PU.M.A. Vol. **17** (2006), No. 1–2, pp. 147–164

A symbolic approach to computing with holonomic functions

Paolo Massazza
Università degli Studi dell'Insubria,
Dipartimento di Informatica e Comunicazione,
Via Mazzini 5, 21100 Varese, Italy
e-mail: paolo.massazza@uninsubria.it

and

Roberto Radicioni
Università degli Studi di Milano,
Dipartimento di Scienze dell'Informazione
Via Comelico 39, 20135 Milano, Italy
e-mail: radicion@dsi.unimi.it

(Received: October 31, 2006)

Abstract. In this work we present a symbolic approach to computing with holonomic functions. More precisely, given a function $f(x)$ suspected to be holonomic, we show an algorithm which computes a linear differential equation with polynomial coefficients satisfied by $f(x)$. The algorithm is based on the construction of a suitable context free grammar which lets us compute a nontrivial linear relation for the derivatives $D^i f(x)$. A prototypical Maple function based on this approach is compared to the Maple function `dfinite_expr_to_diffeq` (in the package Mgfun), and an application to the problem of computing Taylor's coefficients is discussed.

Mathematics Subject Classifications (2000). 68w30, 47e05

1 Introduction

The problem of determining efficient algorithms for computing with holonomic functions is of primary interest due to their importance in many different areas such as combinatorics and theory of languages. In fact, many interesting combinatorial structures admit holonomic generating functions, which are explicitly used for solving classical combinatorial problems (e.g. random generation, counting).

In this context, efficient algorithms for the random generation of combinatorial structures are often based on the computation of coefficients of generating functions (see, for instance, [7, 6]). For example, a random generation method which takes advantage of the properties of the holonomic functions has been proposed in [2], where a linear time algorithm is shown for the problem of random sampling words in a regular language with a fixed number of occurrences of symbols.

Partially supported by Project M.I.U.R. PRIN 2005-2007: *Automata and formal languages: mathematical and application driven studies.*

Because of the increasing interest into holonomic functions, several useful packages have been developed, in particular for the computer algebra system Maple [10, 3].

In this paper, we consider the problem of computing a linear differential equation with polynomial coefficients satisfied by a function $f(x)$ suspected to be holonomic. This problem is usually solved by means of the closure properties of the class of holonomic functions, leading to a method which often presents high running times and also some failures.

Our approach consists of a symbolic computation which finds a nontrivial linear relation for the derivatives $D^i f(x)$ by first computing a suitable context free grammar G (that lets us determine which functions can be obtained by derivating the input function $f(x)$) and then solving an elimination problem in a commutative algebra. So, when the language associated with G turns out to be finite, a finite set of linear equations for $D^i f(x)$ is determined and a linear differential equation satisfied by $f(x)$ is easily computed by gaussian elimination.

By considering this method, we develop a Maple function and compare it to the function `dfinite_expr_to_diffeq` in the Maple package Mgfun. We consider some examples where our method succeeds and computes a differential equation, while `dfinite_expr_to_diffeq` fails or returns a differential equation of higher order.

As a direct application, we consider the problem of computing Taylor's coefficients, showing how to improve the Maple function `coeftayl`.

While the ideas are presented with respect to the univariate case, it is straightforward to generalize the method in order to deal with multivariate functions. In particular, by using the technique illustrated in [9], we could obtain a linear algorithm for computing the coefficients of explicitly known bivariate holonomic functions.

2 Preliminaries

2.1 Holonomic functions

Holonomic functions have been introduced by Bernstein in [1] and deeply investigated by Stanley, Lipshitz, Zeilberger et al. (see [4, 8, 12, 13]). Here we consider holonomic functions in one variable which can be defined in terms of operators of the *Weyl algebra* $A_1(\mathbb{C})$, that is, the noncommutative ring $\mathbb{C}\langle x, D\rangle$ of linear differential operators generated by x and $D = \partial_x$ with the pseudo commutation rule $Dx = xD + 1$.

Definition 1 *A function* $f(x) : \mathbb{C} \mapsto \mathbb{C}$ *is* holonomic *if and only if there exists* $w \in A_1(\mathbb{C})$, $w = \sum_{i=0}^{d} p_i(x) D^i$ $(p_d \neq 0)$, *such that* $w(f) = 0$.

The class of holonomic functions properly contains the class of rational functions $R(x)$ and the class of algebraic functions. Moreover, it admits interesting closure properties which are summarized in the following theorem (see, for instance, [13]).

THEOREM 1 *The class of holonomic functions is closed under the operations of sum, product, indefinite integration, differentiation and right composition with algebraic functions.*

The discrete counterpart of $A_1(\mathbb{C})$ is the so called *shift algebra* $\mathbb{C}\langle n, E_n\rangle$, that is, a particular Ore algebra (see, for instance, [5]) which can be interpreted as a (noncommutative) ring of linear difference operators (acting on sequences) with the pseudo-commutation rule $nE_n = E_n n + E_n$, where E_n is the shift operator. In [12] Stanley introduced the notion of P-recursive sequence (a sequence which is annihilated by a suitable recurrence in $\mathbb{C}\langle n, E_n\rangle$) and proved that a sequence $\{c_n\}$ is P-recursive if and only if the associated formal series $\sum_{n\geq 0} c_n x^n$ is D-finite, that is, it belongs to a particular subclass of formal series that is defined similarly to the class of the holonomic functions.

2.2 Canonical representations

Let A be a set of functions. We say that A is *D-closed* if and only if the derivative of any function in A can be expressed as a finite sum (with coefficients in $R(x)$) of finite products of elements in A. Thus, for any function f in a D-closed set A we have

$$Df(x) = \sum_{i=1}^{k} c_i(x)a_i(x) \quad \text{with} \quad c_i(x) \in R(x),\ a_i(x) = \prod_{j=1}^{e_i} f_{ij}(x),\ f_{ij}(x) \in A.$$

We denote by $\sigma(A)$ the *D-closure degree* of A, defined as the smallest integer k such that there is a D-closed set B including A which contains k functions. If all the D-closed sets including A are not finite then $\sigma(A) = \infty$. For the sake of simplicity, when $A = \{f\}$ we write $\sigma(f)$. We also indicate by CLOSE(A) the closure of A with respect to the operations of sum, product and composition. Moreover, we define Rlin(A) $\supset$ CLOSE(A) as the set of functions obtained as finite linear combinations (with coefficients in $R(x)$) of elements in CLOSE(A). Each element in Rlin(A) is represented by a finite sum of the following type,

$$\sum_{j=0}^{k} r_j(x)a_j(x), \quad r_j(x) \in R(x),\ a_j(x) \in \mathrm{CLOSE}(A),\ a_0(x) = 1, \qquad (1)$$

and can be thought as a *canonical representation* of a suitable function. The terms $a_j(x)$ in (1), $1 \leq j \leq k$, are called *atoms* and are finite products of functions in CLOSE(A), $a_j(x) = \prod_{l=1}^{e_j} t_{jl}(x)$.

Note that different elements in Rlin(A) could be functionally equivalent. For instance, if $A = \{\sin(x), \cos(x)\}$, the two different canonical representations $1 + \sin(x)$ and $\sin(x)^2 + \cos(x)^2 + \sin(x)$ in Rlin(A) are functionally equivalent. Nevertheless, as we will see, this fact is not a problem since our interest is not that of computing normal forms by automatic simplifications.

Note that if A is a finite set of holonomic functions then $\sigma(A) < \infty$. In fact, if $A = \{f_1, \ldots, f_k\}$ and f_i satisfies a differential equation of order e_i then

$$\{f_1(x), Df_1(x), \ldots, D^{e_1}f_1(x), \ldots, f_k(x), Df_k(x), \ldots, D^{e_k}f_k(x)\}$$

is a D-closed set of cardinality $k + \sum_i e_i$.

Some useful properties of $\sigma(A)$ are stated in the following lemma.

Lemma 1 *Let $f(x) = p(x)/q(x)$ with $p(x), q(x) \in \mathbb{C}[x]$ and let $g(x), h(x)$ be two functions such that $\sigma(g(x)) = d$, $\sigma(h(x)) = e$. Then, we have the following properties:*

1. $\sigma(Dg(x)) \leq d + 1$;
2. $\sigma(g(x) + h(x)), \sigma(g(x) \cdot h(x)) \leq d + e + 1$;
3. $\sigma(f(g(x))) \leq \deg(p) + \deg(q) + d + 2$.

Proof. We first consider Property 1. By definition, since $\sigma(g(x)) = d$, we have a D-closed set $\mathcal{F}$ with d elements such that

$$Dg(x) = \sum_{i=1}^{k} c_i(x) \prod_{j=1}^{d} f_j^{m_{ij}}(x), \quad c_i(x) \in R(x),\ f_j(x) \in \mathcal{F},\ m_{ij} \in \mathbb{N}.$$

Then, the derivation rules for the sum and the product imply that $D^2 g(x)$ can be expressed as a finite sum of finite products (with coefficients in $R(x)$) of elements in $\mathcal{F}$. So, the set $\mathcal{F} \cup \{Dg(x)\}$ is a finite D-closed set for $Dg(x)$ and then $\sigma(Dg(x)) \leq d + 1$.

A similar proof holds for Property 2, while for Property 3 we proceed as follows. Consider the functions $G(x) = Dg(x)$, $T_i(x) = (D^i p)(g(x))/q(g(x))$ and $S_i(x) = (D^i q)(g(x))/q(g(x))$, $i \geq 0$. Then, a D-closed set containing $T_0(x) = f(g(x))$ can be found by noting that Property 1 guarantees the existence of a finite D-closed set V containing $G(x)$ and by observing that

$$\begin{array}{rclcrcl}
DT_0 &=& (T_1 - T_0 S_1)G, & \qquad & DS_1 &=& (S_2 - S_1^2)G, \\
DT_1 &=& (T_2 - T_1 S_1)G, & & DS_2 &=& (S_3 - S_1 S_2)G, \\
DT_2 &=& (T_3 - T_2 S_1)G, & & DS_3 &=& (S_4 - S_1 S_3)G, \\
&\vdots& & & &\vdots& \\
DT_{\deg(p)} &=& -T_{\deg(p)} S_1 G, & & DS_{\deg(q)} &=& -S_1 S_{\deg(q)} G.
\end{array}$$

More precisely, let

$$T = \{T_i(x) \mid i = 0, \ldots, \deg(p)\} \quad \text{and} \quad S = \{S_i(x) \mid i = 1, \ldots, \deg(q)\}.$$

Then, a finite D-closed set containing $f(g(x)) = p(g(x))/q(g(x))$ is given by $T \cup S \cup V$. This implies $\sigma(f(g(x))) \leq \deg(p) + \deg(q) + d + 2$. $\square$

By the previous lemma, the finiteness of the D-closure holds for suitable sets of functions. More formally, we have:

Lemma 2 *Let A be a set of functions and B a finite subset of $CLOSE(A)$. If $\sigma(f) < \infty$ for all $f \in A$, then $\sigma(B) < \infty$.*

Proof. We show that $\sigma(B) < \infty$ by proving that $\sigma(f) < \infty$ for all $f \in B$. This is done by induction on the number k of operations $(+, \cdot, \circ)$ used to obtain f.

If $k = 0$ we have $f \in A$ and so $\sigma(f) < \infty$.

Now, let $k > 0$. We have exactly three cases, $f = h+g$, $f = h\cdot g$ and $f = h\circ g$. Note that h and g are obtained from A by using at most $k-1$ operations and so, by induction hypothesis, there are $d, e \in \mathbb{N}$ such that $\sigma(h) = d$ and $\sigma(g) = e$. In the first two cases $(+, \cdot)$, Property 2 in Lemma 1 states that $\sigma(f) \leq d+e+1 < \infty$.

Similarly, let $f = h \circ g$. Since there is a D-closed set $F = \{h_1, \ldots, h_d\}$ which contains h, for $1 \leq k \leq d$ we have

$$Dh_k(x) \;=\; \sum_{i=1}^{e_k} c_{ki}(x) \prod_{j=1}^{d} h_j^{m_{kij}}(x), \quad c_{ki}(x) \in R(x),\; m_{kij}, e_k \in \mathbb{N}.$$

Therefore, since $Dh(g(x)) = Dg(x) \cdot (Dh)(g(x))$ we obtain

$$Dh_k(g(x)) \;=\; Dg(x) \;\cdot\; \sum_{i=1}^{e_k} c_{ki}(g(x)) \prod_{j=1}^{d} h_j^{m_{kij}}(g(x)).$$

Now, Property 3 in Lemma 1 states that $\sigma(c_{ki}(g(x))) = a_{ki} < \infty$ and then, from the previous expression, we get

$$\sigma(h(g(x))) \leq e + \sum_{k,i} a_{ki} + d$$

and so $\sigma(h \circ g) < \infty$. □

Given a function $f \in \mathrm{Rlin}(A)$, we define the *ground set* of f, denoted by GS_f, as the set containing all the factors of the atoms appearing in a canonical representation of a derivative $D^i f(x)$ for some $i \geq 0$. An immediate consequence of the previous lemma is:

Corollary 1 *Let A be a finite D-closed set of functions and $f \in$ Rlin(A). Then GS_f is finite and D-closed.*

In the rest of the paper, we deal with functions belonging to $\mathrm{Rlin}(A)$ for a suitable finite set of functions A satisfying the constraint $\sigma(A) < \infty$.

Example 1 *Consider the function $\sin(\cos(x))$ that belongs to Rlin(A) for the D-closed set $A = \{\sin(x), \cos(x)\}$, $\sigma(A) = 2$. Then, Lemma 2 and Corollary 1 state that $\sigma(\sin(\cos(x))) < \infty$ and $\sharp GS_f < \infty$ respectively. We start with $GS_f = \{\sin(\cos(x))\}$ (the only factor in $D^0 \sin(\cos(x))$). Since $D\sin(\cos(x)) = -\sin(x)\cos(\cos(x))$, the functions $\sin(x)$ and $\cos(\cos(x))$ belong to GS_f. Then, by considering $D^2 \sin(\cos(x)) = -\cos(x)\cos(\cos(x)) - \sin(x)^2 \sin(\cos(x))$, we add to GS_f the function $\cos(x)$. Since no new factors appear in $D^3 \sin(\cos(x))$ we obtain*

$$GS_f = \{\sin(\cos(x)), \cos(\cos(x)), \sin(x), \cos(x)\}.$$

Lastly, note that GS_f is D-closed and that $\sigma(\sin(\cos(x))) = 4$.

3 The problem

In our setting, the problem of computing a linear differential equation satisfied by a given function can be formulated as follows.

Problem CAF (Computing Annihilators for Functions).

Input: A function $f(x) \in \mathrm{Rlin}(A)$ for a suitable finite D-closed set A.

Output: $w \in A_1(\mathbb{C})$ s.t. $w(f(x)) = 0$, (if such w exists, i.e. $f(x)$ is holonomic), 0 otherwise.

This problem arises, for instance, when we want to compute efficiently the Taylor's coefficients of a given function. This is of particular interest in combinatorics when we know the generating function of a combinatorial structure and we need the exact solution of the counting problem associated with it.

A first way of solving CAF is based on the closure properties of the class of holonomic functions. More precisely, we can define a bottom-up algorithm which traverses the expression tree associated with a function, from the leaves towards the root. The leaves correspond to polynomials, rational, algebraic or elementary functions for which we can directly compute an annihilator. The internal nodes are labelled with an operation for which a closure property exists (e.g. sum, product, right composition with algebraic functions). In order to get the annihilator associated with an expression tree rooted at a node p we recursively compute the annihilators associated with the subtrees of p and then we apply the algorithm associated with the closure property identified by the label of p.

The function `dfinite_expr_to_diffeq` in the Maple package Mgfun works similarly. For example, if $f(x) = x^2 \sin(x) + \sqrt{1 - x^3}$ we have the following computation:

1. compute an annihilator for x^2: $-2 + xD$;
2. compute an annihilator for $\sin(x)$: $1 + D^2$;
3. compute an annihilator for $\sqrt{1 - x^3}$: $-3x^2 + (2x^3 - 2)D$;
4. compute an annihilator for the product $x^2 \sin(x)$: $6 + x^2 - 4xD + x^2 D^2$;
5. compute an annihilator for the sum $x^2 \sin(x) + \sqrt{1 - x^3}$:
$$p_0 + p_1 D + p_2 D^2 + p_3 D^3,$$
where
$$\begin{aligned} p_0 &= -120x^4 - 432x^5 - 12x^6 + 204x^7 + 24x^9 - 54x^8 - 165x^{10} - 12x^{12}, \\ p_1 &= -96 - 128x^2 + 240x^3 - 8x^4 + 408x^5 + 132x^6 + 24x^7 - 414x^8 \\ &\quad + 48x^9 - 24x^{10} + 134x^{11} + 8x^{13}, \\ p_2 &= 96x + 32x^3 - 240x^4 - 108x^6 + 84x^7 + 120x^9 - 21x^{10} - 44x^{12}, \\ p_3 &= -48x^2 - 8x^4 + 120x^5 + 24x^7 - 78x^8 - 24x^{10} + 6x^{11} + 8x^{13}. \end{aligned}$$

This approach has two serious drawbacks. First, it does not work for all holonomic functions. In fact, let us consider the function

$$y(x) = \sin(t_1(x)) \quad \text{where} \quad t_1(x) = \int t_2(x)dx,\ t_2(x) = \sqrt{x^4 - 1}$$

and note that it belongs to $\mathrm{Rlin}(A)$ for the D-closed set

$$A = \{\sin(x), \cos(x), t_1(x), t_2(x)\}, \quad Dt_1(x) = t_2(x),\ Dt_2(x) = \frac{2x^3}{x^4 - 1}t_2(x).$$

Moreover, $y(x)$ is the composition of the sine function with a nonalgebraic holonomic function ($t_1(x)$ can be expressed in terms of an elliptic integral) and satisfies the differential equation

$$((x^4 - 1)Dx^2 - 2x^3Dx + x^8 - 2x^4 + 1)y(x) \ = \ 0. \tag{2}$$

Nevertheless, the function `dfinite_expr_to_diffeq` does not recognize $y(x)$ as holonomic since $y(x)$ is not the composition of a holonomic function with an algebraic one.

Second, in some cases the order of the differential equation which is returned is $\Omega(2^h)$, where h is the height of the expression tree. This depends on the fact that the order of the differential equation which is returned for the product $f(x) \cdot g(x)$ is in general the product of the orders of the differential equations associated with $f(x)$ and $g(x)$ respectively (see section 4.1 in [13]).

On the other hand, we rarely deal with functions that satisfy linear differential equations of very high order. In other words, given a canonical representation in $\mathrm{Rlin}(A)$ associated with a holonomic function $f(x)$, the number of derivatives $D^i f(x)$ which are linearly independent is usually a small integer. So, if we suspect that this is true, a direct way of solving CAF could be that of finding the smallest integer k such that $\{D^i f(x) \mid 0 \leq i \leq k\}$ is a set of linearly dependent functions (over $R(x)$), and then computing suitable polynomials $p_j(x)$ such that $\sum_{j=0}^{k} p_j(x) D^j f(x) = 0$. In the next section we illustrate an algorithm that follows this approach.

4 Finding an annihilator: the algorithm

In this section we describe an algorithm for solving CAF. It takes as input a canonical representation of a function $f(x) \in \mathrm{Rlin}(A)$ (for a suitable D-closed set A) and return either an annihilator $w \in A_1(\mathbb{C})$ or 0. The algorithm works in four main steps:

Step 1: compute GS_f;

Step 2: use GS_f to test whether $f(x)$ is holonomic;

Step 3: if $f(x)$ is holonomic find an integer k such that the derivatives $D^i f(x)$, $0 \leq i \leq k$, are expressed as linear combinations of k atoms;

Step 4: construct a $(k+1) \times k$ linear system and use it to compute $w \in A_1(\mathbb{C})$ such that $w(f(x)) = 0$.

4.1 Step 1

From the canonical representation of the input function $f(x)$,

$$f(x) = \sum_{i=1}^{k} r_i(x) \prod_{j=1}^{p} t_j^{m_{ij}}(x),$$

we determine a finite set B of nonrational functions,

$$B = \{t_1(x), \ldots, t_p(x)\},$$

consisting of the factors of the atoms of $f(x)$. Then, Corollary 1 guarantees the existence of a finite D-closed ground set for $f(x)$,

$$B \subseteq GS_f = \{t_1(x), \ldots, t_p(x), \ldots, t_q(x)\},$$

which can be easily obtained by iterating the symbolic derivative of the functions in B, as long as this process produces new factors. Procedure GroundSet in Figure 1 illustrates such computation. It accepts as input the set B of factors in a canonical representation of a function f and returns GS_f together with a system of equations for the derivatives of the elements in GS_f,

$$\begin{aligned} Dt_1(x) &= \sum_{i=1}^{k_1} c_{1i}(x) \prod_{j=1}^{q} t_j^{m_{1ij}}(x), \\ &\vdots \\ Dt_q(x) &= \sum_{i=1}^{k_q} c_{qi}(x) \prod_{j=1}^{q} t_j^{m_{qij}}(x), \end{aligned} \tag{3}$$

where $c_{ij}(x)$ are suitable rational functions.

The function CanonicalRep(f, x) computes a canonical representation of $f(x)$ while GetFactors(r, x) returns the set of factors of the atoms in the canonical representation $r(x)$.

4.2 Step 2

Procedure GroundSet(B) returns a couple (GS_f, S) where S is the system (3). Now, the idea is that of associating with this system a language that lets us determine which atoms eventually occur in a derivative $D^k t_j(x)$.

More formally, let $\Sigma = \{\tau_1, \ldots, \tau_q\}$ and define $L_f \subseteq \Sigma^\star$ as

$$L_f = \{\tau_1^{i_1} \cdots \tau_q^{i_q} \mid \exists k, j,\ t_1^{i_1}(x) \cdots t_q^{i_q}(x) \text{ occurs in } D^k t_j(x)\}.$$

It is immediate to see that if L_f is finite then $t_1(x), \ldots, t_q(x)$ and $f(x)$ are holonomic. Otherwise ($\sharp L_f = \infty$) there is $I \subseteq \{1, \ldots, q\}$ such that, for any $k > 0$ and $i \in I$, we can find a word $\tau_1^{e_1} \cdots \tau_i^{e_i} \cdots t_q^{e_q}$ in L_f with $e_i > k$.

```
Procedure GroundSet(B)
begin
    GS_f ← B; N ← B; S ← ∅;
    while N ≠ ∅
        A ← ∅;
        for t ∈ N
            r ← CanonicalRep(Diff(t, x), x);
            M ← GetFactors(r, x)\GS_f;
            A ← A ∪ M; GS_f ← GS_f ∪ M; S ← S ∪ {(t, r)};
        end for
        N ← A;
    end while
    return (GS_f, S);
end
```

Figure 1: Procedure GroundSet.

Now, note that if I contains only indices referring to algebraic entries, then an atom $t_1^{e_1}(x) \cdots t_i^{e_i}(x) \cdots t_q^{e_q}(x)$ can be replaced with a linear combination of atoms with bounded degree in the algebraic entry $t_i(x)$ (by using the polynomial equation satisfied by $t_i(x)$). Hence, the problem is when I contains indices associated with transcendental entries: in this case we can not conclude that $f(x)$ is holonomic.

In order to compute the indices of I associated with transcendental entries, we consider a suitable context free grammar that is obtained from the system of equations associated with GS_f. Without loss of generality, we suppose that the first h factors $t_1(x), \ldots, t_h(x)$ of GS_f are transcendental, while $t_{h+1}(x), \ldots, t_q(x)$ are algebraic.

So, we define the grammar $G_f = \langle V, \Sigma, P, S \rangle$ where $V = \{T_1, \ldots, T_h, S\}$, $\Sigma = \{\tau_1, \ldots, \tau_h\}$ and P is a suitable set of productions such that, if T_i generates w, then w corresponds to the product of the transcendental factors of an atom occurring in a canonical representation of $D^k t_i(x)$, for a suitable $k \geq 0$. We obtain the productions for T_i from the equation associated with $Dt_i(x)$, by defining a production for each atom admitting a transcendental factor (note that the start symbol S is associated with $f(x)$). More formally, with respect to System (3) the set of productions P is given by

$$
\begin{aligned}
S &\to T_1 \mid T_2 \mid \ldots \mid T_h, \\
T_1 &\to \tau_1 \mid T_1^{m_{111}} T_2^{m_{112}} \cdots T_h^{m_{11h}} \mid \ldots \mid T_1^{m_{1k_1 1}} T_2^{m_{1k_1 2}} \cdots T_h^{m_{1k_1 h}}, \\
&\vdots \\
T_h &\to \tau_h \mid T_1^{m_{h11}} T_2^{m_{h12}} \cdots T_h^{m_{h1h}} \mid \ldots \mid T_1^{m_{hk_h 1}} T_2^{m_{hk_h 2}} \cdots T_h^{m_{hk_h h}}.
\end{aligned}
$$

We say that $L(G_f)$ is the *transcendental language generated* by $f(x)$. By construction, I contains indices which refer to transcendental entries if and only if $\sharp L(G_f) = \infty$. Moreover, note that the indices in I referring to algebraic entries are associated with finite languages. More precisely, if $e \in I$ and $t_e(x)$ is algebraic (and satisfies a polynomial equation of degree d) then the language $L_e = \{\varepsilon, \tau_e, \tau_e^2, \ldots, \tau_e^{d-1}\}$ corresponds to the set of powers of $t_e(x)$ which eventually appear in an atom of $D^k f(x)$.

Example 2 *Let $A = \{\exp(x), \arcsin(x), \sqrt[3]{x}\}$ and consider the function $f(x) = \exp(\arcsin(\sqrt[3]{x} - 1)) \in Rlin(A)$. Then, $GS_f = \{t_1(x), t_2(x), t_3(x)\}$, where*

$$t_1(x) = \exp(\arcsin(\sqrt[3]{x} - 1)), \quad t_2(x) = \frac{1}{\sqrt[3]{x^2}}, \quad t_3(x) = \frac{1}{\sqrt{1 - (\sqrt[3]{x} - 1)^2}}.$$

Indeed, the system associated with it is

$$Dt_1(x) = \frac{1}{3} t_2(x) t_3(x) t_1(x), \qquad Dt_2(x) = \frac{2x}{3} t_2(x)^2,$$
$$Dt_3(x) = \frac{1}{3} t_2(x)^2 t_3(x)^3 \; - \; \frac{1}{3} t_2(x) t_3(x)^3.$$

Then, $G_f = \langle \{S, T_1\}, \{\tau_1\}, \{S \to T_1, T_1 \to \tau_1\}, S\rangle$ and $L(G_f) = \{\tau_1\}$, while the finite languages $L_2 = \{\epsilon, \tau_2\}$ and $L_3 = \{\epsilon, \tau_3, {\tau_3}^2, {\tau_3}^3, {\tau_3}^4, {\tau_3}^5\}$ are associated with the algebraic terms t_2 and t_3, since they satisfy $x^2 t_2(x)^3 - 1 = 0$ and $(x^2 - 8x) t_3(x)^6 + 6x t_3(x)^4 + 1 = 0$.

The importance of the language $L(G_f)$ is shown in the following theorem.

Theorem 2 *Let $f(x)$ be a function such that $L(G_f)$ is finite. Then $f(x)$ is holonomic.*

Proof. If $L(G_f)$ is finite then the cardinality of the set of atoms which eventually occur in a canonical representation of $D^i f(x)$ is finite, say k. Hence, the $k+1$ derivatives $D^i f(x)$, $0 \le i \le k$, are expressed as linear combinations of k atoms and are linearly dependent (over $R(x)$), that is, $f(x)$ satisfies a linear differential equation of order at most k and so is holonomic. □

We point out that if $L(G_f)$ is not finite, by a simple analysis of the words in $L(G_f)$ we can often prove that $f(x)$ is not holonomic.

Example 3 *Recalling Example 1, the system associated with the ground set $\{\sin(\cos(x)), \cos(\cos(x)), \sin(x), \cos(x)\}$ is*

$$\begin{aligned} D\sin(\cos(x)) &= -\sin(x)\cos(\cos(x)), \\ D\cos(\cos(x)) &= \sin(x)\sin(\cos(x)), \\ D\sin(x) &= \cos(x), \\ D\cos(x) &= -\sin(x). \end{aligned}$$

This leads to the grammar $G_f = \langle \{T_1, T_2, , T_3, T_4, S\}, \{\tau_1, \tau_2, \tau_3, \tau_4\}, P, S\rangle$, where P is the set

$$\{T_1 \to \tau_1 | T_3 T_2,\ T_2 \to \tau_2 | T_3 T_1,\ T_3 \to \tau_3 | T_4,\ T_4 \to \tau_4 | T_3,\ S \to T_1 | T_2 | T_3 | T_4\}$$

with $\tau_1 \equiv \sin(\cos(x))$*,* $\tau_2 \equiv \cos(\cos(x))$*,* $\tau_3 \equiv \sin(x)$ *and* $\tau_4 \equiv \cos(x)$*. It is easy to see that* $L(G_f)$ *is not finite. In particular, we have* $\{\tau_3^{2k}\tau_1 \mid k \geq 0\} \subseteq L(G_f)$*. This means that* $\{D^i \sin(\cos(x)), i \geq 0\}$ *is a set of linearly independent functions since the set of atoms associated with it contains* $\{\sin(x)^{2k} \sin(\cos(x))\}$ *which is a set of linearly independent functions over* $R(x)$*. Therefore,* $\sin(\cos(x))$ *is not holonomic.*

EXAMPLE 4 *Let* $f(x) = \sin(y(x)) \in Rlin\,(A)$ *with* $y(x) = \int \sqrt{x^4-1}dx$ *and* $A = \{\sin(x), y(x)\}$*. A ground set for* $f(x)$ *is given by* $\{t_1(x), t_2(x), t_3(x)\}$ *where* $t_1(x) = \sin(y(x))$*,* $t_2(x) = \cos(y(x))$ *and* $t_3(x) = \sqrt{x^4-1}$*. The system associated with it is*

$$Dt_1(x) = t_3(x)t_2(x), \qquad Dt_2(x) = t_3(x)t_1(x), \qquad Dt_3(x) = \frac{2x^3}{x^4-1}t_3(x).$$

The grammar associated with $f(x)$ *is obtained by ignoring the algebraic term* $t_3(x)$*:*

$$G_f = \langle \{T_1, T_2, S\}, \{\tau_1, \tau_2\}, \{S \to T_1|T_2,\ T_1 \to \tau_1|T_2,\ T_2 \to \tau_2|T_1\}, S\rangle\,.$$

Since $L(G_f)$ *is finite,* $f(x)$ *is holonomic. In fact, it satisfies Equation* (2)*.*

The algorithm halts and returns 0 if $f(x)$ is not recognized as holonomic, otherwise it continues to Step 3.

4.3 Step 3

Here we know that $f(x)$ is holonomic and that canonical representations of $D^i f(x)$, $i \geq 0$, can be written using a finite set of atoms. Thus, we compute an integer $\hat{k} \leq \sharp\, L(G_f)$ such that $\hat{k}+1$ canonical representations of the derivatives $D^i f(x)$, $0 \leq i \leq \hat{k}$, are expressed by means of $\hat{k}$ atoms.

EXAMPLE 5 *Consider the function* $f(x) = x\sin(x)\cos(x) + (1+2x)\exp(x)$ *with* $GS_f = \{\sin(x), \cos(x), \exp(x)\}$ *and consider the derivatives*

$$\begin{aligned}
D^0 f(x) &= (1+2x)a_{11} + xa_{12},\\
Df(x) &= (3+2x)a_{11} + a_{12} - xa_{13} + xa_{14},\\
D^2 f(x) &= (5+2x)a_{11} - 4xa_{12} - 2a_{13} + 2a_{14},\\
D^3 f(x) &= (7+2x)a_{11} - 12a_{12} + 4xa_{13} - 4xa_{14},\\
D^4 f(x) &= (9+2x)a_{11} + 16xa_{12} + 16a_{13} - 16a_{14},
\end{aligned}$$

where $a_{11} = \exp(x)$*,* $a_{12} = \sin(x)\cos(x)$*,* $a_{13} = \sin(x)^2$ *and* $a_{14} = \cos(x)^2$*. Here we have* $\hat{k} = 4$*, that is, we can express 5 derivatives by 4 atoms. Note that all the atoms we need to express a derivative* $D^i f(x)$ *are found by computing* $Df(x)$*.*

We show how to determine $\hat{k}$ by an iterative process. Let a_i denote the number of different atoms which appear (at least once) in a canonical representation of $D^k f(x)$ for a suitable k, $0 \leq k \leq i$. So, let $e = \sharp GS_f$ and start with the canonical representation of the input function $f(x)$,

$$D^0 f(x) = \sum_{j=1}^{a_0} r_{0j}(x) a_{0j}(x),$$

where $a_0 \geq 1$, $a_{0j}(x) = \prod_{l=1}^{e} t_l^{m_{l0j}}(x)$, $m_{l0j} \in \mathbb{N}$ and $t_l(x) \in GS_f$.

Then, for $i \geq 1$, we proceed to the ith iteration and compute a canonical representation of $D^i f(x)$ from a canonical representation of $D^{i-1} f(x)$ if and only if $a_{i-1} \geq i$. Note that, in general, the computation of $D(D^{i-1} f(x))$ may generate new atoms, that is, $a_i > a_{i-1}$. Nevertheless, the number of different atoms in the derivatives $D^k f(x)$, $k \geq 0$, is finite (as determined in Step 2). More precisely, this number is bounded by $\sharp L(G_f) \cdot \prod_{j=h+1}^{q} \sharp L_j$, where L_j are the (finite) languages associated with the algebraic entries in GS_f.

Thus, we eventually find an integer $\hat{\imath}$ such that $a_{\hat{\imath}} = a_{\hat{\imath}-1}$, and then $a_{\hat{\imath}+n} = a_{\hat{\imath}}$ for all $n \geq 0$. Once we have found $\hat{\imath}$, we reach the $\hat{k}$th iteration where $\hat{k} = a_{\hat{\imath}}$. Now, we have $\hat{k}+1$ elements (the derivatives $D^i f(x)$, $0 \leq i \leq \hat{k}$) which are expressed as linear combinations of $\hat{k}$ atoms $a_1(x), \ldots, a_{\hat{k}}(x)$. Procedure CountAtoms in Figure 2 illustrates such computation. It accepts as input a canonical representation r of a function $f(x)$ and returns a couple (c, d) where d is a system of c linear equations defining the derivatives $D^i f(x)$, $0 \leq i < c$, in terms of $c-1$ different atoms.

Procedure CountAtoms(r, x)
begin
    d[0] $\leftarrow r$;
    A $\leftarrow$ GetAtoms(d[0], x);
    $c \leftarrow 1$;
    while $c \leq \sharp$A
        d[c] $\leftarrow$ CanonicalRep(Diff(d[$c-1$], x), x);
        B $\leftarrow$ GetAtoms(d[c], x);
        A $\leftarrow$ A $\cup$ (B$\backslash$A);
        $c \leftarrow c+1$;
    end while
    return (c, d);
end

Figure 2: Procedure CountAtoms.

Observe that, in the code in Figure 2, the function GetAtoms(r, x) is used to compute the set of atoms of a canonical representation $r(x)$.

4.4 Step 4

Procedure CountAtoms in Step 3 returns the following system

$$D^i f(x) = \sum_{j=1}^{\hat{k}} r_{ij}(x) a_j(x), \qquad \text{with} \quad r_{ij}(x) \in R(x), \quad i = 0, \ldots, \hat{k},$$

which can be written as $\mathbf{R} \cdot \mathbf{a} = \mathbf{v}$ by setting $\mathbf{R} = [r_{ij}(x)]_{(\hat{k}+1)\times\hat{k}}$ and

$$\mathbf{a} = (a_1(x), \ldots, a_{\hat{k}}(x))^T, \quad \mathbf{v} = (D^0 f(x), Df(x), \ldots, D^{\hat{k}} f(x))^T.$$

Then, an annihilator $w \in A_1(\mathbb{C})$ for the function $f(x)$ is easily determined by considering

$$\det(\mathbf{v} \mid \mathbf{R}) = 0,$$

where $\mathbf{v} \mid \mathbf{R}$ is the augmented matrix of the system. If this determinant is zero, then two or more atoms are linearly dependent. In this case, we first apply the gaussian elimination to $\mathbf{v} \mid \mathbf{R}$ and then we compute the determinant of a square submatrix of the reduced echelon form of $\mathbf{v} \mid \mathbf{R}$.

Example 6 *The augmented matrix* $\mathbf{v} \mid \mathbf{R}$ *for Example 5 is*

$$\begin{bmatrix} D^0 f(x) & (1+2x) & x & 0 & 0 \\ Df(x) & (3+2x) & 1 & -x & x \\ D^2 f(x) & (5+2x) & -4x & -2 & 2 \\ D^3 f(x) & (7+2x) & -12 & 4x & -4x \\ D^4 f(x) & (9+2x) & 16x & 16 & -16 \end{bmatrix}$$

Since the matrix is singular, we obtain an annihilator $w \in A_1(\mathbb{C})$ by computing the determinant for the square submatrix of its reduced echelon form. So, we obtain

$$\begin{aligned} w = {} & (31x + 20x^2 + 20x^3 + 12)D^4 + (-65x - 50x^2)D^3 \\ & + (96 + 248x + 30x^2 + 60x^3)D^2 - 588 - 104x - 200x^2 - 80x^3. \end{aligned}$$

Procedure CompAnnihil in Fig. 3 illustrates the structure of Step 4. It takes as input a couple (c, d) computed in Step 3 by Procedure CountAtoms, that is, a vector d of c linear equations (one for each derivative $D^i f(x)$, $0 \le i < c$) in $c - 1$ atoms. It computes (lines 1–7) the augmented matrix $M = \mathbf{v} \mid \mathbf{R}$ and its echelon form (by using the Maple routine `GaussElim`). Function Coeff(r, a) returns the coefficient of an atom a in a canonical representation r. Lastly (lines 8, 9), the determinant of a square nonsingular submatrix of the echelon form of M is returned: this corresponds to the differential equation satisfied by $f(x)$.

For the sake of simplicity, steps 3 and 4 of the algorithm can be joined in a single procedure FunToDiffeq which accepts as inputs an expression defining a holonomic function and a variable name.

Procedure CompAnnihil(c, d)
begin
  for $i = 1$ to c
    for all j such that $a_j \in \mathrm{A}$
      $M[i, j] \leftarrow$ Coeff(d$[i-1]$, a_j);
    end for
    $M[i, c] \leftarrow f^{(i-1)}$;
  end for
  echelonForm $\leftarrow$ GaussElim(M);
  r $\leftarrow$ Rank(echelonForm);
  return det(echelonForm$[1 \ldots r, 1 \ldots r]$);
end

Figure 3: Procedure CompAnnihil.

Procedure FunToDiffeq(expr, x)
begin
  $r \leftarrow$ CanonicalRep(expr, x);
  $(c, d) \leftarrow$ CountAtoms(r, x);
  return CompAnnihil(c, d);
end

Figure 4: Procedure FunToDiffeq.

5 Applications and examples

5.1 Computing Taylor's coefficients

We can compute the nth Taylor's coefficient of an analytic function by using the Maple function `coeftayl`. Given an expression `expr` representing a function $f(x)$, an integer `n` and a real value x_0, `coeftayl(expr,x=`x_0`,n)` returns the nth coefficient of the Taylor's expansion of $f(x)$ at $x = x_0$ by using the known formula

$$\frac{D^n f(x_0)}{n!}. \tag{4}$$

So, if n is big enough, the time to compute all the symbolic derivatives $D^i f(x)$, $i = 1, \ldots, n$, might be unacceptable. When we deal with a holonomic function, it is possible to exploit the linear recurrence equation satisfied by its Taylor's coefficients in order to compute efficiently the required coefficient. In this case, the performance of the algorithm depends only on n and the growth rate of the coefficients. Thus, we can define a function that takes advantage of the linear recurrence rec $\in \mathbb{C}\langle n, E_n \rangle$ associated with an annihilator $w \in A_1(\mathbb{C})$ which is computed when the input expression is recognized as holonomic.

Let $w \in A_1(\mathbb{C})$ be an annihilator of $f(x)$ and d its degree with respect to D.

The idea is that, if $n \gg d$, the time spent to find w might be significantly less than the time for computing all the derivatives until $D^n f(x)$.

So, we can define two modified versions of FunToDiffeq and CountAtoms which have n as additional input parameter. Then, CountAtoms runs by having n as a bound to the number of iterations, returning a system of equations of dimension less or equal to n (if such a system exists) or reporting a failure otherwise.

In the first case, FunToDiffeq computes an annihilator $w \in A_1(\mathbb{C})$ and its associated recurrence rec $\in \mathbb{C}\langle n, E_n \rangle$ (together with suitable initial conditions obtained by evaluating the derivatives at $x = x_0$). Then, the nth Taylor's coefficient of $f(x)$ is determined with $O(n)$ additional operations.

In the second case, the last canonical representation computed by CountAtoms is associated with $D^n f(x)$ and then Formula (4) can be used. This last case represents in some way the worst case, for which we can observe a running time which is proportional to the running time of the original function `coeftayl`.

5.2 Experimental results

The MGFUN package contains a function `dfinite_expr_to_diffeq` which accepts as input an analytic expression, supposed to represent a holonomic function, and returns a homogeneous linear differential (or difference) equation with polynomial coefficients. This function works also in the case of hybrid multivariate functions containing both continuous and discrete variables. Nevertheless, it recognizes only a subclass of holonomic functions, defined by the closure properties associated with the holonomic class (see Theorem 1).

In this section we present some examples of holonomic functions which are not recognized by `dfinite_expr_to_diffeq` but are successfully elaborated by FunToDiffeq. Further, we consider some functions for which the order of the differential equation returned by FunToDiffeq is less than the order of the differential equation returned by `dfinite_expr_to_diffeq`.

We point out that the GFUN package (the older Maple package dealing with univariate holonomic functions) contains a function `holexprtodiffeq` which exhibits the same behaviour of the function `dfinite_expr_to_diffeq` on univariate input expressions. In the following examples we refer only to `dfinite_expr_to_diffeq`, nevertheless the same results have been obtained by considering `holexprtodiffeq`.

Example 7 *Let $f(x) = \sin(\log(x^2+1))$. While we can immediately see that this function is holonomic (by using the exponential form of the sine function and doing a trivial simplification), we have*

$$GS_f = \{\sin(\log(x^2+1)), \cos(\log(x^2+1))\},$$

and $\sharp L(G_f) = 2$. So, $f(x)$ is holonomic by Theorem 2. Note that the function `dfinite_expr_to_diffeq` *does not recognize $f(x)$ as holonomic since it does* *not simplify the input expression and it can not apply the closure properties of the holonomic class.* FunToDiffeq *computes the augmented matrix*

$$\begin{bmatrix} D^0 f(x) & 1 & 0 \\ D^1 f(x) & 0 & 2x/(x^2+1) \\ D^2 f(x) & -4x^2/(x^4+2x^2+1) & (-2x^2+2)/(x^4+2x^2+1) \end{bmatrix}$$

and returns the annihilator

$$(x^5+2x^3+x)D^2+(x^4-1)D+4x^3.$$

The function $f(x) = \exp(\arctan(x))$ provides us with another simple example where we succeed. In this case, the annihilator is $(1+x^2)D^2-1$.

The method works also with some functions which are implicitly defined by integrals. We only require that the function satisfies a derivation rule which defines a finite D-closed set. For instance, consider the function $f(x) = \exp(-1)Ei(1, 1-x)$, where Ei is the exponential integral $Ei(a, x) = \int_1^\infty t^{-a} \exp(-tx)dt$ (defined for positive real values of x). The derivative of $f(x)$ is $(1-x)^{-1}\exp(-x)$. While `dfinite_expr_to_diffeq` does not recognize it as a holonomic function, FunToDiffeq returns the annihilator $(x-1)D^2+xD$.

We can also observe that the order of the differential equation returned by `dfinite_expr_to_diffeq` is not always as small as possible. We present here two examples where FunToDiffeq finds an annihilator of smaller degree in D.

Example 8 *Consider the Error function erf$(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt$. Its indefinite integral is the function*

$$f(x) = x\ \mathit{erf}(x) + \frac{\exp -x^2}{\sqrt{\pi}}.$$

`dfinite_expr_to_diffeq` *returns an equation of order 3,*

$$D^3f(x)+4xD^2f(x)+4x^2Df(x)-4xf(x) \;=\; 0$$

while FunToDiffeq *returns the annihilator* $D^2+2xD-2$.

Example 9 *A more interesting result can be obtained by considering the sequence of products*

$$f(x) = \sin(x)\cos(x)\sinh(x)\sqrt{1-x}\cosh(x).$$

FunToDiffeq *returns the following annihilator of order* 4,

$$\begin{aligned}&(16x^4-64x^3+96x^2-64x+16)D^4+(-32x^3+96x^2+32-96x)D^3\\ &\quad+(72x^2-144x+72)D^2+(-120x+120)D\\ &\quad+(1024x^4-4096x^3+6144x^2-4096x+1129),\end{aligned}$$

while `dfinite_expr_to_diffeq` *returns an equation of order* 9 *(the result is not reported here because the symbolic expression is very large, but you can easily compute it in a Maple session).*

6 Conclusions

In this work we have described an algorithm which takes as input a function $f(x)$, supposed to be holonomic, and returns a linear differential equation satisfied by $f(x)$, if $f(x)$ is recognized as holonomic. Our technique is based on a symbolic manipulation of the expressions representing the derivatives $D^i f(x)$ and leads to a sufficient condition for a function to be holonomic, given in Theorem 2. If $L(G_f)$ is finite, then $f(x)$ is provably holonomic and an annihilator for it can be found by means of a simple process based on linear elimination in a commutative algebra. Actually, we have not proved that the finiteness of $L(G_f)$ (the transcendental language generated by $f(x)$) is a necessary condition for $f(x)$ to be holonomic. However, we have no examples of holonomic functions which generate infinite languages. Therefore, it would be interesting to look for a nontrivial class A of functions such that

1. $\sigma(A) < \infty$;

2. for any $f(x) \in \mathrm{Rlin}(A)$, $f(x)$ is holonomic if and only if $L(G_f)$ is finite.

This problem seems to be related to that of zero-equivalence in function fields obtained as towers over the field of rational functions (see, for instance, [11]); we think that a deeper investigation would lead to a suitable structure theorem.

In many cases, the running time of our prototypical implementation can be positively compared to that of `dfinite_expr_to_diffeq`. A formal investigation about the complexity of our method is in progress, with the goal of theoretically supporting our experimental results.

All the examples have been tested using the Maple system and the packages Mgfun (written by Chyzak et al. [3, 4, 5]) and Gfun (written by Salvy et al. [10]). A prototypical implementation of the algorithm is available at the link `http://homes.dsi.unimi.it/~radicion/holonomicUtility/`.

References

[1] I. N. Bernšteĭn, *Modules over a ring of differential operators. An investigation of the fundamental solutions of equations with constant coefficients*, Funkcional. Anal. i Priložen., **5** (2) (1971), 1–16.

[2] A. Bertoni, P. Massazza, and R. Radicioni, *Random generation of words in regular languages with fixed occurrences of symbols*, In: Proceedings of WORDS'03, volume 27 of TUCS Gen. Publ., pages 332–343. Turku Cent. Comput. Sci., Turku, 2003.

[3] F. Chyzak, *Holonomic systems and automatic proofs of identities*, Technical Report 2371, Institut National de Recherche en Informatique et en Automatique, October 1994.

[4] F. Chyzak, *An extension of Zeilberger's fast algorithm to general holonomic functions*, Discrete Math., **217** (1-3) (2000), 115–134. Formal power series and algebraic combinatorics (Vienna, 1997).

[5] F. Chyzak and B. Salvy, *Non-commutative elimination in Ore algebras proves multivariate identities*, J. Symbolic Comput., **26** (2) (1998), 187–227.

[6] A. Denise, *Génération aléatoire uniforme de mots de langages rationnels*, Theoret. Comput. Sci., **159** (1) (1996), 43–63. Selected papers from the "GASCOM '94" (Talence, 1994) and the "Polyominoes and Tilings" (Toulouse, 1994) Workshops.

[7] P. Flajolet, P. Zimmerman and B. Van Cutsem, *A calculus for the random generation of labelled combinatorial structures*, Theoret. Comput. Sci., **132** (1–2) (1994), 1–35.

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

[9] P. Massazza and R. Radicioni, *On computing the coefficients of bivariate holonomic formal series*, Theoret. Comput. Sci., **346** (2–3) (2005), 418–438.

[10] B. Salvy and P. Zimmerman, *Gfun: a maple package for the manipulation of generating and holonomic functions in one variable*, ACM Trans. Math. Softw., **20** (2) (1994), 163–177.

[11] J. Shackell, *Zero-equivalence in function fields defined by algebraic differential equations*, Trans. Amer. Math. Soc., **336** (1) (1993), 151–171.

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

[13] D. Zeilberger, *A holonomic systems approach to special functions identities*, J. Comput. Appl. Math., **32** (3) (1990), 321–368.