
SeriesTerm[f_, z_Symbol, n_] := f When[n == 0, 1, 0] /; FreeQ[f, z]


    (* rational functions *)

SeriesTerm[f_. g_^k_Integer?Negative, z_Symbol, n_] :=
    Block[{ff = Apart[f g^k, z]},
        If[Head[ff] === Plus,
            Return[SeriesRat[#, z, n]& /@ ff],
            Return[SeriesRat[ff, z, n]] ]] /;
    PolynomialQ[f, z] && PolynomialQ[g, z] && !FreeQ[g, z]


SeriesRat[f_, z_Symbol, n_] := SeriesTerm[f, z, n] /; PolynomialQ[f, z]

SeriesRat[c_. z_^k_Integer?Negative, z_Symbol, n_] :=
    SeriesTerm[c z^k, z, n] /; FreeQ[c, z]
```

```
SeriesRat[f_. g_^k_Integer?Negative, z_Symbol, n_] :=

    Block[{ff = Expand[f], gg = Expand[g], poleList, poleSet,
           mp, denom = Expand[g^(-k)], i, deg, mod, modList,
           displace, temp, l, pr = precision, inverseRoot, kk},

         deg = Exponent[gg, z];
         If[useMod,
             l = CoefficientList[ff, z];
             mod = Exponent[gg, z, GCD];
             modList = Union[Mod[#, mod]& /@ (Select[Range[Length[l]],
                 (l[[#]] =!= 0)&] - 1)];
             If[Length[modList] == 1,
                 displace = First[modList],
                {displace, mod} = {0, 1}],
           {displace, mod} = {0, 1}];
        {ff, gg, denom} = {Expand[ff/z^displace], gg, denom} /.
             z -> z^(1/mod);
         deg = deg/mod;
         Which[precision === Automatic && deg > 2,
                  pr = Indeterminate,
               precision === Automatic && deg <= 2,
                  pr = Infinity];
         If[pr == Infinity && !FreeQ[Roots[gg == 0, z], Roots],
             pr = Indeterminate];
         Which[
             pr === Indeterminate,
                 poleList = Cancel[#[[1,2]]& /@ {ToRules[
                            NRoots[gg == 0, z] ]}];
                 poleSet = Union[poleList];
                 mp = -k Count[poleList, #]& /@ poleSet;
                 poleUse = poleSet;
             temp = SeriesTerm[Plus @@ Table[SeriesDivide[Horner[
             ff, {z, poleSet[[i]], mp[[i]]}], Drop[Horner[denom, {z,
             poleSet[[i]], 2 mp[[i]]}], mp[[i]]], mp[[i]] ] .
             Table[(z - poleUse[[i]])^(-j), {j, mp[[i]], 1, -1}],
             {i, Length[poleSet]}], z, Expand[(n-displace)/mod]],

             pr =!= Infinity,
                 poleList = Cancel[#[[1,2]]& /@ {ToRules[
                            NRoots[gg == 0, z, pr] ]}];
                 poleSet = Union[poleList];
                 mp = -k Count[poleList, #]& /@ poleSet;
                 poleUse = poleSet;
             temp = SeriesTerm[Plus @@ Table[SeriesDivide[Horner[
             ff, {z, poleSet[[i]], mp[[i]]}], Drop[Horner[denom, {z,
             poleSet[[i]], 2 mp[[i]]}], mp[[i]]], mp[[i]] ] .
             Table[(z - poleUse[[i]])^(-j), {j, mp[[i]], 1, -1}],
             {i, Length[poleSet]}], z, Expand[(n-displace)/mod]],

             pr === Infinity && deg <= 2,
                 poleList = Cancel[#[[1,2]]& /@ {ToRules[
                            Roots[gg == 0, z] ]}];
                 poleSet = Union[poleList];
                 mp = -k Count[poleList, #]& /@ poleSet;
                 inverseRoot = Expand[(1 - gg/(gg /. z -> 0))/z];
                 poleUse = ArgPi[Apart[
                 inverseRoot /. z -> #]]^(-1)& /@ poleSet;
             temp = SeriesTerm[Plus @@ Table[SeriesDivide[Horner[
             ff, {z, poleSet[[i]], mp[[i]]}], Drop[Horner[denom, {z,
             poleSet[[i]], 2 mp[[i]]}], mp[[i]]], mp[[i]] ] .
             Table[(z - poleUse[[i]])^(-j), {j, mp[[i]], 1, -1}],
             {i, Length[poleSet]}], z, Expand[(n-displace)/mod]],

             pr === Infinity && deg > 2,
                 poleList = Cancel[#[[1,2]]& /@ {ToRules[
                            Roots[CoefficientList[gg, z].
                            Table[z^kk, {kk, deg, 0, -1}] == 0, z] ]}];
                 poleSet = Union[poleList];
                 mp = -k Count[poleList, #]& /@ poleSet;
                 poleUse = Complex[Together[ReP[#]//.SimplifyCubic],
                           Together[ImP[#]//.SimplifyCubic]]& /@ poleSet;
                 poleUse = poleUse /. Complex[a_, 0] -> a;
                 If[FreeQ[poleUse, Complex], makeReal = False];
             temp = SeriesTerm[Plus @@ Table[SeriesDivide[Horner[
             ff /. z -> z / poleUse[[i]], {z, 1, mp[[i]]}],
             Drop[Horner[denom /. z -> z / poleUse[[i]], {z,
             1, 2 mp[[i]]}], mp[[i]]], mp[[i]] ] .
             Table[(z poleUse[[i]] - 1)^(-j), {j, mp[[i]], 1, -1}],
             {i, Length[poleSet]}], z, Expand[(n-displace)/mod]]
              ];
         temp = temp //. a_ c_If + b_ c_ -> (a + b) c;
        ((temp = temp /. SimplifyComplexE1) /; pr === Infinity;
          (temp = temp /. SimplifyComplex2) /; pr === Infinity && deg == 2;
          (temp = temp /. SimplifyComplex3) /; pr === Infinity && deg == 3;
         (temp = temp //. SimplifyComplex4) /; pr === Infinity && deg == 4;
          (temp = temp /. SimplifyComplexE2) /; pr === Infinity;
          (temp = temp //. SimplifyComplexN) /; pr =!= Infinity) /;
                                                                  makeReal;
           temp = temp /. Complex[a_, b_] :> a + b I;
           Return[If[Release[SymbolicMod[n, mod] == Mod[displace, mod]],
                Release[temp], 0]]
           ] /; PolynomialQ[f, z] && PolynomialQ[g, z] && !FreeQ[g, z]


    (* special functions *)

SeriesTerm[1 / Sqrt[a_ + b_.z_ + c_.z_^2], z_Symbol, n_] :=
    When[n >= 0,
        Sqrt[c/a]^n / Sqrt[a] LegendreP[n, -Sgn[a] b /(2 Sqrt[a c])],
            0] /; specialFunctions && FreeQ[{a,b,c}, z]

SeriesTerm[Power[a_ + b_.z_ + c_.z_^2, Rational[p_, 2]], z_Symbol, n_] :=
    SeriesTerm[Expand[(a + b z + c z^2)^((p + 1)/2)] /
        MySqrt[a + b z + c z^2], z, n] /;
                        p != -1 && specialFunctions && FreeQ[{a,b,c}, z]

SeriesTerm[1 / MySqrt[a_], z_Symbol, n_] :=
    SeriesTerm[1 / Sqrt[a], z, n]

SeriesTerm[(a_ + b_.z_)^m_ (c_ + d_.z_)^m_, z_Symbol, n_] :=
    SeriesTerm[Expand[(a + b z) (c + d z)]^m, z, n] /;
        specialFunctions && FreeQ[{a,b,c,d}, z] &&
        Expand[a c]^m == Expand[a^m c^m]

SeriesTerm[(a_ + b_.z_)^m_ (c_ + d_.z_)^m_, z_Symbol, n_] :=
   -SeriesTerm[Expand[(a + b z) (c + d z)]^m, z, n] /;
        specialFunctions && FreeQ[{a,b,c,d}, z] &&
        Expand[a c]^m == -Expand[a^m c^m]


    (* products with a rational function *)

SeriesTerm[a_ b_, z_Symbol, n_] :=
    Block[{temp, deg, i, m, bn = SeriesTerm[b, z, m], sd = sumDepth},
         Increment[sumDepth];
         Off[General::itervar];
         If[PolynomialQ[a, z],
             deg = Exponent[a, z];
             temp = CoefficientList[a, z] .
                 Table[bn /. m -> i, {i, n, n - deg, -1}] /. If -> When,
           temp = Sum[SeriesTerm[Apart[a, z], z, n - K[sd]] bn /.
               m -> K[sd], {K[sd],
              -PoleMultiplicity[b, {z, 0}], n + PoleMultiplicity[a,
              {z, 0}] }] ];
         On[General::itervar];
         Return[temp] ] /; SeparateProduct[a b, (PolynomialQ[#, z] ||
                               PolynomialQ[#^(-1), z])& ] == {a, b}

    (* general products *)

SeriesTerm[a_ b_, z_Symbol, n_] :=
    Block[{temp, sd = sumDepth},
         Increment[sumDepth];
         Off[General::itervar];
         temp = Sum[SeriesTerm[a, z, n - K[sd]] *
             SeriesTerm[b, z, K[sd]],
           {K[sd], -PoleMultiplicity[b, {z, 0}],
                  n + PoleMultiplicity[a, {z, 0}]}];
         On[General::itervar];
         Return[temp] ] /; SeparateProduct[a b, (PolynomialQ[#, z] ||
                               PolynomialQ[#^(-1), z])& ] == {1, a b}

SeriesTerm[a_^k_Integer?Positive, z_Symbol, n_] :=
    Block[{temp, sd = sumDepth},
         Increment[sumDepth];
         Off[General::itervar];
         temp = Sum[SeriesTerm[a, z, n - K[sd]] *
             SeriesTerm[a^(k - 1), z, K[sd]],
           {K[sd], -PoleMultiplicity[a^(k - 1), {z, 0}],
                  n + PoleMultiplicity[a, {z, 0}]}];
         On[General::itervar];
         Return[temp] ] /; !FreeQ[a, z]
    (* trinomials *)

SeriesTerm[(a_ + b_.z_ + c_.z_^2)^alpha_, z_Symbol, n_] :=
    Block[{temp},
         Off[General::itervar];
         temp = Sum[Binomial[alpha, K[sumDepth]] Binomial[K[sumDepth],
             n - K[sumDepth]] a^(alpha - K[sumDepth]) b^(2 K[sumDepth] -
             n) c^(n - K[sumDepth]), {K[sumDepth], 0, n}];
         On[General::itervar];
         ++sumDepth;
         Return[temp] ] /; FreeQ[{a,b,c}, z]


    (* derivatives *)

SeriesTerm[Derivative[k_][f_][z_], z_Symbol, n_] :=
    Pochhammer[n + 1, k] SeriesTerm[f[z], z, n + k]



(* HORNER and SERIES DIVIDE *)

Horner[p_, {z_, alpha_, m_}] :=
    Block[{qh = Expand[p], ih, kh, nh, ah},
        {nh, ah} = {Exponent[qh, z], CoefficientList[qh, z]};
        Do[ah[[ih]] = alpha ah[[ih + 1]] + ah[[ih]], {kh, 1, Min[nh, m]},
           {ih, nh, kh, -1}];
        Do[AppendTo[ah, 0], {ih, m - nh - 1}] /; m > nh + 1;
        Return[Take[ah, m]] ]


SeriesDivide[a_, b_, n_] :=
    Block[{is, cs = Range[n], ks},
        Do[cs[[is]] = (a[[is]] - Sum[cs[[ks]] b[[is - ks + 1]],
           {ks, 1, is - 1}])/b[[1]],
           {is, 1, n}];
        Return[cs] ]



(* SYMBOLIC MOD *)

SymbolicMod[n_, 1] = 0

SymbolicMod[n_?NumberQ, k_] := Mod[n, k]

SymbolicMod /: (SymbolicMod[n_Symbol + a_., 2] == b_) := EvenQ[n] /;
                                                          EvenQ[b - a]

SymbolicMod /: (SymbolicMod[n_Symbol + a_., 2] == b_) := OddQ[n] /;
                                                          OddQ[b - a]


(* PARTIAL FRACTIONS *)

PartialFractions[f_, z_, opts___Rule] :=
    Block[{ff = f, useApart},
        useApart = UseApart /.{opts} /.Options[PartialFractions];
        Which[useApart == Automatic,
                ff = Apart[ff],
           useApart == All,
                ff = Apart[ff, z],
           useApart =!= None,
                Message[PF::badopt, UseApart, useApart ] ];
        If[Head[ff] === Plus,
           Return[PartFrac[#, z, opts]& /@ ff],
           Return[PartFrac[f, z, opts]] ]];


PartFrac[f_, z_, opts___Rule] :=
    Block[{num = Numerator[f], den = Denominator[f], poleList, poleSet,
          mp, i, j, k, deg, temp, precision},
         precision = PrecisionST /.{opts} /. Options[PartialFractions];
        (deg = Exponent[den, z];
         Which[precision === Automatic && deg > 2,
                  precision = Indeterminate,
               precision === Automatic && deg <= 2,
                  precision = Infinity];
         poleList = Cancel[#[[1,2]]& /@ {ToRules[Which[
             precision == Indeterminate, NRoots[den == 0, z],
             precision =!= Infinity, NRoots[den == 0, z, precision],
             precision === Infinity, Roots[den == 0, z]] ]}];
         poleSet = Union[poleList];
```

```
            mp = Count[poleList, #]& /@ poleSet;
            If[precision === Infinity,
                poleUse = ArgPi[#]& /@ poleSet,
                poleUse = poleSet];
            temp = Plus @@ Table[SeriesDivide[Horner[num, {z,
                poleSet[[i]], mp[[i]]}], Drop[Horner[den, {z,
                poleSet[[i]], 2 mp[[i]]}], mp[[i]]], mp[[i]] ] .
                Table[(z - poleUse[[i]])^(-j), {j, mp[[i]], 1, -1}],
               {i, Length[poleSet]}]
            ) /; PolynomialQ[num, z] && PolynomialQ[den, z] ];


(* REALQ, REP, IMP, ANGLE, SGN, ABSV, and CONJUGATEQ *)

RealQ[x_] := True /; IntegerQ[x]
RealQ[x_Rational] = True
RealQ[x_Real] = True
RealQ[Pi] = True
RealQ[E] = True
RealQ[n_!] := True /; RealQ[n]

ReP[x_] := x /; RealQ[x]
ImP[x_] := 0 /; RealQ[x]

ReP[k_Integer x_] := k ReP[x]
ReP[k_Rational x_] := k ReP[x]
ReP[k_Real x_] := k ReP[x]

ImP[k_Integer x_] := k ImP[x]
ImP[k_Rational x_] := k ImP[x]
ImP[k_Real x_] := k ImP[x]

ReP[Complex[x_, y_]] := x
ImP[Complex[x_, y_]] := y

ReP[x_ + y_] := ReP[x] + ReP[y]
ImP[x_ + y_] := ImP[x] + ImP[y]

ReP[x_ Sqrt[y_?Positive]] :=  ReP[x] Sqrt[y]
ImP[x_ Sqrt[y_?Positive]] :=  ImP[x] Sqrt[y]
ReP[x_ Sqrt[y_?Negative]] := -ImP[x] Sqrt[-y]
ImP[x_ Sqrt[y_?Negative]] :=  ReP[x] Sqrt[-y]

ReP[x_ y_] := ReP[x] ReP[y] - ImP[x] ImP[y]
ImP[x_ y_] := ReP[x] ImP[y] + ImP[x] ReP[y]

ReP[x_?Positive ^n_] := x^n /; ImP[n] == 0
ImP[x_?Positive ^n_] := 0   /; ImP[n] == 0

ReP[x_?Negative ^n_Rational] := 0 /; IntegerQ[2n]
ImP[x_?Negative ^n_Rational] := (-x)^n (-1)^(n-1/2) /;
                                                   IntegerQ[2n]

ReP[1/x_] := ReP[x] / (ReP[x]^2 + ImP[x]^2)
ImP[1/x_] := -ImP[x] / (ReP[x]^2 + ImP[x]^2)

ReP[x_^Rational[p_,q_]] :=
    ReP[x^Quotient[p,q] x^(Mod[p,q]/q)] /;
                                 AbsV[p] >= AbsV[q]


ReP[x_^q_Rational] := AbsV[x]^q Cos[q Angle[x]] /;
                                       AbsV[q] <= 1
ImP[x_^q_Rational] := AbsV[x]^q Sin[q Angle[x]] /;
                                       AbsV[q] <= 1

ReP[E^x_] := Cos[ImP[x]] Exp[ReP[x]]
ImP[E^x_] := Sin[ImP[x]] Exp[ReP[x]]

ReP[x_^2] := ReP[x]^2 - ImP[x]^2
ImP[x_^2] := 2 ReP[x] ImP[x]

ReP[x_^3] := ReP[x]^3 - 3 ReP[x] ImP[x]^2
ImP[x_^3] := 3 ReP[x]^2 ImP[x] - ImP[x]^3

ReP[x_^n_Integer] := Block[{a, b},
        a = Round[n/2];
        b = n-a;
        ReP[x^a] ReP[x^b] - ImP[x^a] ImP[x^b]
        ] /; n != -1

ImP[x_^n_Integer] := Block[{a, b},
        a = Round[n/2];
        b = n-a;
        ReP[x^a] ImP[x^b] + ImP[x^a] ReP[x^b]
        ] /; n != -1


Angle[z_] :=
    Block[{x = ReP[z], y = ImP[z]},
        Which[x == 0, Pi/2 Sgn[y],
              y == 0, Pi/2 (1 - Sgn[x]),
              True,
              ArcTan[y/x] - Pi/2 (Sgn[x]-1) Sgn[y]] /;
                  FreeQ[{x,y}, ReP] && FreeQ[{x,y}, ImP]]

Sgn[0] = 0
Sgn[a_?Positive] = 1
Sgn[a_?Negative] = -1
Sgn[a_ b_] := Sgn[a] Sgn[b]
Sgn[a_^b_] := Sgn[a]^b
Sgn[Sqrt[a_]] = 1
Sgn[a_ + b_] := Sgn[Chop[N[a + b]]]


AbsV[0] = 0
AbsV[x_?Positive] := x
AbsV[x_?Negative] := -x
AbsV[x_ y_] := AbsV[x] AbsV[y]
AbsV[x_^n_] := AbsV[x]^n /; Head[n] =!= Complex
AbsV[x_] :=
    Block[{a = ReP[x], b = ImP[x]},
        Which[b == 0, Expand[a Sgn[a]],
              a == 0, Expand[b Sgn[b]],
              True, Sqrt[Expand[a^2 + b^2]]]]

ConjugateQ[a_, b_] :=
    Block[{l = PatternList[{a, b},
              _Symbol?(Context[#] =!= "System`"&) ]},
        Which[
            l === {},
               Chop[N[ReP[a]] - N[ReP[b]]] == 0 &&
               Chop[N[ImP[a]] + N[ImP[b]]] == 0,
            l =!= {},
               Expand[ReP[a] - ReP[b]] == 0 &&
               Expand[ImP[a] + ImP[b]] == 0 ] ]


(* SIMPLIFICATION RULES *)

SimplifyTrig = {
    Sin[x_. + r_Rational Pi] :> Cos[x] /; EvenQ[r-1/2],
    Sin[x_. + Pi] :> -Sin[x],
    Sin[x_. + n_Integer?OddQ Pi] :> -Sin[x],
    Sin[x_. + r_Rational Pi] :> -Cos[x] /; EvenQ[r-3/2],
    Sin[x_. + n_Integer?EvenQ Pi] :> Sin[x],
    Cos[x_. + r_Rational Pi] :> -Sin[x] /; EvenQ[r-1/2],
    Cos[x_. + Pi] :> -Cos[x],
    Cos[x_. + n_Integer?OddQ Pi] :> -Cos[x],
    Cos[x_. + r_Rational Pi] :> Sin[x] /; EvenQ[r-3/2],
    Cos[x_. + n_Integer?EvenQ Pi] :> Cos[x],
    Sin[a_?Negative b_.] :> -Sin[-a b],
    Cos[a_?Negative b_.] :>  Cos[-a b],
    Sin[a_Plus] :> -Sin[Expand[-a]] /; MatchQ[a[[1]], _?Negative _.],
    Cos[a_Plus] :>  Cos[Expand[-a]] /; MatchQ[a[[1]], _?Negative _.],
    ArcTan[a_?Negative b_.] :> -ArcTan[-a b],
    ArcTan[a_Plus] :> -ArcTan[Expand[-a]] /;
                                      MatchQ[a[[1]], _?Negative _.],
    ArcTan[Sqrt[3]] -> Pi/3,
    ArcTan[Sqrt[3]/3] -> Pi/6,
    ArcTan[1/Sqrt[3]] -> Pi/6}


SimplifySqrt = {a_^Rational[b_,2] :> a^((b-1)/2) Sqrt[a]}

CanonicRadicals = {
   (a_ + b_)^Rational[k_,n_] :> a (a + b)^(Mod[k, n] / n) +
       b (a + b)^(Mod[k, n] / n)   /; Quotient[k, n]==1,
    a_^Rational[k_,n_] :> a^(Mod[k, n]/n) Expand[a^Quotient[k, n]]}


SimplifyCubic = Join[SimplifyTrig, SimplifySqrt]


SimplifyBinomial = {
    Binomial[1/2, k_ + 1] :> (-4)^(-k) Binomial[2 k, k]/(2(k + 1)) /;
        Entails[Info[k], k != -1],
    Binomial[1/2, k_] -> (-4)^(-k) Binomial[2 k, k]/(1 - 2 k),
    Binomial[-1/2, k_] -> (-4)^(-k) Binomial[2 k, k],
    Binomial[k_ - 1/2, k_] -> 4^(-k) Binomial[2 k, k]}

SimplifySum = {
    Sum[0, {__}] -> 0,
    Sum[1, {k_, a_, b_}] :> When[a <= b, b - a + 1, 0],
    Sum[a_. b_, {k_, c__}] :> b Sum[a, {k, c}] /;
        FreeQ[b, k],
    Sum[If[k_ == m_, a_, b_] c_., {k_, m_, n_}] :>
       ((a c) /. k -> m) When[n >= m, 1, 0] +
        Sum[b c, {k, m + 1, n}],
    Sum[(If[-k_ + n_ >= 0, a_, b_] c_. + d_.) e_., {k_, m_, n_}] :>
         Sum[Release[Expand[a c e]], {k, m, n}] + Expand[d e],
    Sum[If[k_ >= m_, a_, b_] c_., {k_, m_, n_}] :>
         Sum[Release[Expand[a c]], {k, m, n}] }

SimplifyComplexE1 = {
    a_.I^c_ + d_.(-I)^c_ :> ArgPi[a] E^Expand[I Pi c/2] +
        ArgPi[d] E^Expand[-I Pi c/2] /; ConjugateQ[a, d],
    a_.E^b_ + d_.E^e_ :> 2 a Cos[ImP[b]] /;
       (ImP[a] == a - d == ReP[b] == b + e == 0),
    a_.E^b_ + d_.E^e_ :> -2 ImP[a] Sin[ImP[b]] /;
       (ReP[a] == a + d == ReP[b] == b + e == 0),
    a_.E^b_ + d_.E^e_ :> ArgPi[a] E^b + ArgPi[d] E^e}


SimplifyComplex2 = {
    a_.b_^c_ + d_.e_^c_ :>
        2 AbsV[a]AbsV[b]^c SimplifyCos[Cos[Angle[a] + c Angle[b]]] /;
        (ConjugateQ[a, d] && ConjugateQ[b, e] && ImP[c] == 0)}

SimplifyComplex3 = {
    a_.b_Complex^c_ + d_.e_Complex^c_ :>
        2 AbsV[a]AbsV[b]^c Cos[Angle[a] + c Angle[b]] /;
        (ConjugateQ[a, d] && ConjugateQ[b, e] && ImP[c] == 0)}

SimplifyComplex4 = {
    a_.b_Complex^c_ + d_.e_Complex^c_ :>
        2 AbsV[a]AbsV[b]^c SimplifyCos[Cos[Angle[a] + c Angle[b]]] /;
        (ConjugateQ[a, d] && ConjugateQ[b, e] && ImP[c] == 0)}

SimplifyComplexE2 = {
    a_.E^b_ + d_.E^e_ :> Block[{c = ImP[b]},
        2 ReP[a] Cos[c] - 2 ImP[a] Sin[c] /;
        (ConjugateQ[a, d] && ConjugateQ[b, e] && ReP[b] == 0)]}


SimplifyComplexN = {
    a_.b_Complex^c_ + d_.e_Complex^c_ :>
        2 Abs[a]Abs[b]^c Cos[Arg[a] + c Arg[b]] /;
        (ConjugateQ[a, d] && ConjugateQ[b, e] && ImP[c] == 0)}

SimplifyCos[x_] :=
    Block[{x1 = (x /. Cos[a_] :> Cos[Expand[a]]) //. SimplifyTrig,
           x2 =  x //. SimplifyTrig },
        If[LeafCount[x1] <= LeafCount[x2], x1, x2 ]]

SimplifyFloat = {1. -> 1, -1. -> -1, 0. -> 0}


(* FACTORIAL SIMPLIFY *)

FactorialSimplify[expr_] := PostSimplify[DeFact[expr]]


DeFact[expr_] :=

    Block[{kin, j, sub, diff, t, tt, arg = {}, ex = expr},

        While[!FreeQ[ex, Factorial],
            t = First[Position[ex, Factorial[_]]];
            tt = First[Part @@ Prepend[t, ex]];
            kin = Scan[If[IntegerQ[Expand[# - tt]],
                Return[Fact[#]] ]&, arg];
            If[kin === Null,
                AppendTo[arg, tt]; sub = Fact[tt],
                diff = Expand[First[kin] - tt];
                If[diff > 0,
                    sub = kin / Product[tt + j, {j, diff}],
                    sub = kin Product[tt - j + 1, {j, -diff}] ] ];
```

```
          ex = MapAt[sub&, ex, {t}] ];
       Factor[ex] /. Fact -> Factorial ]


PostSimplify[expr_] :=

    Block[{ex = expr /. n! :> Expand[n]!, i},

       ex //.
          {n_!^k_. m_!^l_.:> Product[m + i, {i, n - m}]^k /;
                                    Expand[n - m] > 0  &&  k + l == 0,

           n_!^k_. m_^l_. :> Expand[n + 1]!^k Expand[n + 1]^(l - k) /;
                        Expand[m - n - 1] == 0  && Sign[k] == Sign[l],

           n_!^k_. m_^l_. :> (-1)^k Expand[n + 1]!^k m^(l - k) /;
                         Expand[m + n + 1] == 0  && Sign[k] == Sign[l],

           n_!^k_. n_^l_. :> Expand[n - 1]!^k  /;  Expand[k + l] == 0,

           n_!^k_. m_^l_. :> (-1)^k Expand[n - 1]!^k /;
                              Expand[m + n] == Expand[k + l] == 0 } ]


(** (Utilities.m) **)


(* ARGPI *)

ArgPi[z_] :=
    Block[{fraction = Rationalize[N[Arg[z]/Pi]]},
       If[Head[fraction] === Rational,
          Return[AbsV[z] E^(fraction Pi I)],
          Return[z]] ]


(* ENTAILS *)

Entails[a_, a_] = True

Entails[a___ && b_ && c___, b_] = True

Entails[False, a_] = True

Entails[a_, b_] :=
    Block[{l = Union[PatternList[{a, b},
                       _Symbol?(Context[#] =!= "System`"&) ]]},
       IneqSolve[Implies[a, b], First[l]] ===
              {Interval[-Infinity, Infinity]} /; Length[l] == 1]


(* FIRST POS and FREE L *)

FirstPos[l_, pattern_] :=
    SafeFirst[Select[Range[Length[l]], MatchQ[l[[#]], pattern]& ]]

FreeL[expr_, l_] := And @@ (FreeQ[expr, #]& /@ l)


(* ISOLVE, INEQSOLVE, LEQ *)

IneqSolve[x_ + a_ > b_, x_] := IneqSolve[x > b - a, x]

IneqSolve[x_ + a_ >= b_, x_] := IneqSolve[x >= b - a, x]

IneqSolve[x_ > n_Integer, x_] := {Interval[n+1, Infinity]}

IneqSolve[x_ >= n_Integer, x_] := {Interval[n, Infinity]}

IneqSolve[a_, x_] := a /; Length[Union[PatternList[a,
                     _Symbol?(Context[#] =!= "System`"&) ]]] > 1

IneqSolve[a_ && b_, x_] :=
    Block[{aa = IneqSolve[a, x],
          bb = IneqSolve[b, x],
          merge, accum, begin, nn},
       If[FreeQ[aa, IneqSolve] && FreeQ[bb, IneqSolve],
       aa = Append[aa, {}];
       bb = Append[bb, {}];
       merge = Sort[Join[
          Thread[{Join @@ (aa /. Interval -> List),
                 Join @@ (aa /. Interval[_,_] -> {-1,1}) }],
            Thread[{Join @@ (bb /. Interval -> List),
                    Join @@ (bb /. Interval[_,_] -> {-1,1}) }] ],
                    LEQ];
        accum = Thread[{First /@ merge,
                        Accumulate[Plus, Last /@ merge]}];
        begin = Join @@ Append[Position[accum, {_,-2}], {}];
        Return[(Interval @@ # & /@ Thread[{begin, begin + 1}]) /.
                            nn_Integer :> First[accum[[nn]]]],
        Return[aa && bb] ]]

IneqSolve[!a_, x_] :=
    Block[{aa = IneqSolve[a, x], merge, c, d},
        If[FreeQ[aa, IneqSolve],
            aa = Append[aa, {}];
            merge = Join @@ (aa /. Interval[c_, d_] -> {c-1, d+1});
            If[merge === {},
                merge = {-Infinity, Infinity},
                If[First[merge] === -Infinity,
                    merge = Rest[merge],
                    PrependTo[merge, -Infinity] ];
                If[Last[merge] === Infinity,
                    merge = Drop[merge,-1],
                    AppendTo[merge, Infinity] ] ];
            Return[Interval @@ # & /@ Partition[merge, 2]],
          Return[!aa]] ]

IneqSolve[a_ || b_, x_] :=
    Block[{aa = IneqSolve[!(!a && !b), x]},
        If[FreeQ[aa, IneqSolve],
            Return[aa],
          Return[IneqSolve[a, x] || IneqSolve[b, x]]] ]

IneqSolve[Inequality[a_, b_, c_, d_, e__], x_] :=
    IneqSolve[And @@ (Inequality @@ # & /@
        Partition[{a,b,c,d,e},3,2]), x]


IneqSolve[h_[a_, b_, c__], x_] :=
    IneqSolve[And @@ (h @@ # & /@
            Partition[{a,b,c},2,1]), x] /;
    MemberQ[{Less, LessEqual, Greater, GreaterEqual, Equal,
           Unequal}, h]

IneqSolve[ineq: h_[a_, b_], x_] :=
    Block[{lhs = Together[a - b],
          num, den, zeroList, poleList, points, merge, c, d},

       {num, den} = {Numerator[lhs], Denominator[lhs]};
        Off[Roots::neq];
        zeroList = If[FreeQ[num, x], {},
            #[[1,2]]& /@ {ToRules[NRoots[num == 0, x]]} ];
        poleList = If[FreeQ[den, x], {},
            #[[1,2]]& /@ {ToRules[NRoots[den == 0, x]]} ];
        On[Roots::neq];
        zeroList = Select[zeroList, (Im[#] == 0)&] // Union;
        poleList = Select[poleList, (Im[#] == 0)&] // Union;
        points = Union[zeroList, poleList];
        merge = Partition[Append[Prepend[points,-Infinity],Infinity],
            2, 1];
        merge = Select[merge, (ineq /. x :>
                Which[# === {-Infinity, Infinity}, 0,
                      #[[1]] === -Infinity, #[[2]] - 1,
                      #[[2]] ===  Infinity, #[[1]] + 1,
                      True, (Plus @@ #)/2 ] )&];
        merge = (Interval @@ # & /@ merge) /.
            Interval[c_, d_] :> Interval[Ceiling[c], Floor[d]];
        merge = merge /. {Ceiling[-Infinity] -> -Infinity,
                          Floor[Infinity] -> Infinity};
        merge = Select[merge, #[[1]] <= #[[2]]&];
        zeroList = Sort[Join[Floor /@ zeroList, Ceiling /@ zeroList]];
        zeroList = Select[zeroList, ((den /. x->#) != 0)&];
        zeroList = Select[zeroList, (ineq /. x->#)&];
        merge = Sort[Join[merge, Thread[Interval[zeroList, zeroList]]],
            LEQ];
        merge = merge /.
            Interval[c_?((NumberQ[#] && ((den /. x:>#) == 0 ||
                !ineq/.x->#))&), d_] :> Interval[c+1, d];
        merge = merge /.
            Interval[c_, d_?((NumberQ[#] && ((den /. x:>#) == 0 ||
                !ineq/.x->#))&)] :> Interval[c, d-1];
        merge = Select[merge, #[[1]] <= #[[2]]&];
        args = List @@ (Join @@ Append[merge, Interval[]]);
        If[args === {}, {},
            {first, last} = {First[args], Last[args]};
             Interval @@ # & /@ Partition[Append[Prepend[Join @@
                 Append[Select[Partition[Drop[Rest[args], -1], 2],
                 (#[[2]] - #[[1]] > 1)&], {}], first], last], 2] ] ]  /;
        MemberQ[
           {Less, LessEqual, Greater, GreaterEqual, Equal, Unequal}, h]


ISolve[expr_] :=
    Block[{l = UserSymbols[expr], n, temp, p, nn},
        If[Length[l] == 1,
            n = First[l];
            temp = IneqSolve[expr /. n -> nn, nn] /. nn -> n;
            temp = temp /. List[p___Interval] :> Or[p];
            temp = temp /.
              {Interval[-Infinity, Infinity] -> True,
               Interval[-Infinity, b_] -> n <= b,
               Interval[a_, Infinity] -> n >= a,
               Interval[a_, a_] -> n == a,
               Interval[a_, b_] -> a <= n <= b};
            temp = temp /. IneqSolve[a_, n_] -> a;
            Return[temp],
          Return[expr]] ]

LEQ[h_[a_,b_], h_[c_,d_]] := (a < c) || (a === c && b <= d)


(* INFO *)

Info[_?NumberQ] = True
Info[_Symbol] = True
Info[_[a___]] := And @@ (Info /@ {a})


(* LEADING COEF *)

LeadingCoef[poly_, x_] := If[poly === 0, 0,
    Coefficient[poly, x, Exponent[poly, x]] ]


(* MAKE LIST *)

MakeList[expr_, head_:List] :=
    If[Head[expr] === head, List @@ expr, List @ expr]


(* MAKE TRINOMIAL and LIST COMPLEMENT *)

MakeTrinomial[a: Binomial[b_, c_] Binomial[d_, e_]] :=
    Block[{l1 = {b, d},
           l2 = {c, b - c, e, d - e}, l},
        l = Intersection[l1, l2];
        If[Length[l] != 1, Return[a]];
        l1 = ListComplement[l1, l];
        l2 = ListComplement[l2, l];
        If[{Plus @@ l2} != l1, Return[a]];
        Return[Sort[Multinomial @@ l2]] ]

MakeTrinomial[a_] := a


ListComplement[l1_List, l2_List] :=
    Block[{l = l1},
        If[MemberQ[l, #],
            l = Drop[l, {FirstPos[l, #]}] ]& /@ l2;
         Return[l] ]


(* PARSING ROUTINE *)

Parse[list1_, list2_, n_] :=
    Block[{l1 = MakeList[Release @@ (Hold[list1] /. Condition -> Pair)],
           unknowns = MakeList[list2],
           eqns, conds, recur, extra, range, startValues, aa, kk,
           lo, hi, check},

        unknowns = Head /@ unknowns;

        check = If[Head[#] === Pair, #[[1]], #]& /@ l1;
        If[!MatchQ[check, {__Equal}],
            Message[Parse::eqn, check];
            Return[Fail] ];
        eqns =  Select[ll, !FreeQ[#, n]&];
        conds = Select[ll,  FreeQ[#, n]&];

            (* make  n >= 0  the default range of validity *)

        eqns = If[Head[#] =!= Pair, Pair[#, n >= 0], #]& /@ eqns;

            (* solve inequalities, and separate recurrences from
               initial conditions *)

        eqns = Pair[#[[1]], IneqSolve[#[[2]], n]]& /@ eqns;
        recur = Select[eqns, !FreeQ[#, Infinity]&];
        extra = Union[Select[eqns,  FreeQ[#, Infinity]&],
                        Pair[#[[1]], Drop[#[[2]], -1]]& /@ recur];
        recur = Pair[#[[1]], Last[#[[2]]][[1]]]& /@ recur;
        extra = Select[extra, #[[2]] =!= {} &];
        extra = Union @@ (Thread /@ extra);
        conds = Union[conds, Union @@ (Table[#[[1]], {n, #[[2,1]],
            #[[2,2]]}]& /@ extra) ];

            (* shift n so that all recurrences will be valid
               for  n >= 0  *)

        recur = (#[[1]] /. (n -> n + #[[2]]) )& /@ recur;

            (* determine starting values of n for all sequences *)

       {recur, conds} = {recur, conds} //.
            Sum[a_, {k_, m_}] :> Sum[a, {k, 1, m}];
       {recur, conds} = {recur, conds} /.
            {EvenQ[a_] :> SymbolicMod[a, 2] == 0,
             OddQ[a_]  :> SymbolicMod[a, 2] == 1,
             Mod -> SymbolicMod};
        range = {conds, recur /. (Sum[aa_, {kk_, lo_, hi_}] :>
            {aa /. kk -> lo, aa /. kk -> hi} )};
        startValues = Min[First /@ (PatternList[range, #[_]] /.
            n -> 0)]& /@ unknowns;

        Return[{recur, conds, unknowns, startValues}] ]


(* PATTERN LIST *)

PatternList[expr_?(FreeQ[#, Rule]&), pattern_] :=
    Apply[Part, Prepend[#, expr]]& /@ Position[expr, pattern]

PList[expr_, pattern_] :=
    Block[{xpr = expr /. ((a_ -> b_) -> {a,b})},
        Apply[Part, Prepend[#, xpr]]& /@ Position[xpr, pattern]]


(* POLE MULTIPLICITY *)

PoleMultiplicity[f_, {z_, a_}] :=
    Block[{pom = SafeSeries[f /. z -> z + a, {z, 0, 0}]},
        If[!FreeL[pom, {Fail, Series}], Return[Infinity],
           pom = Normal[pom];
           If[pom == 0, Return[0]];
           Return[Max[0, Exponent[Expand[pom /. z -> 1/z], z]]] ]]


(* RESET *)

Reset[recur_, conds_, unknowns_, startValues_, s0_] :=
    Block[{start},
       (Release[start /@ unknowns]) = startValues;
        Return[{recur, conds} /.
            aa_?(MemberQ[unknowns,#]&)[kk_] :>
            aa[kk + s0 - start[aa]] ]]


(* SAFE FIRST, SAFE SERIES *)

SafeFirst[l_] :=
    If[Length[l] == 0, Null, First[l]]

SafeSeries[f_, {z_, a_, n_Integer}] :=
    Block[{poles = z /. Solve[Denominator[f] == 0, z,
           InverseFunctions -> True], ord, tem},
        ord = Max[n, Count[poles, a]];
        Off[General::dby0, General::bvar, Series::arg, SeriesData::csa];
        tem = Check[Series[f, {z, a, ord}], Fail, Series::esss];
        On[General::dby0, General::bvar, Series::arg, SeriesData::csa];
        If[tem =!= Fail, tem = Series[tem, {z, a, n}]];
        Return[tem] ]


(* SEPARATE PRODUCT *)

SeparateProduct[p_Times, predicate_] :=
   {Select[p, predicate], Select[p, !predicate[#]&]}


(* USER SYMBOLS *)

UserSymbols[expr_] :=
    Block[{h = If[Length[expr] == 0, expr, Head[expr]]},
         Which[!FreeQ[h,
                  _Symbol?(Context[#] =!= "System`"&)],
                   {expr},
                 Length[expr] == 0,
                   {},
                 True,
                   Union @@ (UserSymbols /@ (List @@ expr))]]


(* TTQ, WHEN *)

TTQ[True] := True
TTQ[a_] := Entails[Info[a], a]

When[cond_, a_, b_] :=
    Which[Entails[Info[cond],  cond], a,
          Entails[Info[cond], !cond], b,
          True, If[cond, a, b]]



End[]

(* Protect[RSolve, PowerSum, ExponentialPowerSum, GeneratingFunction,
ExponentialGeneratingFunction, Gf, EGf, HSolve, HypergeometricF, K,
SeriesTerm, SymbolicMod, RealQ, ReP, ImP, ConjugateQ, SimplifySum,
SimplifyTrig, SimplifyComplexE1, SimplifyComplexE2, SimplifyComplex2,
SimplifyComplex3, SimplifyComplex4, SimplifyComplexN, FactorialSimplify,
PartialFractions, ArgPi, Entails, FirstPos, FreeL, Info, ISolve,
LeadingCoef, ListComplement, MakeList, MakeTrinomial, PatternList,
PoleMultiplicity, Reset, SafeFirst, SafeSeries, TTQ, UserSymbols, When,
Methods, MethodGF, MethodEGF, PrecisionHS, PrecisionST, UseApart, UseMod,
MakeReal, SpecialFunctions] *)

EndPackage[]
```

Bibliography

[Abr74] S. A. Abramov. Reshenie lineĭnykh konechno-raznostnykh uravneniĭ s postoi͡annymi koefficientami v pole racional'nykh funkciĭ. *Zh. vychisl. matem. i matem. fiz.*, 14:1067 – 1070, 1974.

[Abr89a] S. A. Abramov. Racional'nye reshenii͡a lineĭnykh differencial'nykh i raznostnykh uravneniĭ s polinomial'nymi koefficientami. *Zh. vychisl. matem. i matem. fiz.*, 29:1611 – 1620, 1989.

[Abr89b] S. A. Abramov. Zadachi komp'i͡uternoĭ algebry, svi͡azannye s poiskom polinomial'nykh resheniĭ lineĭnykh differencial'nykh i raznostnykh uravneniĭ. *Vestn. MGU. Ser. 15. Vychisl. matem. i kibernetika*, (3):56 – 60, 1989.

[AP91] George E. Andrews and Peter Paule. Some questions concerning computer-generated proofs of double-sum identity. Technical report, RISC Linz, Johannes Kepler University, 1991.

[Bal80] F. Baldassari. On second order linear differential equations with algebraic solutions on algebraic curves. *Amer. J. Math.*, 102:517 – 535, 1980.

[BD79] F. Baldassari and B. Dwork. On second order linear differential equations with algebraic solutions. *Amer. J. Math.*, 101:42 – 76, 1979.

[BGMP86] D. Babić, A. Graovac, B. Mohar, and T. Pisanski. The matching polynomial of a polygraph. *Discr. Appl. Math.*, 15:11 – 24, 1986.

[Bia62] A. Białynicki-Birula. On Galois theory of fields with operators. *Amer. J. Math.*, 84:89 – 109, 1962.

[Bir67] Garrett Birkhoff. *Lattice Theory*, volume 25 of *Colloquium Publications of the American Mathematical Society.* American Mathematical Society, Providence, RI, 3rd edition, 1967.

[BO78] Carl M. Bender and Steven A. Orszag. *Advanced Mathematical Methods for Scientists and Engineers*, volume 17 of *International Series in Pure and Applied Mathematics.* McGraw-Hill, New York, 1978.

[Boo58] George Boole. *Calculus of Finite Differences.* Chelsea Publ. Co., New York, 4th edition, 1958.

[Bra66] Louis Brand. *Differential and Difference Equations.* John Wiley & Sons, New York, 1966.

[Bro90] Manuel Bronstein. Integration of elementary functions. *J. Symbolic Computation*, 9(4):117 – 173, 1990.

[Buc65] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, University of Innsbruck, 1965.

[Buc85] Bruno Buchberger. Gröbner bases: an algorithmic method in polynomial ideal theory. In N.K. Bose, editor, *Multidimensional Systems Theory*, pages 184 – 232. D. Reidel Publishing Co., Hingham, Massachusetts, 1985.

[C$^+$85] B.W. Char et al. *Maple User's Guide.* WATCOM Publications Ltd., Waterloo, Ontario, fourth edition, 1985.

[Cel84] Pedro Celis. Corrections and errors in john ivie's some MACSYMA programs for solving recurrence relations [john ivie, *acm trans. math. softw. 4*, 1 (march 1978), 24 – 33]. *ACM Transactions on Mathematical Software*, 10(4):477 – 478, 1984.

[Cho65] Frank Chorlton. *Differential and Difference Equations.* Van Nostrand, London, 1965.

[CK77] Jacques Cohen and Joel Katcoff. Symbolic solution of finite–difference equations. *ACM Transactions on Mathematical Software*, 3(3):261 – 271, 1977.

[CN58] Edward J. Cogan and Robert Z. Norman. *Handbook of Calculus, Difference and Differential Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1958.

[Coh65] Richard M. Cohn. *Difference Algebra*, volume 17 of *Interscience Tracts in Pure and Applied Mathematics.* Interscience Publishers, New York – London – Sydney, 1965.

[Com74] Louis Comtet. *Advanced Combinatorics: The Art of Finite and Infinite Expansions.* D. Reidel Publ. Co., Dordrecht–Holland / Boston–U.S.A., 1974. Analyse Combinatoire, Tomes I et II, first published in 1970 by Presses Universitaires de France, Paris, translated from the French by J. W. Nienhuys..

[Dic13] L.E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Am. J. Math.*, 35:413 – 426, 1913.

[DS86] James H. Davenport and Michael F. Singer. Elementary and Liouvillian solutions of linear differential equations. *J. Symbolic Computation*, 2:237 – 260, 1986.

[FO90] Philippe Flajolet and Andrew M. Odlyzko. Singularity analysis of generating functions. *SIAM J. Discrete Math.*, 3:216 – 240, 1990.

[Foa74] D. Foata. *La série génératrice exponentielle dans les problèmes d'énumeration.* Presses Univ. Montréal, Montréal, 1974.

[For48] Tomlinson Fort. *Finite Differences and Difference Equations in the Real Domain.* Clarendon Press, Oxford, 1948.

[Fra63] Charles H. Franke. Picard–Vessiot theory of linear homogeneous difference equations. *Trans. Amer. Math. Soc.*, 108:491 – 515, 1963.

[Fra66] Charles H. Franke. Solvability of linear homogeneous difference equations by elementary operations. *Proc. Amer. Math. Soc.*, 17:240 – 246, 1966.

[Fra67] Charles H. Franke. A note on the Galois theory of linear homogeneous difference equations. *Proc. Amer. Math. Soc.*, 18:548 – 551, 1967.

[Fra69] Charles H. Franke. The Galois correspondence for linear homogeneous difference equations. *Proc. Amer. Math. Soc.*, 21:397 – 401, 1969.

[Fra71] Charles H. Franke. Linearly reducible linear difference operators. *Æquationes Math.*, 6:188 – 194, 1971.

[Fra73] Charles H. Franke. Reducible linear difference operators. *Æquationes Math.*, 9:136 – 144, 1973.

[Fra74] Charles H. Franke. A characterization of linear difference equations which are solvable by elementary operations. *Æquationes Math.*, 10:97 – 104, 1974.

[FSZ89] Philippe Flajolet, Bruno Salvy, and Paul Zimmermann. Lambda-Upsilon-Omega: An assistant algorithms analyzer. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, number 357 in Lecture Notes in Computer Science, pages 201 – 212, 1989. Proceedings AAECC-6, Rome, July 1988.

[Gel71] A.O. Gel'fond. *Calculus of Finite Differences.* Hinduston, Delhi, India, 1971.

[GKP89] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics.* Addison-Wesley, Reading, Massachusetts, 1989.

[Gol58] Samuel Goldberg. *Introduction to Difference Equations.* John Wiley & Sons, New York, 1958.

[Gos77] R. William Gosper, Jr. Indefinite hypergeometric sums in MACSYMA. In *Proc. MACSYMA Users Conference*, pages 237 – 252, Berkeley CA, 1977.

[Gos78] R. William Gosper, Jr. Decision procedure for indefinite hypergeometric summation. *Proc. Natl. Acad. Sci. USA*, 75(1):40 – 42, 1978.

[Gou72] H. Gould. *Combinatorial Identities.* Morgantown Printing Company, Morgantown, W.Va., 1972.

[GR64] Sergei K. Godunov and V.S. Ryabenki. *Theory of Difference Schemes.* North-Holland, Amsterdam, 1964.

[Hal86] Marshall Hall, Jr. *Combinatorial Theory.* John Wiley & Sons, New York, second edition, 1986.

[Hen74] Peter Henrici. *Applied and Computational Complex Analysis*, volume 1: Power Series – Integration – Conformal Mapping – Location of Zeros of *Pure and Applied Mathematics – A Wiley-Interscience Series of Texts, Monographs, and Tracts.* Wiley–Interscience, 1974.

[HP73] Frank Harary and E. Palmer. *Graphical Enumeration.* Academic Press, New York, 1973.

[Imm81] Geertrui K. Immink. *Asymptotics of Analytic Difference Equations.* Springer-Verlag, New York – Heidelberg – Berlin, 1981.

[Ivi78] John Ivie. Some MACSYMA programs for solving recurrence relations. *ACM Transactions on Mathematical Software*, 4(1):24 – 33, 1978.

[Jen84] Richard D. Jenks. *A Primer: 11 Keys to the New Scratchpad*, volume 174 of *Lecture Notes in Computer Science*, pages 123 – 147. Springer–Verlag, New York – Heidelberg – Berlin, 1984.

[Jor60] Charles Jordan. *Calculus of Finite Differences.* Chelsea Publ. Co., New York, second edition, 1960.

[Jur64] Eliahu I. Jury. *Theory and Application of the z-Transform.* John Wiley & Sons, New York, 1964.

[Kac85] T. Kaczorek. *Two-Dimensional Linear Systems*, volume 68 of *Lecture Notes in Control and Information Sciences.* Springer–Verlag, New York – Heidelberg – Berlin, 1985.

[Kap57] Irving Kaplansky. *An Introduction to Differential Algebra*, volume 1251 of *Actualités Scientifiques et Industrielles.* Hermann, Paris, 1957.

[Kar81] Michael Karr. Summation in finite terms. *J. Assoc. Comp. Mach.*, 28(2):305 – 350, 1981.

[Kar85] Michael Karr. Theory of summation in finite terms. *J. Symbolic Computation*, 1:303 – 315, 1985.

[Ken61] M.G. Kendall. *A Course in the Geometry of n Dimensions.* Hafner Publ. Co., New York, 1961.

[Knu68] Donald E. Knuth. *The Art of Computer Programming, volume 1: Fundamental Algorithms.* Addison-Wesley, Reading, Massachusetts, 1968.

[Kol73] E. R. Kolchin. *Differential Algebra and Algebraic Groups*, volume 54 of *Pure and Applied Mathematics.* Academic Press, New York and London, 1973.

[Kov86] Jerald J. Kovacic. An algorithm for solving second order linear homogeneous differential equations. *J. Symbolic Computation*, 2:3 – 43, 1986.

[Lan58] Serge Lang. *Introduction to Algebraic Geometry*, volume 5 of *Interscience Tracts in Pure and Applied Mathematics.* Interscience Publishers, New York – London, 1958.

[Lip88] Leonard Lipshitz. The diagonal of a D-finite power series is D-finite. *J. Algebra*, 113:373 – 378, 1988.

[Lip89] Leonard Lipshitz. D-finite power series. *J. Algebra*, 122:353 – 373, 1989.

[Liu68] C.L. Liu. *Introduction to Combinatorial Mathematics.* McGraw–Hill, New York, 1968.

[LL61] Hyman Levy and F. Lessman. *Finite Difference Equations.* Macmillan, New York, 1961.

[LP90] Peter Luksch and Marko Petkovšek. An explicit formula for $!FM(1+1+n)!$. *Order*, 6:319 – 324, 1990.

[Lue80] George S. Lueker. Some techniques for solving recurrences. *Computing Surveys*, 12(4):425 – 436, 1980.

[Luk87] Peter Luksch. The cardinality of $FM(1+1+n)$. *Order*, 4:15 – 30, 1987.

[Mac15] Percy A. MacMahon. *Combinatory Analysis*, volume 1. Cambridge University Press, 1915.

[Mac16] Percy A. MacMahon. *Combinatory Analysis*, volume 2. Cambridge University Press, 1916.

[Mac89] M. A. H. MacCallum. An ordinary differential equation solver for REDUCE. In Patrizia Gianni, editor, *Symbolic and Algebraic Computation*, volume 358 of *Lecture Notes in Computer Science*, pages 196 – 205. Springer–Verlag, New York – Heidelberg – Berlin, 1989. International Symposium ISSAC '88, Rome, Italy, July 1988.

[Man74] Zohar Manna. *Mathematical Theory of Computation.* McGraw–Hill Book Co., New York, 1974.

[McB71] Elna B. McBride. *Obtaining Generating Functions*, volume 21 of *Springer Tracts in Natural Philosophy.* Springer-Verlag, New York – Heidelberg – Berlin, 1971.

[Mes59] Herbert Meschkowski. *Differenzengleichungen*, volume 14 of *Studia Mathematica / Mathematische Lehrbücher.* Vandenhoeck & Ruprecht, Göttingen, 1959.

[Mic87] Ronald E. Mickens. *Difference Equations.* Van Nostrand Reinhold, New York, 1987.

[Mil66] Kenneth S. Miller. *An Introduction to the Calculus of Finite Differences and Difference Equations.* Dover, New York, 1966.

[Mil68] Kenneth S. Miller. *Linear Difference Equations.* W.A. Benjamin, Inc., New York – Amsterdam, 1968.

[Moe77] R. Moenck. On computing closed forms for summations. In *Proc. MACSYMA Users Conference*, pages 225 – 236, Berkeley, CA, 1977.

[MOW84] O. Martin, A. M. Odlyzko, and S. Wolfram. Algebraic properties of cellular automata. *Commun. Math. Phys.*, 93:219 – 258, 1984.

[MT33] L.M. Milne-Thomson. *The Calculus of Finite Differences.* Macmillan and Co., Ltd., London, 1933.

[Pet90] Marko Petkovšek. Solving difference equations in Mathematica (Summary). In *1990 Mathematica Conference: Conference Guide*, Redwood City, CA, 1990.

[PW85] Richard Pavelle and Paul S. Wang. MACSYMA from F to G. *J. Symbolic Computation*, 1:69 – 100, 1985.

[Ric54] C.H. Richardson. *An Introduction to the Calculus of Finite Differences.* D. van Nostrand Co., Princeton, NJ, 1954.

[Rio68] John Riordan. *Combinatorial Identities.* John Wiley & Sons, New York, 1968.

[Ris69] R. Risch. The problem of integration in finite terms. *Transactions American Mathematical Society*, 139:167 – 189, 1969.

[Rit32] Joseph Fels Ritt. *Differential Equations from the Algebraic Standpoint*, volume 14 of *American Mathematical Society Colloquium Publications.* American Mathematical Society, New York, 1932.

[Sal89] Bruno Salvy. Examples of automatic asymptotic expansions. Preprint, 1989.

[Sin] Michael F. Singer. Formal solutions of differential equations. To appear.

[Sin80] Michael F. Singer. Algebraic solutions of n-th order homogeneous linear differential equations. In *Proceedings of the 1979 Queens University Conference on Number Theory*, volume 54 of *Queens Papers in Pure and Applied Mathematics*, pages 379 – 420, 1980.

[Sin81] Michael F. Singer. Liouvillian solutions of n-th order homogeneous linear differential equations. *Amer. J. Math.*, 103:661 – 682, 1981.

[Sin85] Michael F. Singer. Solving homogeneous linear differential equations in terms of second order linear differential equations. *Amer. J. Math.*, 107:663 – 696, 1985.

[Sin90] Michael F. Singer. An outline of differential Galois theory. In E. Tournier, editor, *Computer Algebra and Differential Equations*, Computational Mathematics and Applications, pages 3 – 57. Academic Press, 1989 or 1990.

[SLL89] Dominic Y. Savio, Edmund A. Lamagna, and Shing-Min Liu. Summation of harmonic numbers. In *Computers and Mathematics 1989*, pages 12 – 20, MIT, Boston MA, 1989.

[Slo73] N. Sloane. *Handbook of Integer Sequences.* Academic Press, New York, 1973.

[Spi71] Murray R. Spiegel. *Calculus of Finite Differences and Difference Equations.* Schaum's Outline Series. McGraw–Hill Book Co., New York, 1971.

[Sta78] Richard P. Stanley. Generating functions. In Gian-Carlo Rota, editor, *MAA Studies in Combinatorics*, volume 17 of *Studies in Mathematics*, pages 100 – 141. The Mathematical Association of America, 1978.

[Sta80] Richard P. Stanley. Differentiably finite power series. *European J. Combin.*, 1:175 – 188, 1980.

[Sua89] Rodolfo Suarez. Difference equations and a principle of double induction. *Math. Magazine*, 62:334 – 339, 1989.

[SW70] Josef Stoer and Christoph Witzgall. *Convexity and Optimization in Finite Dimensions I*, volume 163 of *Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen.* Springer-Verlag, New York – Heidelberg – Berlin, 1970.

[Tra84] Barry M. Trager. *Integration of Algebraic Functions.* PhD thesis, Massachusetts Institute of Technology, 1984.

[Tuc80] Alan Tucker. *Applied Combinatorics.* John Wiley & Sons, New York, 1980.

[vdP79] Alfred van der Poorten. A proof that Euler missed... Apéry's proof of the irrationality of $\zeta(3)$. *Math. Intelligencer*, 1:195 – 203, 1979.

[Wil90] Herbert S. Wilf. *generatingfunctionology.* Academic Press, Inc., Boston, 1990.

[Wil91] Herbert S. Wilf. Sums of closed form functions satisfy recurrence relations. Preprint, 1991.

[Wim84] Jet Wimp. *Computation and Recurrence Relations.* Pitman, Marshfield, MA, 1984.

[Wol88] Stephen Wolfram. *Mathematica$^{\text{TM}}$: A System for Doing Mathematics by Computer.* Addison–Wesley, Redwood City, CA, 1988.

[WZ85] Jet Wimp and Doron Zeilberger. Resurrecting the asymptotics of linear recurrences. *J. Math. Anal. Appl.*, 111:162 – 176, 1985.

[WZ90] Herbert S. Wilf and Doron Zeilberger. Rational functions certify combinatorial identities. *J. Amer. Math. Soc.*, 3:147 – 158, 1990.

[Zeia] Doron Zeilberger. A holonomic systems approach to special functions identities. To appear.

[Zeib] Doron Zeilberger. The method of creative telescoping. To appear.