J. Symbolic Computation (1995) **20**, 653–671

Computer Algebra Libraries for Combinatorial Structures

PHILIPPE FLAJOLET AND BRUNO SALVY

Algorithms Project, INRIA, 78153 Le Chesnay, France

(Received 29 December 1994)

This paper introduces the framework of decomposable combinatorial structures and their traversal algorithms. A combinatorial type is decomposable if it admits a specification in terms of unions, products, sequences, sets, and cycles, either in the labelled or in the unlabelled context. Many properties of decomposable structures are decidable. Generating function equations, counting sequences, and random generation algorithms can be compiled from specifications. Asymptotic properties can be determined automatically for a reasonably large subclass. Maple libraries that implement such decision procedures are briefly surveyed (`LUO`, `combstruct`, `equivalent`). In addition, libraries for manipulating holonomic sequences and functions are presented (`gfun`, `Mgfun`).

Introduction

This paper is a short account of research† concerning enumerative combinatorics and computer algebra with applications to the average case analysis of algorithms. It is a summary of invited lectures given by P. Flajolet and B. Salvy at the Workshop on Combinatorics and Computer Algebra held at Cornell University in September 1993. We describe here the major principles and functionalities of a collection of libraries aimed at the manipulation of combinatorial generating functions. All programs are being developed within the computer algebra system Maple. At the moment, they are available on the server `ftp.inria.fr` (under the directory `lang/maple/INRIA`).

The present paper offers a concise perspective on an approach developed, in detail, in previous works (Flajolet, Salvy and Zimmermann, 1991; Flajolet, Zimmermann and Van Cutsem, 1994; Zimmermann, 1994) as well as on the logic underlying recent developments. Our presentation is principally based on the following work.

— A general framework for the automatic analysis of so-called "decomposable" combinatorial structures and its extension to traversal procedures as described in Flajolet, Salvy and Zimmermann (1991). Two major components are needed: one deals with the algebraic manipulation of combinatorial generating functions (Zimmermann,

† The main participants in the research programme described are B. Salvy and P. Zimmermann with additional developments due to X. Gourdon, F. Chyzak, E. Murray, and with supporting theoretical research contributed by P. Flajolet.

0747–7171/95/050653 + 19 $12.00/0

1991), the other one with the asymptotic analysis of coefficients of such generating functions (Salvy, 1991a). This theory has given rise to the integrated system LUO (Lambda-Upsilon-Omega, $\Lambda\Upsilon\Omega$) for the analysis of traversal procedures on decomposable structures.

— A matching collection of random generation algorithms for selecting amongst a decomposable combinatorial class uniformly at random an element of given size. The principles are described in Flajolet, Zimmermann and Van Cutsem (1994), with a first implementation, called `Gaia`, presented in Zimmermann (1994). The current version of `Gaia`, called `combstruct`, will be the basis of further developments and should be merged with LUO.

— A library called `equivalent` for extracting the asymptotic form of coefficients, of generating functions that cover a large subset of generating functions arising from decomposable classes, and forms the basis of the analytic engine of LUO.

— A library called `gfun` for the algebraic manipulation of generating functions, especially of the so-called "holonomic" type.

Roughly speaking, the development related to LUO has enabled us to validate a general theory and a global system architecture. In a second phase, attention focuses on the design of a complete set of algorithms and libraries for the basic functions whose need was revealed by the LUO design. A later phase will be dedicated to the integration of these algorithms and libraries into a successor to LUO.

1. Decomposable Structures

1.1. DECOMPOSABLE STRUCTURES

Research conducted by several combinatorialists like Chomsky and Schützenberger (1963), Foata and Schützenberger (1970), Rota (1975), Foata (1974), Joyal (1981) and a few others in the course of the last three decades has led to a considerable change of perspective regarding combinatorial enumerations. While combinatorial enumerations were previously conceived largely as a collection of techniques for solving recurrences, several frameworks developed with the aim of directly relating combinatorial structures and their associated generating functions. See the accounts given in Bergeron, Labelle and Leroux (1994), Goulden and Jackson (1983), Stanley (1978, 1986).

A remarkable fact is that a collection of basic set-theoretic constructions acting on combinatorial classes like

(disjoint) union, product, formation of sequences, of sets, of cycles (1.1)

have translations into operators on associated generating functions in the rough form of

sum, product, quasi-inverse, exponential, logarithm. (1.2)

Technically, two "universes" exist: in the labelled universe, all "atoms" (nodes, letters, etc) composing a structure are distinguishable, being, for instance, labelled by distinct integers from $\{1, \ldots, n\}$ for a structure of size n; in the unlabelled universe, nodes are indistinguishable.

From there, one introduces a specification language for combinatorial structures. A *specification* is a formal grammar of generalized context-free type involving the combinatorial constructions of (1.1). A specification thus resembles a type description in a classical programming language. A combinatorial class is said to be *decomposable* if it admits such a specification. The basic framework is detailed in Flajolet, Salvy and Zimmermann (1991).

Simple examples of decomposable classes in the unlabelled universe are (atoms are indicated on the right):

Binary words:	`Word = Sequence(Union(a,b));`	(letters `a`, `b`)
Binary plane trees:	`B = Union(N,Prod(B,B));`	(external nodes `N`)
General non-plane trees:	`G = Prod(Z,Set(G));`	(nodes `Z`)

Here, `Set` means a set whose elements may be replicated, i.e., a *multiset*. (Another construction called `Powerset` takes care of the case where duplicates are forbidden.)

Examples in the labelled universe are:

Permutations:	`Per = Set(Cycle(Z));`	(`Z` is a labelled atom)
Set partitions:	`SP = Set(Set(Z,card>=1));`	(`Z` is a labelled atom).

Like in a context-free grammar, auxiliary non-terminals may be used. For instance, a functional graph (`FG`) is defined as a labelled digraph in which nodes all have outdegree 1; any such graph decomposes into connected components, each in the form of a cycle of Cayley trees (labelled non-plane trees). Hence, the specification

Functional Graphs : `FG=Set(K);K=Cycle(T);T=Prod(Z,Set(T));`. (1.3)

1.2. GENERATING FUNCTIONS

Let C be a combinatorial class of some kind, with C_n the number of objects in C having size n. Then, the ordinary generating function (OGF) and the exponential generating function (EGF) of C are defined by

$$C(z) = \sum_n C_n z^n \quad \text{and} \quad \hat{C}(z) = \sum_n C_n \frac{z^n}{n!}.$$

A general convention proves particularly useful: we constantly represent a class like C, its enumeration sequence $\{C_n\}$, and the corresponding generating functions $C(z)$, $\hat{C}(z)$ by the same group of letters.

PRINCIPLE 1.1. *Generating function equations can be compiled automatically for any decomposable class.*

The translations affect OGFs in the unlabelled case, and EGFs in the labelled case. They are summarized in Table 1.

Thus, the generating function of any combinatorial class that is decomposable is implicitly defined by a system of equations derived from Table 1. In many practical situations, these generating functions can be solved explicitly. Within `LUO`, a dedicated solver (Flajolet, Salvy and Zimmermann, 1991; Zimmermann, 1991) was developed, based on the capabilities of Maple's `solve` function. For instance, binary trees lead to an algebraic equation for the OGF of a simple form

$$B(z) = z + B^2(z) \quad \Rightarrow \quad B(z) = \frac{1-\sqrt{1-4z}}{2},$$

Table 1. Translation tables for basic combinatorial constructions in the unlabelled and labelled universes.

Construction	Translation (OGF)
$\mathcal{F} = \mathcal{G} \uplus \mathcal{H}$	$F(z) = G(z) + H(z)$
$\mathcal{F} = \mathcal{G} \times \mathcal{H}$	$F(z) = G(z) \cdot H(z)$
$\mathcal{F} = \mathtt{Sequence}(\mathcal{G})$	$F(z) = [1 - G(z)]^{-1}$
$\mathcal{F} = \mathtt{Set}(\mathcal{G})$	$F(z) = \exp[G(z) + G(z^2)/2 + G(z^3)/3 + \cdots]$
$\mathcal{F} = \mathtt{Powerset}(\mathcal{G})$	$F(z) = \exp[G(z) - G(z^2)/2 + G(z^3)/3 - \cdots]$
$\mathcal{F} = \mathtt{Cycle}(\mathcal{G})$	$F(z) = \log(1 - G(z))^{-1} + \cdots$
Construction	**Translation (EGF)**
$\mathcal{F} = \mathcal{G} \uplus \mathcal{H}$	$\hat{F}(z) = \hat{G}(z) + \hat{H}(z)$
$\mathcal{F} = \mathcal{G} * \mathcal{H}$	$\hat{F}(z) = \hat{G}(z) \cdot \hat{H}(z)$
$\mathcal{F} = \mathtt{Sequence}(\mathcal{G})$	$\hat{F}(z) = [1 - \hat{G}(z)]^{-1}$
$\mathcal{F} = \mathtt{Set}(\mathcal{G})$	$\hat{F}(z) = \exp(\hat{G}(z))$
$\mathcal{F} = \mathtt{Cycle}(\mathcal{G})$	$\hat{F}(z) = \log(1 - \hat{G}(z))^{-1}$

and the specification of functional graphs (1.3) yields for EGFs

$$\widehat{FG}(z) = \exp(\hat{K}(z)), \quad \hat{K}(z) = \log \frac{1}{1 - \hat{T}(z)}, \quad \hat{T}(z) = ze^{\hat{T}(z)}. \tag{1.4}$$

This last example involves the implicitly defined function $\hat{T}(z)$ which in Maple reduces to $-W(-z)$, with W the inverse function of ye^y, so that

$$\widehat{FG}(z) = \frac{1}{1 + W(-z)}.$$

Incidentally, this demonstrates that the capabilities of the solver are tightly coupled with the design of the underlying computer algebra system.

The automatic computation of generating function equations is the basis of all further developments. In particular, it provides the basis for the computation of counting sequences.

1.3. COUNTING SEQUENCES AND RANDOM GENERATION

Once generating function equations have been determined, coefficients can sometimes be obtained by direct Taylor series expansions. However, such an approach presents some difficulties since series solutions for systems of equations that are general enough are not available in computer algebra systems.

A general result (Flajolet, Zimmermann and Van Cutsem, 1994; Zimmermann, 1991) is the following.

PRINCIPLE 1.2. *The first n elements of the counting sequence of any decomposable class can be determined in $\mathcal{O}(n^2)$ arithmetic operations.*

The method consists in reducing specifications to a binary normal form (that generalizes Chomsky's normal form for context-free grammars) at the expense of introducing the differential operator $\Theta = z\frac{d}{dz}$. For instance, a specification for the class $\mathcal{G}$ of general non-plane trees is only one line in `combstruct`

```
g:=[G,G=Prod(Z,Set(G)),unlabelled];
```

which is to be interpreted as follows: the "axiom" is `G`, the grammar itself consists of one equation, and the specification is to be taken in the unlabelled universe; `Z` is implicitly defined as "atomic". Correspondingly, the generating function obeys

$$G(z) = z \exp\left[G(z) + \frac{1}{2}G(z^2) + \frac{1}{3}G(z^3) + \dots\right].$$

Applying Θ on both sides yields

$$\Theta G(z) = G(z)[1 + (\Theta G)(z) + (\Theta G)(z^2) + (\Theta G)(z^3) + \cdots],$$

itself equivalent to a recurrence on coefficients $G_n = [z^n]G(z)$, namely

$$nG_n = G_n + \sum_{k=1}^{n}(kG_k)G_{n-k} + \sum_{k=1}^{\lfloor n/2 \rfloor}(kG_k)G_{n-2k} + \cdots.$$

Given the specification of any decomposable class, the `combstruct` package automatically produces procedures to compute the counting sequences. These counting procedures can then be executed at will. Here, the corresponding count is simply obtained by `count(g,size=100);`

51 384 328 351 659 326 880 337 136 395 054 298 255 277 970

It takes about 3 seconds on our reference machine (100 MIPS) to determine this number G_{100}—this is Sequence #454 of Sloane (1973)—and 4 seconds in the corresponding labelled case of T_{100}—known otherwise to equal 100^{99}.

Random generation is another major function provided by `combstruct`. This makes it possible to write simulation routines to study various parameters of combinatorial structures. The `draw` command will uniformly generate at random an object amongst all elements of size n, for example,

```
draw(g,size=5);
        Prod(Z,Set(Prod(Z,Set(Prod(Z,Eps),Prod(Z,Eps),Prod(Z,Eps)))))
```

and the returned object is a Maple structure fully consistent with the specification (here `Eps` represents the empty set).

PRINCIPLE 1.3. *Any decomposable structure has a random generation algorithm of worst-case arithmetic complexity $\mathcal{O}(n \log n)$ that is effectively computable.*

The random generation procedures are compiled from the specification and they make full use of the counting tables. The algorithmic design combines the reduction of specifications to binary form, a top-down recursive algorithm, and a general so-called boustrophedonic search. The principles, in the labelled case at least, are given in Flajolet, Zimmermann and Van Cutsem (1994) and an earlier implementation of `combstruct`, called `Gaia`, is described in Zimmermann (1994). The algorithms only require $\mathcal{O}(n \log n)$ arithmetic operations, and it takes of the order of 0.5 seconds to generate a random tree in $\mathcal{G}$ of size 100. In this particular case, the system automatically compiles an optimized version of the Ranrut algorithm of Nijenhuis and Wilf (1978).

2. Asymptotic Analysis

Coefficients of generating functions associated to decomposable structures do not generally have explicit expressions. However, for large classes of combinatorial structures, it is possible to deduce asymptotic expansions of the combinatorial sequences *directly* from the equations they satisfy.

The mathematical principles are based on Cauchy's coefficient formula which relates the nth Taylor coefficient of a series to the function itself:

$$[z^n]f(z) = \frac{1}{2i\pi} \oint \frac{f(z)}{z^{n+1}}\, dz. \tag{2.1}$$

This makes it possible to relate the asymptotic behaviour of the coefficients of $f(z)$ to the location of the dominant singularities of f (those of smallest modulus) and to the singular behaviour of f at those points.

PRINCIPLE 2.1. *For a large subclass of generating functions associated to decomposable structures, the asymptotic form of coefficients is computable automatically.*

2.1. RATIONAL FUNCTIONS

In the case of rational functions, the approach consists in computing a partial fraction decomposition and then extracting the coefficients of the terms corresponding to the poles of smallest modulus.

Bronstein and Salvy (1993) showed how to compute a partial fraction decomposition over the algebraic closure of the ground field by simple gcd computations, without resorting to polynomial factorization. This leads to a fast decomposition as a sum of terms of the form

$$\frac{c_\alpha}{(z-\alpha)^{k_\alpha}},$$

where α is defined by $Q(\alpha) = 0$ for some polynomial Q. Extracting the coefficients of the terms corresponding to the poles of minimal modulus and maximal order gives the first-order asymptotic behaviour of the coefficients. To determine more terms of the expansion, it is necessary to determine how many poles lie on successive circles of increasing modulus. This can be done algebraically using resultants and Sturm sequences, albeit with an exponential complexity. A better approach is based on *guaranteed numerical approximations* and leads to a polynomial time algorithm (Gourdon and Salvy, 1993). This uses a polynomial root-finding algorithm from Schönhage (1982, 1987) which has been implemented by Gourdon (1993).

EXAMPLE 2.1. *The following combinatorial problem was considered by Conway (1987). Starting with* 1, *write down a sequence of words by counting the number of contiguous identical digits in the previous word. Thus the second word is* 11 *because there is one* 1 *in* "1". *Then the third word is* 21, *and so on. The first few words are:* 1, 11, 21, 1211, 111221, 312211, 13112221, ... *It turns out that the sequence, of lengths, of these words:* 1, 2, 2, 4, 6, 6, 6, 8, ... *has a rational generating function whose denominator has* *degree 72. From the table in Conway (1987; pp. 177–178), it is possible to compute this fraction by solving a linear system. One of the nice theorems of Conway's states that the denominator is actually independent of the starting string, provided it is different from "22". Thus in the leading term of the asymptotic expansion, only the constant factor depends on the initial string.*

Despite the large degree of this denominator, the asymptotic expansion is not too difficult to find. The partial fraction decomposition is

$$R(z) + \sum_{Q(\alpha)=0} \frac{F(\alpha)}{z-\alpha},$$

where R *is a polynomial induced by the first terms. This means that all the singularities are simple poles. Here,* F *is a polynomial of degree* 71 *with* 250*-digit rational coefficients. If one is only interested in the first-order estimate, it then remains to determine the number of roots of smallest modulus. As expected since the coefficients of the generating function are positive, one of these roots is a positive real number. Using Gourdon's program, we get the two smallest moduli, approximately* 0.767 *and* 0.861*, with error bounds of the order* 10^{-40}*, which shows that there is a unique root of smallest modulus (which is necessarily real). Thus,* $[z^n]f(z) \sim F(\rho_1)\rho_1^{-n-1}$*, with* $\rho_1 \simeq 0.767\,119$ *and* $F(\rho_1) \simeq 1.566$*. All the* 72 *moduli belong to the interval* $(0.767, 1.151)$*, showing the need for a carefully designed polynomial solver.*

2.2. SINGULARITY ANALYSIS

The computation of the asymptotic behaviour of coefficients of rational functions obeys a pattern which is actually much more general.

PRINCIPLE 2.2. *For generating functions of moderate growth at dominant singularities, there is a systematic correspondence between singular expansion of the function and asymptotic expansion of coefficients.*

The method can be outlined as follows (see, Flajolet and Odlyzko, 1990; for a rigorous description):

1. locate the dominant singularities (the ones of smallest modulus);
2. check that the function is analytically continuable in a small region outside of its circle of convergence;
3. compute the local expansion of the function in the neighbourhood of its dominant singularities;
4. translate these singular expansions into corresponding expansions of the coefficients.

In practice, step 2 above is always ensured by the fact that singularities of large classes of functions are isolated. The property holds *a priori* for most generating functions associated to decomposable structures—for instance, all functions presented by their closed-form in terms of exp, log and rational functions. The last step is the basis of the method. It asserts that under mild conditions, the nth coefficient of

$$f(z) = f_1(z) + f_2(z) + \cdots + f_k(z) + O(g(z)), \qquad z \to \rho,$$

Table 2. The correspondence between asymptotic growth of the coefficients and the singular growth of the function.

Behaviour of the function		Growth of its coefficients
$c(1-z/\rho)^\alpha$	$\alpha \notin \mathbb{N}$	$c\rho^{-n}n^{-\alpha-1}/\Gamma(-\alpha)$
$c(1-z/\rho)^\alpha \log^\beta[1/(1-z/\rho)]$	$\alpha \notin \mathbb{N}$	$c\rho^{-n}n^{-\alpha-1}\log^\beta n/\Gamma(-\alpha)$
$c(1-z/\rho)^k \log^\beta[1/(1-z/\rho)]$	$k \in \mathbb{N}$	$c(-1)^k\beta k!\rho^{-n}n^{-k-1}\log^{\beta-1} n$

where ρ is the dominant singularity of f, behaves asymptotically like

$$[z^n]f_1(z) + [z^n]f_2(z) + \cdots + [z^n]f_k(z) + \mathcal{O}([z^n]g(z)), \qquad n \to \infty.$$

Growth orders for standard functions are represented in Table 2. Actually, Flajolet and Odlyzko (1990) give full expansions for the whole class of *algebraic–logarithmic* singularities.

This method has been implemented in Maple by Salvy (1991a). The program, called `equivalent`, starts from an explicit (exp–log) generating function. It looks for the dominant singularities by a simple iterative procedure which reduces singularity finding to root finding. Then a local asymptotic expansion is computed. Translating these expansions to expansions of the coefficients is then an easy matter.

EXAMPLE 2.2. *The generalized bracketing problem of Schröder (Comtet, 1974; p. 56). The problem is to determine the number of bracketings of n symbols such that pairs of brackets always enclose at least two terms. For instance, here are the 11 bracketings of 4 symbols:*

$$(ab)(cd), \quad ((ab)c)d, \quad (a(bc))d, \quad a((bc)d), \quad a(b(cd)),$$
$$abcd, \quad (abc)d, \quad a(bcd), \quad (ab)cd, \quad a(bc)d, \quad ab(cd).$$

The corresponding specification is

```
Bracketing = Union(Symbol,Sequence(Bracketing,card>=2));.
```

By the methods described in Section 1, the generating function $Y(z)$ of these bracketings satisfies

$$Y = z + \frac{Y^2}{1-Y},$$

from which the solver deduces that the relevant solution is

$$Y(z) = \frac{1}{4}(1 + z - \sqrt{1 - 6z + z^2}).$$

From this explicit form, `equivalent` *will deduce that the dominant singularity is at* $3 - 2\sqrt{2}$ *and a local analysis at this point leads to the result:*

```
equivalent((1+z-sqrt(1-6*z+z^2))/4,z,n);
```

$$\frac{\sqrt{12\sqrt{2}-16}(3-2\sqrt{2})^{-n}}{8\sqrt{\pi}n^{3/2}} + \mathcal{O}\left(\frac{1}{n^{5/2}(3-2\sqrt{2})^n}\right).$$

EXAMPLE 2.3. *Stanley's children rounds (Stanley, 1978). Children group in circles with one child at the centre of each circle. The problem is to compute the number of ways this* *can be done with n children. Using the language of Section 1, the specification is*

```
Rounds = Set(Product(Child,Cycle(Child)));
```

leading to the generating function

$$\hat{R}(z) = \exp\left(z \log\left(\frac{1}{1-z}\right)\right).$$

The dominant singularity is at 1 where R has a logarithmic behaviour. Hence the first four terms of the expansion:
`equivalent(exp(z*log(1/(1-z))),z,n,4);`

$$1 - \frac{1}{n} - \frac{\ln n}{n^2} + \frac{1-\gamma}{n^2} + \mathcal{O}\left(\frac{\ln(n)^2}{n^3}\right).$$

Here γ is Euler's constant.

The difficult part in this computation comes from the frequent need for algebraic–logarithmic asymptotic scales, which are not provided by computer algebra systems. A program called `gdev` was developed for that purpose (Salvy, 1991a, b). This program does not completely solve the problem of finding local expansions for the class of exp–log functions. A general algorithm for doing so was given by Shackell (1990), but no implementation of this algorithm is yet available. An approach similar to Shackell's has been developed by Gonnet and Gruntz (1992). This may soon provide Maple with the best asymptotic expander of all existing computer algebra systems (Gruntz, 1995).

2.3. SADDLE-POINT METHOD

Not all combinatorial generating functions have an algebraic–logarithmic singularity. In many cases the function is entire or has an essential singularity at a finite distance. A large class of such functions is handled by the saddle-point method.

PRINCIPLE 2.3. *A decidable subclass of functions with fast growing singular behaviour have coefficients that are approximated by saddle-point integrals.*

The idea is to choose a circle of integration in Cauchy's formula (2.1) such that the integral is concentrated in the neighbourhood of a special point (the *saddle-point*), where the integral can be approximated by a Gaussian integral. The point is determined by

$$R_n \frac{f'(R_n)}{f(R_n)} = n+1, \tag{2.2}$$

and the asymptotic approximation furnished by the method is

$$[z^n]f(z) \sim \frac{f(R_n)}{R_n^n\sqrt{2\pi h''(R_n)}}, \tag{2.3}$$

where $h = \log(f) - (n+1)\log z$. Sufficient conditions for this method to be valid were given by Hayman (1956) and they can be checked by a computer. In some cases a full expansion is available (Wyman, 1959) and effective criteria for deciding this are available (Harris and Schoenfeld, 1968; Odlyzko and Richmond, 1985). These were also implemented in `equivalent`.

EXAMPLE 2.4. *The number of increasing subsequences in permutations was considered by Lifschitz and Pittel (1981). For instance, the permutation* 524 361 *has* 15 *increasing subsequences, namely the empty sequence, each single element and*

$$56, \quad 24, \quad 246, \quad 23, \quad 236, \quad 26, \quad 46, \quad 36.$$

Determining the mean number of increasing subsequences in a random permutation is equivalent to counting "marked permutations" which are permutations with a distinguished increasing subsequence. Such a marked permutation decomposes as a regular permutation followed by a sequence of fragments with the additional condition that initial elements of the fragments run in increasing order, for instance

$$\begin{aligned} \tau = 36\underline{2}\,41\underline{5}\,72\underline{8} &\equiv (36)(\underline{2}41)(\underline{5}72)(\underline{8}) \\ &\equiv (36)\{(\underline{5}72), (\underline{2}41), (\underline{8})\}. \end{aligned}$$

The specification of marked permutations is therefore

```
Perm = Sequence(Z); Fragment = Sequence(Z, card>=1);
MarkedPerm = Product(Perm,Set(Fragment));
```

From this the generating function is found to be

$$\frac{1}{1-z}\exp\left[\frac{z}{1-z}\right].$$

The asymptotic behaviour of its coefficients is then determined automatically:

```
equivalent(1/(1-z)*exp(z/(1-z)),z,n);
```

$$\frac{e^{2\sqrt{n}}}{2\sqrt{e}\sqrt{\pi}n^{1/4}} + O\left(\frac{e^{2\sqrt{n}}}{n^{3/4}}\right).$$

In an example like this, Equation (2.2) admits a simple closed-form solution. In general though, no such closed-form exists and it is necessary to find an asymptotic expansion of the saddle-point location in terms of n. A general procedure for doing so when f is any exp–log function was given by Salvy and Shackell (1992) and a fast algorithm in a special but frequent case was given in Salvy (1994). Knowing the asymptotic expansion of the location of the saddle-point is not always sufficient to complete the asymptotic expansion of the coefficients, since in some cases substituting the expansion of R_n in (2.3) does not lead to a genuine asymptotic expansion of Poincaré type.

EXAMPLE 2.5. *Bell numbers are the number of partitions of a set into non-empty sets. From Section 1, it follows that their exponential generating function is* $\exp(e^z - 1)$. *Here the saddle-point expansion must not be substituted into the expansion, and we get automatically (compare with De Bruijn 1981)*

```
equivalent(exp(exp(z)-1),z,n);
```

The saddle-point is $W(n+1)$

Saddle-point's expansion:

$$\zeta = \ln n - \ln\ln n + \frac{\ln\ln n}{\ln n} + \mathcal{O}\left(\frac{\ln^2\ln n}{\ln^2 n}\right)$$

$$\frac{\sqrt{2}\zeta^{-n-1}e^{-\frac{\zeta}{2}e^{e^\zeta}}}{2e\sqrt{\pi}\zeta^n} + \mathcal{O}\left(\frac{e^{e^\zeta}}{(\zeta^n)^2\zeta^2\sqrt{e^\zeta}}\right).$$

3. Procedures

Like combinatorial enumeration, the average-case analysis of algorithms is often treated by recurrences. Flajolet and Steyaert first recognized that several general algorithmic schemes admit a direct translation into generating functions; see Flajolet and Steyaert (1987); Steyaert (1984) for the particular case of tree algorithms. This approach was later extended to a coherent collection of traversal mechanisms that applies to all decomposable combinatorial structures (Flajolet, Salvy and Zimmermann, 1991; Zimmermann, 1991). The LUO system implements these ideas.

A procedure is specifiable in LUO if it involves only data types that are decomposable structures in the sense of Section 1 and simple traversal procedures of a purely functional form. The allowed procedures can test cases (for types defined by unions), descend into components (of products), and iterate on components (of sequences, sets, or cycles). A cost measure is specified in terms of the number of times a designated procedure is executed.

Internally, the LUO system comprises three parts: an "algebraic analyzer" systematically determines generating function equations from specifications of types and procedures (according to principles, extending those of Section 1), a "solver" (briefly mentioned in Section 1) is dedicated to finding closed-form solutions of generating function equations whenever available, and finally an "asymptotic analyzer" uses as a basic engine the `equivalent` program of Section 2. In LUO, the algebraic analyzer has been implemented as a special set of CAML procedures since the specifications are of a Pascal-like format. *In the future, the descendant of* LUO *should be entirely Maple-based with a syntax extending that of* `combstruct`.

Given a data type $\mathcal{C}$ which is a decomposable class, a LUO-admissible procedure P of signature $P(c : \mathcal{C})$ and a fixed cost measure, the cost of executing P on $c \in \mathcal{C}$ is well-defined and is denoted by $\tau P[c]$. The total costs are then

$$\tau P_n = \sum_{c\in\mathcal{C},\,|c|=n} \tau P[c],$$

and the corresponding generating function is called the *complexity descriptor* of P. Naturally, EGFs or OGFs are taken according to whether the universe is labelled or not.

PRINCIPLE 3.1. *Complexity descriptors of traversal procedures over decomposable types are automatically computable. Their coefficients are amenable to singularity analysis or saddle-point methods.*

A noteworthy fact is that complexity descriptors in the labelled case lie in the same class as the counting generating functions of the basic data types, and in the unlabelled case, in a class that is only marginally larger. In particular, the exact values of $\{\tau P_n\}$ can be determined in $\mathcal{O}(n^2)$ arithmetic operations and the asymptotic values can be obtained by the same methods as discussed in Section 2.

```
type expression = zero | one | x
                  | product(plus,expression,expression)
                  | product(times,expression,expression)
                  | product(expo,expression);
plus,times,expo,zero,one,x = atom(1);

function diff(e:expression):expression;
case e of
  plus(e1,e2)     : plus(diff(e1),diff(e2));
  times(e1,e2)    : plus(times(diff(e1),copy(e2)),
                       times(copy(e1),diff(e2)));
  expo(e1)        : times(diff(e1),copy(e));
  zero            : zero;
  one             : zero;
  x               : one
end;

function copy (e:expression):expression;
case e of
  plus(e1,e2)    : plus(copy(e1),copy(e2));
  times(e1,e2)   : times(copy(e1),copy(e2));
  expo(e1)       : expo(copy(e1));
  zero           : zero;
  one            : one;
  x              : x
end;

measure plus,times,expo,zero,one,x:1;
```

Figure 1. Symbolic differentiation, the LUO specification.

EXAMPLE 3.1. *Symbolic differentiation. Figure 1 displays the* LUO *specification of a symbolic differentiation algorithm* `diff` *that operates on expression (trees) built of symbols* 0, 1, x, $+$, $\times$, $\exp$; *the cost here is measured by the number of nodes of the differentiated expression tree (see Flajolet, Salvy and Zimmermann, 1989; for details). Counting generating functions and cost descriptors are determined automatically and the solver finds that they are all of the form*

$$R(z, \sqrt{1-2z-23z^2}) \qquad \text{with } R(z,y) \text{ a rational function in } \mathbb{C}(z,y).$$

From here, the asymptotic analyzer `equivalent` *obtains the expected cost from its built-in singularity analysis mechanism,*

$$\overline{\tau\mathtt{diff}}_n = \frac{(126+\sqrt{6})\sqrt{\pi}}{(-1+2\sqrt{6})^{3/2}6^{3/4}23^{1/2}} n^{3/2} + \mathcal{O}(n) \simeq 0.80\,421 n^{3/2} + \mathcal{O}(n).$$

Thus the cost grows super-linearly but subquadratically on average. A variant algorithm based on subexpression sharing is also within the capabilities of the system which finds for the a priori linear complexity a precise form that evaluates to

$$1.41523957n + \mathcal{O}(n^{1/2}).$$

The "Cookbook" (Flajolet, Salvy and Zimmermann, 1989) provides about twenty such analysis reports, in such diverse areas as addition chains, concurrent access problems, planar bipartite graphs, differentiation algorithms, tree rewriting, random trains, random functional graphs, etc.

The method specializes to parameters of combinatorial structures defined by the number of occurrences of a given construction. In such a case, one can always design a traversal procedure whose cost will record the number of occurrences of the construction in question. In addition, it becomes possible to automatically derive bivariate generating functions giving the distribution of the parameter in question, by extending the approach described in Section 1. This makes it possible, at least in principle, to approach such problems as automatic computations of variance and (in some cases) limit distributions, see Soria-Cousineau (1990). This idea should be explored further in future works.

4. Combinatorial Sequences and Generating Functions

Many combinatorial sequences are amenable to the asymptotic analysis of Section 2, provided a "nice" generating function can be found. In particular, it is our aim in the long term to automate the asymptotic analysis of Zeilberger's holonomic sequences. Meanwhile, a set of tools dealing with holonomic sequences and functions have been developed.

4.1. GUESSING A GENERATING FUNCTION

Sequences occurring in practice tend to have simple generating functions. Padé approximants can be used to check whether various kinds of generating functions derived from a particular sequence are rational. In this way Plouffe (1992) estimates that about 13% of the table (Sloane, 1973) have a rational generating function; Bergeron and Plouffe (1992) found that about 23% of the sequences in Sloane's table had a generating function which could be guessed. Similar ideas are incorporated into **gfun** which looks for holonomic generating functions. It was then found that about 18% of Sloane's sequences are holonomic. The first version of **gfun** (Salvy and Zimmermann, 1994) used a method of indeterminate coefficients; the algorithm has been changed recently to use a technique of Hermit Padé approximants due to Harm Derksen (see also Beckermann and Labahn, 1992), which leads to an appreciable speed-up.

EXAMPLE 4.1. *Numerators of convergents to e. It is known since Euler that the continued fraction expansion of e has the regular quotient sequence* 2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, ... *The fractions obtained by truncating this expansion define the convergents to e, and restricting attention to elements of index* $3k+1$ *yields*

$$3,\ \frac{19}{7},\ \frac{193}{71},\ \frac{2721}{1001},\ \frac{49171}{18089},\ \frac{1084483}{398959},\ \frac{28245729}{10391023},\ \frac{848456353}{312129649},\ \frac{28875761731}{10622799089},\ \ldots.$$

We input the sequence of numerators of this to the procedure `listtodiffeq` *of the* `gfun` *package:*

```
l:=[3,19,193,2721,49171,1084483,28245729,848456353,28875761731]:
listtodiffeq(l,y(z));
```

$$\left[\left\{y(0)=3, D(y)(0)=19, \frac{y(z)}{4}+\frac{5}{2}\frac{d}{dz}y(z)+(z-1/4)\frac{d^2}{dz^2}y(z)\right\}, \mathrm{egf}\right].$$

The "egf" term means that this is the equation satisfied by the exponential generating function (this is user-settable). Then the Maple differential equation solver finds a solu-

tion to this equation, which after simplification reduces to

$$y(z) = \frac{\sqrt{e^{1-\sqrt{1-4z}}}}{1-4z}\left(1+\frac{2}{\sqrt{1-4z}}\right). \tag{4.1}$$

Of course, this does not constitute a proof, but consistency with the next values strongly militates in favour of its validity. The result can then be proved formally by the methods of the next section.

This part of gfun has been incorporated into a mail server created by N. Sloane (`superseeker@research.att.com`).

4.2. HOLONOMY IN ONE VARIABLE

Order constraints on the labels of decomposable structures lead to generating functions obeying differential equations. A function is called holonomic when it satisfies a *linear* differential equation with polynomial coefficients. Similarly, holonomic sequences are sequences that satisfy a linear recurrence with polynomial coefficients.

PRINCIPLE 4.1. (ZEILBERGER, LIPSCHITZ, STANLEY) *Closure properties of holonomic functions and sequences are effectively computable. Consequently, identities between holonomic functions and sequences are decidable.*

Another part of **gfun** implements the numerous closure properties of holonomic functions and sequences (Lipschitz, 1989; Stanley, 1980; Zeilberger, 1990). In particular, it is known that:

— a sequence is holonomic (P-recursive) if and only if its generating function is holonomic (D-finite);
— the sum and product of two holonomic functions or sequences are holonomic;
— algebraic functions are holonomic;
— the composition of a holonomic function with an algebraic function is holonomic.

EXAMPLE 4.2. *We compute the linear recurrence satisfied by*

$$f_n = \sum_{k=0}^{n}\binom{n}{k}\frac{(-1)^k}{k^2+1}. \tag{4.2}$$

The existence of such a recurrence is ensured by the closure properties mentioned above, the summation being provided by Euler's transform

$$[z^n]\frac{1}{1-z}g\left(\frac{z}{z-1}\right) = \sum_{k=0}^{n}\binom{n}{k}(-1)^k g_k. \tag{4.3}$$

A simple heuristic approach consists in computing sufficiently many terms of the sequence and then letting the guessing mechanism of gfun *find the recurrence. Since the summand is hypergeometric in both variables, it is also possible to use Zeilberger's fast algorithm to get the recurrence (see e.g., Paule and Schorn, 1996). We detail here how the equation is rigorously constructed via the univariate closure properties. We start from the trivial order 0 recurrence satisfied by* $1/(k^2+1)$

```
rec:=(k^2+1)*u(k)=1:
```

from which we deduce the differential equation satisfied by its generating function:
`rectodiffeq(rec,u(k),y(z));`

$$\{(1-z)z^2y''(z)+(1-z)zy'(z)+(1-z)y(z)-1, y(0)=1, y'(0)=1/2\}.$$

Then we perform the algebraic substitution $z \mapsto z/(z-1)$:
`algebraicsubs(",y*(z-1)-z,y(z));`

$$z^2(z-1)^2y''(z)+z(-1+2z)(z-1)y'(z)+y(z)-1+z.$$

Now we have to multiply $y(z)$ by $1/(1-z)$. Although this can be done directly, we illustrate the more general mechanism of multiplication of holonomic functions:
`'diffeq*diffeq'(",(1-z)*y(z)-1,y(z));`

$$\{z^2(z-1)^2y''(z)+z(4z-1)(z-1)y'(z)+(2z^2-z+1)y(z)-1, y(0)=1, y'(0)=1/2\}. \quad (4.4)$$

We then have to just translate this into a recurrence for the coefficients:
`diffeqtorec(",y(z),f(n));`

$$\{(n^2+3n+2)f(n)-(7n+6+2n^2)f(n+1)+(5+4n+n^2)f(n+2), f(0)=1, f(1)=1/2\}.$$

Finding such equations serves various purposes.

1. IDENTITY PROVING

To prove a combinatorial identity $a=b$, the technique promoted by Zeilberger (1990), Wilf and Zeilberger (1992) consists in building up the equation satisfied by $a-b$ as exemplified above. The identity is then proved by checking sufficiently many initial conditions. Zeilberger published numerous examples of uses of this method, a detailed example based on `gfun` being given in Flajolet and Salvy (1993).

2. FAST COMPUTATION

A linear recurrence makes it possible to compute n terms of a sequence in $\mathcal{O}(n)$ arithmetic operations. In particular, the fastest known algorithm to compute the series expansion of an algebraic function consists in first computing the differential equation it satisfies (the procedure `algeqtodiffeq` in `gfun`), then using the recurrence on its Taylor coefficients (Chudnovsky and Chudnovsky, 1986).

3. FINDING CLOSED-FORMS

Several algorithms are known to find closed-form solutions of linear differential or difference equations. Abramov (1989) gave fast algorithms to find rational solutions of such equations. Algorithms for finding Liouvillian solutions of linear differential equations are now implemented in most computer algebra systems (see Singer, 1990 for a survey of the algorithms). Petkovšek (1992) gave an algorithm to find hypergeometric solutions of linear recurrences with polynomial coefficients. This in turn gives an algorithm to find generalized hypergeometric solutions of linear differential equations with polynomial coefficients (Petkovšek and Salvy, 1993). Our prototype implementations should soon make their way into Maple's library.

4. ASYMPTOTICS

The area of computer algebra needed to compute expansions of solutions of linear differential equations has undergone extensive research (Tournier, 1987; Duval, 1987; Thomann, 1990). Completely solving the problem requires a mixture of formal computation with algebraic numbers and numerical resummation of divergent series. There are partial implementations of this in most computer algebra systems. Using the local expansion of solutions at their singularities makes it possible to compute asymptotics of linear recurrences via singularity analysis.

PRINCIPLE 4.2. *The asymptotic form of a univariate holonomic sequence is computable.*

An alternative approach based on Birkhoff's work can be found in Wimp and Zeilberger (1985).

EXAMPLE 4.3. *We describe the main steps of the method on the sequence (4.2). Singularities of holonomic functions can be read off the corresponding differential equation: they are located at the roots of the leading coefficient. Here the generating functions satisfies (4.4), hence the dominant singularity is at 1. This is a regular singular point and a local analysis (e.g., using Maple's* `dsolve/series`*) shows that the local behaviour at 1 is of the form*

$$c_1(1-z)^{-1-i}\phi_1(z) + c_2(1-z)^{-1+i}\phi_2(z) + c_3\phi_3(z),$$

where $i = \sqrt{-1}$, the c_k are undetermined constants and the $\phi_k(z)$ are formal series in $1-z$, with $\phi_k(1) = 1$. Singularity analysis then shows that the asymptotic behaviour of the sequence f_n is

$$f_n \sim C\cos(\log n + \vartheta)$$

for constants C and ϑ that could be determined numerically from the series ϕ_k. In this particular example, Rice's method (see Flajolet and Sedgewick, 1994, and Knuth, 1973) also applies and yields the more precise result that $C = |\Gamma(i)|$ and an explicit form for ϑ.

4.3. MULTIVARIATE HOLONOMY

Holonomy extends to multivariate functions or sequences and to mixed cases like orthogonal polynomials that satisfy both a linear recurrence with respect to the index and a linear differential equation with respect to the argument (Zeilberger, 1990). One is then led to consider systems of linear operators and algebras of such operators. Under mild conditions, the left ideals of these algebras are finitely generated and a non-commutative variant of Buchberger's algorithm works in this context (Chyzak, 1994). Many of the closure properties of the univariate case still hold and some of them have been implemented by Chyzak in the `Mgfun` package. Identities satisfied by combinatorial sequences like Apéry's sequence (see Van der Poorten, 1979) can be obtained almost routinely by elimination using Zeilberger's (1991) creative telescoping (more details are given in Chyzak and Salvy, 1996).

Acknowledgement

This work was supported in part by the Esprit III Basic Research Action of the EEC under contract ALCOM II (#7141).

References

Abramov, S.A. (1989). Rational solutions of linear differential and difference equations with polynomial coefficients. *USSR Computational Mathematics and Mathematical Physics* **29**(11), 1611–1620. Translation of the Zhurnal vychislitel'noi matematiki i matematicheskoi fiziki.

Beckermann, B., Labahn, G. (1992). A uniform approach for Hermit Padé and simultaneous Padé approximants and their matrix-type generalizations. *Numerical Algorithms* **3**, 45–54.

Bergeron, F., Labelle, G., Leroux, P. (1994). *Théorie des espèces et combinatoire des structures arborescentes*. Number 19 in Publications du LACIM. Université du Québec à Montréal.

Bergeron, F., Plouffe, S. (1992). Computing the generating function of a series given its first terms. *J. Experimental Mathematics* **1**(4), 308–312.

Bronstein, M., Salvy, B. (1993, July). Full partial fraction decomposition of rational functions. In M. Bronstein (ed.), *ISSAC'93*, pp. 157–160. ACM Press.

Chomsky, N., Schützenberger, M.P. (1963). The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg (eds), *Computer Programing and Formal Languages*, pp. 118–161. Amsterdam: North-Holland.

Chudnovsky, D.V., Chudnovsky, G.V. (1986). On expansion of algebraic functions in power and Puiseux series, I. *J. Complexity* **2**(4), 271–294.

Chyzak, F. (1994, October). Holonomic systems and automatic proofs of identities. Research Report 2371, Institut National de Recherche en Informatique et en Automatique.

Chyzak, F., Salvy, B. (1996). Non-commutative elimination in Ore algebras proves multivariate identities. Research report 2799, INRIA.

Comtet, L. (1974). *Advanced Combinatorics*. Dordrecht: Reidel.

Conway, J.H. (1987). The weird and wonderful chemistry of audioactive decay. In T.M. Cover and B. Gopinath (eds), *Open Problems in Communication and Computation*, pp. 173–188. Springer-Verlag.

De Bruijn, N.G. (1981). *Asymptotic Methods in Analysis*. Dover. A reprint of the third (Amsterdam: North-Holland) edition, 1970 (first edition, 1958).

Duval, D. (1987). *Diverses questions relatives au calcul formel avec des nombres algébriques*. Doctorat d'État, Université scientifique, technologique et médicale de Grenoble.

Flajolet, P., Odlyzko, A.M. (1990). Singularity analysis of generating functions. *SIAM J. Discrete Mathematics* **3**(2), 216–240.

Flajolet, P., Salvy, B. (1993). A finite sum of products of binomial coefficients. *SIAM Review* **35**(4), 645–646. Solution to Problem 92-18 by C.C. Grosjean.

Flajolet, P., Salvy, B., Zimmermann, P. (1989, August). Lambda-Upsilon-Omega: The 1989 Cookbook. Research Report 1073, Institut National de Recherche en Informatique et en Automatique. 116 pages.

Flajolet, P., Salvy, B., Zimmermann, P. (1991, February). Automatic average-case analysis of algorithms. *Theoretical Computer Science, Series A* **79**(1), 37–109.

Flajolet, P., Sedgewick, R. (1994, March). Mellin transforms and asymptotics: finite differences and Rice's integrals. Research Report 2031, Institut National de Recherche en Informatique et en Automatique. 21 pages. To appear in *Theoretical Computer Science*.

Flajolet, P., Steyaert, J.-M. (1987). A complexity calculus for recursive tree algorithms. *Mathematical Systems Theory* **19**, 301–331.

Flajolet, P., Zimmermann, P., Van Cutsem, B. (1994). A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science* **132**(1–2), 1–35.

Foata, D. (1974). *La série génératrice exponentielle dans les problèmes d'énumération*. S.M.S. Montreal University Press.

Foata, D., Schützenberger, M.-P. (1970). *Théorie Géométrique des Polynômes Euleriens*, Volume 138 of *Lecture Notes in Mathematics*. Springer-Verlag.

Gonnet, G.H., Gruntz, D. (1992, November). Limit computation in computer algebra. Technical Report 187, ETH, Zürich.

Goulden, I.P., Jackson, D.M. (1983). *Combinatorial Enumeration*. New York: Wiley.

Gourdon, X. (1993, February). Algorithmique du théorème fondamental de l'algèbre. Research Report 1852, Institut National de Recherche en Informatique et en Automatique.

Gourdon, X., Salvy, B. (1993). Asymptotics of linear recurrences with rational coefficients. In A. Barlotti, M. Delest, and R. Pinzani (eds), *Formal Power Series and Algebraic Combinatorics*, pp. 253–266. Proceedings of FPACS'5, Florence (Italy).

Gruntz, D. (1995). *On Computing Limits in a Symbolic Manipulation System*. PhD. thesis, ETH, Zürich.

Harris, B., Schoenfeld, L. (1968). Asymptotic expansions for the coefficients of analytic functions. *Illinois J. Mathematics* **12**, 264–277.

Hayman, W.K. (1956). A generalization of Stirling's formula. *Journal für die reine und angewandte Mathematik* **196**, 67–95.

Joyal, A. (1981). Une théorie combinatoire des séries formelles. *Advances in Mathematics* **42**(1), 1–82.

Knuth, D.E. (1973). *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley.

Lifschitz, V., Pittel, B. (1981). The number of increasing subsequences of the random permutation. *J. Combinatorial Theory, Series A* **31**, 1–20.

Lipshitz, L. (1989). *D*-finite power series. *J. Algebra* **122**(2), 353–373.

Nijenhuis, A., Wilf, H.S. (1978). *Combinatorial Algorithms* (second edition). New York: Academic Press.

Odlyzko, A.M., Richmond, L.B. (1985). Asymptotic expansions for the coefficients of analytic generating functions. *Aequationes Mathematicae* **28**, 50–63.

Paule, P., Schorn, M. (1996). A Mathematica version of Zeilberger's algorithm for proving binomial coefficient identities. *J. Symbolic Computation* (this issue).

Petkovšek, M. (1992). Hypergeometric solutions of linear recurrences with polynomial coefficients. *J. Symbolic Computation* **14**, 243–264.

Petkovšek, M., Salvy, B. (1993, July). Finding all hypergeometric solutions of linear differential equations. In M. Bronstein (ed.), *ISSAC'93*, pp. 27–33. ACM Press.

Plouffe, S. (1992, September). Approximations de séries génératrices et quelques conjectures. Masters thesis, Université du Québec à Montréal. Also available as Research Report 92–61, Laboratoire Bordelais de Recherche en Informatique, Bordeaux, France.

Rota, G.-C. (1975). *Finite Operator Calculus*. New York: Academic Press.

Salvy, B. (1991a). *Asymptotique automatique et fonctions génératrices*. PhD. thesis, École polytechnique.

Salvy, B. (1991b, April). Examples of automatic asymptotic expansions. *SIGSAM Bulletin* **25**(2), 4–17.

Salvy, B. (1994). Fast computation of some asymptotic functional inverses. *J. Symbolic Computation* **17**, 227–236.

Salvy, B., Shackell, J. (1992). Asymptotic expansions of functional inverses. In P.S. Wang (ed.), *Symbolic and Algebraic Computation*, pp. 130–137. ACM Press. Proceedings of ISSAC'92, Berkeley, July 1992.

Salvy, B., Zimmermann, P. (1994). Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software* **20**(2), 163–177.

Schönhage, A. (1982). The fundamental theorem of algebra in terms of computational complexity. Technical report, Mathematisches Institut der Universität Tübingen. Preliminary report.

Schönhage, A. (1987). Equation solving in terms of computational complexity. In *Proceedings of the International Congress of Mathematicians*, pp. 131–153. Berkeley, California, 1986.

Shackell, J. (1990, December). Growth estimates for exp-log functions. *J. Symbolic Computation* **10**, 611–632.

Singer, M.F. (1990). An outline of differential Galois theory. In E. Tournier (ed.), *Computer Algebra and Differential Equations*, pp. 3–57 New York: Academic Press.

Sloane, N.J.A. (1973). *A Handbook of Integer Sequences*. New York: Academic Press.

Soria-Cousineau, M. (1990, July). *Méthodes d'analyse pour les constructions combinatoires et les algorithmes*. Doctorat d'état, Université de Paris-Sud, Orsay.

Stanley, R.P. (1978). Generating functions. In G.-C. Rota (ed.), *Studies in Combinatorics, M.A.A. Studies in Mathematics*, Vol. 17, pp. 100–141. The Mathematical Association of America.

Stanley, R.P. (1980). Differentiably finite power series. *European J. Combinatorics* 1(2), 175–188.

Stanley, R.P. (1986). *Enumerative Combinatorics, Volume I*. Wadsworth and Brooks/Cole.

Steyaert, J.-M. (1984, April). *Structure et complexité des algorithmes*. Doctorat d'état, Université Paris VII.

Thomann, J. (1990). Resommation des séries formelles. Solutions d'équations différentielles linéaires du second ordre dans le champ complexe au voisinage de singularités irrégulières. *Numerische Mathematik* **58**(5), 503–535.

Tournier, É. (1987). *Solutions formelles d'équations différentielles*. Doctorat d'état, Université scientifique, technologique et médicale de Grenoble.

Van der Poorten, A. (1979). A proof that Euler missed .... Apéry's proof of the irrationality of $\zeta(3)$. *Mathematical Intelligencer* **1**, 195–203.

Wilf, H.S., Zeilberger, D. (1992). An algorithmic proof theory for hypergeometric (ordinary and "q") multisum/integral identities. *Inventiones Mathematicae* 108, 575–633.

Wimp, J., Zeilberger, D. (1985). Resurrecting the asymptotics of linear recurrences. *J. Mathematical Analysis and Applications* **111**, 162–176.

Wyman, M. (1959). The asymptotic behavior of the Laurent coefficients. *Canadian J. Mathematics* **11**, 534–555.

Zeilberger, D. (1990). A holonomic systems approach to special functions identities. *J. Computational and Applied Mathematics* **32**(3), 321–368.

Zeilberger, D. (1991). The method of creative telescoping. *J. Symbolic Computation* **11**, 195–204.

Zimmermann, P. (1991). *Séries génératrices et analyse automatique d'algorithmes.* Thèse de Doctorat, École polytechnique, Palaiseau, France.

Zimmermann, P. (1994). Gaïa: A package for the random generation of combinatorial structures. *Maple Tech* **1**(1), 38–46.