Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 346 (2005) 418–438

Theoretical Computer Science

www.elsevier.com/locate/tcs

On computing the coefficients of bivariate holonomic formal series[☆]

P. Massazza[a,*], R. Radicioni[b]

[a] Università degli Studi dell'Insubria, Dipartimento di Informatica e Comunicazione, Via Mazzini 5, 21100 Varese, Italy

[b] Università degli Studi di Milano, Dipartimento di Scienze dell'Informazione, via Comelico 39, 20135 Milano, Italy

Abstract

In this work, we study the problem of computing the coefficients of holonomic formal series in two commuting variables. Given a formal series $\phi(x, y) = \sum_{n,k \geqslant 0} c_{nk} x^n y^k$ specified by a holonomic system $\sum_{j=0}^{d_1} p_j(x, y)\partial_x^j \phi = 0$ and $\sum_{j=0}^{d_2} q_j(x, y)\partial_y^j \phi = 0$, with a suitable finite set of initial conditions $\{[x^a y^b]\phi(x, y)\}$, we show that the coefficient $[x^i y^j]\phi(x, y)$ can be computed in time $O(i + j)$ under the uniform cost criterion.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Formal series; Coefficient problem; Holonomic systems

1. Introduction

The problem of computing the coefficients of formal power series (known as the *coefficient problem*) is of primary interest in many different areas such as combinatorics and theory of languages. For example, the problem of counting objects with a given property that belong to a combinatorial structure S can be reduced to computing the coefficients of suitable formal power series: we represent the property by a weight function $w : S \to \mathbb{N}$

[☆] This work has been supported by the Project M.I.U.R. COFIN 2003–2005 "Formal languages and automata: methods, models and applications".

[*] Corresponding author.
E-mail addresses: paolo.massazza@uninsubria.it (P. Massazza), radicion@dsi.unimi.it (R. Radicioni).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.08.011

and then we consider the formal series $\sum_{s\in S} w(s)\, s$. So, the counting problem associated with S and w consists of computing the function $f(n) = \sharp\{s \in S | w(s) = n\}$.

Another setting where the coefficient problem arises is the random generation of combinatorial structures (see, for instance, [7]). Efficient algorithms for the random generation of strings in a language can also be derived by considering the generating function associated with the language (see, for example, [5]).

We can also consider a more general version of the coefficient problem for multivariate formal series in commutative variables. More precisely, the coefficient problem for a class $\mathcal{A}$ of commutative formal series consists of computing, given a series in k variables $f = \sum_{\underline{n}\in\mathbb{N}^k} c_{\underline{n}} \underline{x}^{\underline{n}} \in \mathcal{A}$ and a multi-index $\underline{i} \in \mathbb{N}^k$, the coefficient $c_{\underline{i}}$ of f, $[\underline{x}^{\underline{i}}]f(\underline{x})$. When we deal with counting and random generation, this generalization appears whenever a multiple output weight function $w : S \to \mathbb{N}^k$ is considered. For example, multivariate formal series are used in counting or random sampling words with fixed occurrences of symbols [2] as well as in the random generation through object grammars [6].

In this paper we consider the coefficient problem for the class $\mathbb{Q}[[\{x, y\}]]_h$ of the holonomic formal series in two commuting variables. These are power series expansions of suitable holonomic functions defined by systems of linear differential equations with polynomial coefficients.

Given a couple of integers (i, j) and a holonomic system for a formal series $\phi(x, y)$,

$$\sum_{k=0}^{d_1} p_k(x, y)\partial_x^k \phi = 0, \qquad \sum_{k=0}^{d_2} q_k(x, y)\partial_y^k \phi = 0,$$

we show that the coefficient $[x^i y^j]\phi(x, y)$ can be computed in time $O(i + j)$ under the uniform cost criterion. Our method is based on the theory of holonomic functions and extends the result given in [10], where an efficient algorithm for computing the coefficient of a bivariate rational series is given. In order to compute $[x^i y^j]\phi(x, y)$ in an efficient way, we use suitable recurrence equations with polynomial coefficients associated with the holonomic system for $\phi(x, y)$. This is a remarkable improvement with respect to a more general algorithm presented in [9], where it is shown how to compute $[x^i y^j]\phi(x, y)$ in time $O(i \cdot j)$.

2. Preliminaries

We denote by $\mathbb{N}$ ($\mathbb{Q}$) the set of the natural (rational) numbers. A 2-dimensional sequence c with values in $\mathbb{Q}$ is a function $c : \mathbb{N}^2 \mapsto \mathbb{Q}$, usually denoted by $\{c_{nk}\}$. We denote by $\mathbb{Q}^{(2)}$ the ring of 2-dimensional sequences on $\mathbb{Q}$ with the operations of sum, $\{a_{nk}\} + \{b_{nk}\} = \{a_{nk} + b_{nk}\}$, and product, $\{a_{nk}\} \cdot \{b_{nk}\} = \{c_{nk}\}$ where $c_{nk} = \sum_{\substack{l+m=n\\ i+j=k}} a_{li} b_{mj}$.

Moreover, we consider the following operators from $\mathbb{Q}^{(2)}$ into $\mathbb{Q}^{(2)}$:

- External product by $e \in \mathbb{Q}$: $e \cdot \{a_{nk}\} = \{ea_{nk}\}$;
- (left) Shift : $E_n\{a_{nk}\} = \{a_{n-1\,k}\}$, $\quad E_k\{a_{nk}\} = \{a_{n\,k-1}\}$;
- Multiplication by n, k : $n\{a_{nk}\} = \{na_{nk}\}$, $\quad k\{a_{nk}\} = \{ka_{nk}\}$.

Then, the so called *shift algebra* $\mathbb{Q}\langle n, k, E_n, E_k\rangle$ is a particular Ore algebra (see, for instance, [4]) and can be interpreted as a (noncommutative) ring of linear operators on

$\mathbb{Q}^{(2)}$, with pseudo-commutative rules given by

$$nk = kn, \quad nE_k = E_k n, \quad kE_n = E_n k, \quad E_k E_n = E_n E_k,$$
$$nE_n = E_n n + E_n, \quad kE_k = E_k k + E_k.$$

Note that every normalized polynomial in $\mathbb{Q}\langle n, k, E_n, E_k\rangle$ corresponds to a linear recurrence with polynomial coefficients and vice versa. Particular sequences, called *P-recursive*, satisfy linear recurrences with polynomial coefficients and will be of interest in this paper.

A one-dimensional P-recursive sequence is defined as follows:

Definition 1. A sequence $\{a_n\}$ is called *P-recursive* if and only if there exists $P(n, E_n) \in \mathbb{Q}\langle n, E_n\rangle$ such that

$$P(n, E_n)\{a_n\} = \sum_{i=0}^{d} p_i(n) E_n^i \{a_n\} = 0.$$

The notion of P-recursiveness can be extended to n-dimensional sequences as shown in [9]. We recall here useful definitions for the two-dimensional case (see [9, Definitions 3.1, 3.2] for the general case).

Definition 2. Let $h \in \mathbb{N}$. An h-section of a sequence $\{c_{nk}\}$ is a one-dimensional subsequence that is obtained by holding one of the variables n or k fixed at values $< h$.

Then, we have:

Definition 3. A sequence $\{c_{nk}\}$ is called *P-recursive* if there exists $h \in \mathbb{N}$ such that
(1) for $i, j \in \{0, \ldots, h\}$ there are polynomials $p_{ij}(n)\, q_{ij}(k)$ such that

$$\sum_{i,j=0}^{h} p_{ij}(n) {E_n}^i {E_k}^j \{c_{nk}\} = P(n, E_n, E_k)\{c_{nk}\} = 0,$$
$$\sum_{i,j=0}^{h} q_{ij}(k) {E_n}^i {E_k}^j \{c_{nk}\} = Q(k, E_n, E_k)\{c_{nk}\} = 0,$$

where $p_{i_1 j_1}, q_{i_2 j_2} \neq 0$ for some integers $i_1, j_1, i_2, j_2 \in \{0, \ldots, h\}$;
(2) all the h-sections of $\{c_{nk}\}$ are P-recursive.

2.1. Formal series, holonomic functions and recurrences

Let X^{c} be the commutative free monoid generated by the alphabet $X = \{x, y\}$. A *commutative formal series* ϕ with values in $\mathbb{Q}$ and variables in X is a function $\phi : X^{\mathrm{c}} \mapsto \mathbb{Q}$, usually encoded by the sum $\phi(x, y) = \sum_{n,k \geqslant 0} c_{nk} x^n y^k$ where $c_{nk} = \phi(x^n y^k)$. We denote by $\mathbb{Q}[[X]]$ the ring of commutative formal series with coefficients in $\mathbb{Q}$ equipped with the usual operations of sum (+) and product (·). The *support* of ϕ is the set of monomials $\{x^n y^k \mid \phi(x^n y^k) \neq 0\}$ and $\mathbb{Q}[X]$ is the ring of polynomials, that is, formal series with finite support. The coefficient c_{nk} of a formal series ϕ will be often denoted by $[x^n y^k]\phi(x, y)$.

We are interested in a particular subclass of $\mathbb{Q}[[X]]$ that is the class $\mathbb{Q}[[X]]_h$ of the holonomic formal series in two commutative variables. This class can be formally defined as follows (see [9, Proposition 2.2]).

Definition 4. A series $\phi(x, y) \in \mathbb{Q}[[X]]$ is *holonomic* if and only if there exist some polynomials

$$p_{ij} \in \mathbb{Q}[X], \quad 1 \leqslant i \leqslant 2, \quad 0 \leqslant j \leqslant d_i, \quad p_{id_i} \neq 0$$

such that

$$\sum_{j=0}^{d_1} p_{1j} \partial_x^j \phi = 0, \quad \sum_{j=0}^{d_2} p_{2j} \partial_y^j \phi = 0.$$

The above equations are said to be a *holonomic system* for ϕ.

As a matter of fact, holonomic series are power series expansions of suitable functions that belong to the class of holonomic functions. This class was first introduced by I.N. Bernstein in the 1970s [1] and deeply investigated by Stanley, Lipshitz, Zeilberger and others (see, for instance, [3,8,9,11,12]). Note that there exist holonomic functions that are not holonomic formal series. For example, the function $f(x, y) = 1/(x + y)$ is holonomic since $(x + y)\partial_x f + f = 0$ and $(x + y)\partial_y f + f = 0$, while it is not a holonomic formal series since it has a pole in $(0, 0)$.

In our setting, we are interested in recurrence equations satisfied by sequences of coefficients of holonomic formal series. Therefore, we recall the following result [9, Theorem 3.7]:

Theorem 5. *Let $\phi(x, y) = \sum_{n,k \geqslant 0} c_{nk} x^n y^k$ be a holonomic series. Then the sequence $\{c_{nk}\}$ is P-recursive.*

By noting the following correspondence between operators,

$$x^r y^s \partial_x^i \partial_y^j \equiv \left(\prod_{h=1}^{i} (n - r + h) \prod_{h=1}^{j} (k - s + h) \right) E_n^{r-i} E_k^{s-j} \quad r > i, s > j, \tag{1}$$

we can easily associate with a holonomic system a system of recurrence equations. In fact, if we left-multiply a holonomic system by a suitable monomial $x^\alpha y^\beta$ and apply the correspondence above, we obtain the recurrences $P(n, E_n, E_k)$, $Q(k, E_n, E_k)$ given in Definition 3. Moreover, given a holonomic series, we can obtain a system of recurrences that is useful for computing most of the values belonging to h-sections of $\{c_{nk}\}$ (for arbitrary integers h).

Theorem 6. *Let $\phi(x, y) = \sum_{n,k \geqslant 0} c_{nk} x^n y^k$ be a holonomic series. Then the sequence of coefficients $\{c_{nk}\}$ satisfies a system of linear recurrence equations with polynomial coefficients*

$$P(n, k, E_n)\{c_{nk}\} = 0, \quad Q(n, k, E_k)\{c_{nk}\} = 0, \tag{2}$$

where $P(n, k, E_n) = \sum_{i=0}^{r} p_i(n, k) {E_n}^i$ and $Q(n, k, E_k) = \sum_{j=0}^{s} q_j(n, k) {E_k}^j$ belong to $\mathbb{Q}\langle n, k, E_n, E_k \rangle$ and $p_0, q_0 \neq 0$.

Proof. By [9, Lemma 2.4] we know that there exists a system

$$P(x, x\partial_x, y\partial_y)\phi = \textstyle\sum_{i,j}^{\hat{r}} \hat{p}_{ij}(x)(x\partial_x)^{(i)}(y\partial_y)^{(j)}\phi = 0,$$
$$Q(y, x\partial_x, y\partial_y)\phi = \textstyle\sum_{i,j}^{\hat{s}} \hat{q}_{ij}(y)(x\partial_x)^{(i)}(y\partial_y)^{(j)}\phi = 0,$$

satisfied by ϕ. According to Correspondence (1) we have $x \equiv E_n$, $y \equiv E_k$, $x\partial_x \equiv n$ and $y\partial_y \equiv k$. Then, the system associated with the sequence $\{c_{nk}\}$ is

$$P(n, k, E_n)\ \{c_{nk}\} = \textstyle\sum_{i,j}^{\hat{r}} \hat{p}_{ij}(E_n)n^i k^j\ \{c_{nk}\} = 0,$$
$$Q(n, k, E_k)\ \{c_{nk}\} = \textstyle\sum_{i,j}^{\hat{s}} \hat{q}_{ij}(E_k)n^i k^j\ \{c_{nk}\} = 0.$$

At last, by using the pseudo-commutative rules of $\mathbb{Q}\langle n, k, E_n, E_k\rangle$, we obtain the system

$$P(n, k, E_n)\{c_{nk}\} = \textstyle\sum_{i=0}^{r} p_i(n, k){E_n}^i\{c_{nk}\} = 0,$$
$$Q(n, k, E_k)\{c_{nk}\} = \textstyle\sum_{j=0}^{s} q_j(n, k){E_k}^j\{c_{nk}\} = 0,$$

for suitable polynomials p_i, q_j and suitable integers r, s. □

We point out that the computation of a system of type (2) starting from a holonomic system requires the solution of an elimination problem in $\mathbb{Q}\langle n, k, E_n, E_k\rangle$. As shown in [12, Sections 4.1, 4.2], it is first necessary to consider a system with appropriate initial conditions (some more equations might be added to the system from the annihilation ideal of ϕ or $\{c_{nk}\}$). Then, an elimination method is applied to the system. In [12, Section 5] Sylvester's classical dialytic elimination method is adapted to $\mathbb{Q}\langle n, k, E_n, E_k\rangle$. A different elimination method is based on the computation of the Gröbner basis of a set of polynomials in $\mathbb{Q}\langle n, k, E_n, E_k\rangle$ (with suitable initial conditions, as indicated before) that we can obtain by mapping the holonomic system into $\mathbb{Q}\langle n, k, E_n, E_k\rangle$. A useful package for such computation, for example, is given by [4].

3. Computing the coefficient

Given a holonomic system for a series $\phi \in \mathbb{Q}[[X]]_h$, we can easily obtain two linear recurrence equations with univariate coefficients, $P(n, E_n, E_k)$ and $Q(k, E_n, E_k)$, satisfied by $\{[x^n y^k]\phi(x, y)\}$. Then, we can use them for computing an arbitrary coefficient $[x^i y^j]\phi(x, y)$ once a suitable set of initial conditions is known. On the other hand, because both E_n and E_k might appear in P and Q, this technique requires in general $O(ij)$ coefficients in order to determine $[x^i y^j]\phi(x, y)$.

As shown before, the theory of holonomic systems allows us to obtain particular linear recurrence equations with polynomial coefficients that are more suitable for computing coefficients. More precisely, Theorem 6 provides us with two operators of the shift algebra $\mathbb{Q}\langle n, k, E_n, E_k\rangle$ that depend on n, k and either E_n or E_k. So, we can efficiently compute all the coefficients along a line $n = \overline{n}$ or $k = \overline{k}$ if the leading and the least coefficients of such recurrences do not vanish on that line.

Our approach takes advantage of both types of recurrences (those in Definition 3 and Theorem 6) in order to get a method that efficiently computes the coefficient $[x^i y^j]\phi(x, y)$ by starting with a suitable set *I* of initial conditions and proceeding by choosing at each step the "right" recurrence to use. So, we formally define the following problem for holonomic series in two variables:

Problem: The coefficient problem for $\mathbb{Q}[[X]]_h$

Input: A tuple $\langle B_n, B_k, \mathrm{Sec}, U_n, U_k, I, i, j\rangle$ where:

- B_n, B_k are polynomials in $\mathbb{Q}\langle n, k, E_n, E_k\rangle$,

$$B_n(n, E_n, E_k) = \sum_{i,j=0}^{h} p_{ij}(n){E_n}^i {E_k}^j,$$
$$B_k(k, E_n, E_k) = \sum_{i,j=0}^{h} q_{ij}(k){E_n}^i {E_k}^j,$$

  that identify two recurrences satisfied by the sequence $\{[x^n y^k]\phi(x, y)\}$ of $\phi(x, y) \in \mathbb{Q}[[X]]_h$ (see Definition 3).
- $\mathrm{Sec} = \{T_i(n, E_n), V_i(k, E_k) \mid 0 \leqslant i < h\}$ is a family of $2h$ univariate recurrences for the h-sections of $\{[x^n y^k]\phi(x, y)\}$ with

$$T_i(n, E_n)\{[x^n y^i]\phi(x, y)\} = \sum_{j=0}^{d_i} t_{ij}(n)E_n^j\{[x^n y^i]\phi(x, y)\} = 0,$$
$$V_i(k, E_k)\{[x^i y^k]\phi(x, y)\} = \sum_{j=0}^{f_i} r_{ij}(k)E_k^j\{[x^i y^k]\phi(x, y)\} = 0.$$

- U_n, U_k are two polynomials in $\mathbb{Q}\langle n, k, E_n, E_k\rangle$,

$$U_n(n, k, E_n) = \sum_{i=0}^{r} a_i(n, k){E_n}^i \quad a_r(n, k) \neq 0,$$
$$U_k(n, k, E_k) = \sum_{j=0}^{s} b_j(n, k){E_k}^j \quad b_s(n, k) \neq 0,$$

  that identify two recurrences satisfied by the sequence $\{[x^n y^k]\phi(x, y)\}$ (see Theorem 6).
- *I* is a finite set of initial conditions, $I = \{[x^n y^k]\phi(x, y) \mid n, k \in \{0, \ldots, d\}\}$, where:
  - $\alpha = \alpha(B_n) = \max\{n \in \mathbb{N} \mid \exists i, j \in \{0, \ldots, h\},\ p_{ij}(n) = 0\}$;
  - $\beta = \beta(B_k) = \max\{n \in \mathbb{N} \mid \exists i, j \in \{0, \ldots, h\},\ q_{ij}(n) = 0\}$;
  - $\gamma = \gamma(\mathrm{Sec}) = \max\{n \in \mathbb{N} \mid \exists i, j, (t_{ij} \neq 0 \wedge t_{ij}(n) = 0) \vee (r_{ij} \neq 0 \wedge r_{ij}(n) = 0)\}$;
  - $d = \max\{r, s, h, \alpha, \beta, \gamma, f_0, f_1, \ldots, f_{h-1}, d_0, d_1, \ldots, d_{h-1}\}$.
- $(i, j) \in \mathbb{N}^2$.

Output: The coefficient $[x^i y^j]\phi(x, y)$.

Note that several recurrences appears in the previous definition. Therefore, for sake of clarity, we show in Fig. 1 where the different recurrences are used by the algorithm we describe later.

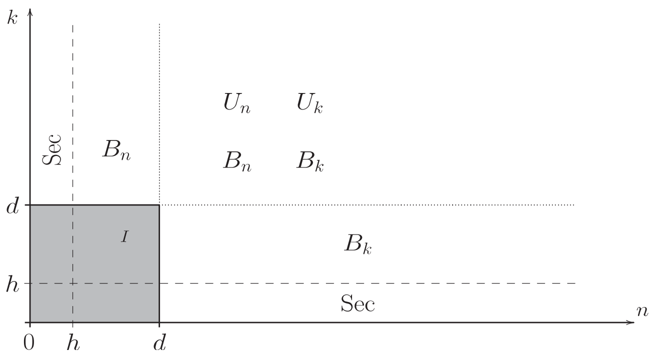


Fig. 1. Initial conditions and use of recurrences.

3.1. Clusters and coefficients

Informally, our algorithm works by finding a sequence S of $O(i + j)$ points from $(0, 0)$ to (i, j) such that for each (a, b) in S the coefficient $[x^a y^b]\phi(x, y)$ can be computed by at least one of the known recurrences applied to suitable values $[x^u y^v]\phi(x, y)$ with (u, v) preceding (a, b) in S.

In order to do that, we consider a tiling of $\mathbb{N}^2$ with suitable defined parallelograms, together with a set of rules that let us to compute the coefficients associated with the points within a parallelogram from the knowledge of the coefficients associated with parallelograms in the neighbourhood. Hence, we first introduce the definition of parallelogram and some related notations.

Definition 7 ($P^{(ab)}(x, y)$). Let $a, b \in \mathbb{N}$. Then, for each $(x, y) \in \mathbb{N}^2$ such that $x \geqslant (b+1)a$ and $y \geqslant b$, the *parallelogram* $P^{(ab)}(x, y)$ is the set of points

$$\begin{aligned} P^{(ab)}(x, y) = \{(x', y') \in \mathbb{N}^2 \mid y - b \leqslant y' \leqslant y \\ \wedge\ x - (b + 1)a + (y - y')a \leqslant x' \leqslant x + (y - y')a\}. \end{aligned}$$

Definition 8 ($P^{(ab)}_{\text{Start}}$). The starting parallelogram $P^{(ab)}_{\text{Start}}$ is the parallelogram $P^{(ab)} \times ((b + 1)a, b)$.

We consider a tiling of $\mathbb{N}^2$ with parallelograms that are obtained by moving at each step the starting parallelogram $P^{(ab)}_{\text{Start}}$ (see Fig. 2) towards *N*orth or *E*ast according to the following partial functions (see, for instance, Fig. 3):

$$\begin{aligned} &N(P^{(ab)}(x, y)) = P^{(ab)}(x - ba, y + b) \quad (\text{defined if } x \geqslant (2b + 1)a), \\ &E(P^{(ab)}(x, y)) = P^{(ab)}(x + (b + 1)a, y). \end{aligned}$$

Note that $\sharp(P^{(ab)}(x, y) \cap N(P^{(ab)}(x, y))) = (b + 1)a + 1$ and $\sharp(P^{(ab)}(x, y) \cap E(P^{(ab)}(x, y))) = b + 1$. More formally, we are interested in the set of parallelograms defined as follows:

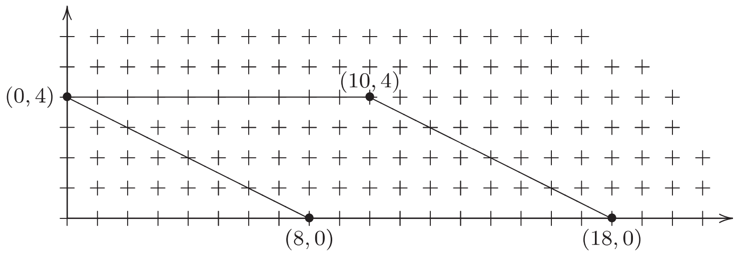


Fig. 2. The starting parallelogram $P^{(2\,4)}_{\text{Start}}$.

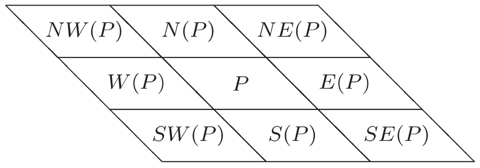


Fig. 3. P and its neighbours.

Definition 9 (PAR$^{(ab)}$). Let $a, b \in \mathbb{N}$. The set of parallelograms generated by $P^{(ab)}_{\text{Start}}$ is

$$\text{PAR}^{(ab)} = \left\{ P^{(ab)}(x, y) \mid \exists (u, v) \in \mathbb{N}^2, P^{(ab)}(x, y) = N^u(E^v(P^{(ab)}_{\text{Start}})) \right\}.$$

Note that all the parallelograms in PAR$^{(ab)}$ lie in the first quadrant.

Actually, the algorithm deals with sequences of elements in PAR$^{(ab)}$ that are obtained by moving at each step a parallelogram towards *N*orth, *E*ast, *S*outh or *W*est, so we define also the functions

$$\begin{aligned} S(P^{(ab)}(x, y)) &= P^{(ab)}(x + ba, y - b) \quad (\text{defined if } y \geqslant 2b), \\ W(P^{(ab)}(x, y)) &= P^{(ab)}(x - (b+1)a, y) \quad (\text{defined if } x \geqslant 2(b+1)a). \end{aligned}$$

Given a direction $T \in \{N, E, S, W\}$ and $P \in \text{PAR}^{(ab)}$, we write $T^i(P)$ for $T(T^{i-1}(P))$, $T^0(P) = P$. Moreover, we also consider the shortcuts $SW(P) = W(S(P))$, $NW(P) = N(W(P))$, $SE(P) = S(E(P))$ and $NE(P) = N(E(P))$.

The neighbours of a parallelogram are defined through the following reflexive and symmetric relations on PAR$^{(ab)}$.

Definition 10 ($\Diamond$,$\Box$). Given $P_1^{(ab)}$, $P_2^{(ab)} \in \text{PAR}^{(ab)}$, then

- $P_1^{(ab)} \Diamond P_2^{(ab)} \Leftrightarrow \exists T \in \{N, NE, E, SE, S, SW, W, NW\}, \quad P_1^{(ab)} = T(P_2^{(ab)})$,
- $P_1^{(ab)} \Box P_2^{(ab)} \Leftrightarrow \exists T \in \{N, E, S, W\}, \ P_1^{(ab)} = T(P_2^{(ab)})$.

We say that P_1 is 8-*adjacent* (4-*adjacent*) to P_2 if and only if $P_1 \Diamond P_2$ ($P_1 \Box P_2$). Particular sequences of parallelograms will be of interest when considering the behaviour of the algorithm.

Definition 11. Let Seq $= P_1, \ldots, P_k$ be a sequence of parallelograms in $\mathrm{PAR}^{(ab)}$. Then, Seq is

- 8-*connected* iff $P_i \Diamond P_{i+1}$, $0 \leqslant i < k$;
- 4-*connected* iff $P_i \Box P_{i+1}$, $0 \leqslant i < k$;
- *descending* iff Seq is 8-connected or 4-connected and $P_{i+1} = T_i(P_i)$ with $T_i \in \{E, SE, S, SW, W\}$, $0 \leqslant i < k$;
- *ascending* iff Seq is 8-connected or 4-connected and $P_{i+1} = T_i(P_i)$ with $T_i \in \{W, NW, N, NE, E\}$, $0 \leqslant i < k$.

Henceforward, we fix an instance $\langle B_n, B_k, \mathrm{Sec}, U_n, U_k, I, i, j\rangle$ of the coefficient problem for a series $\phi(x, y) \in \mathbb{Q}[[X]]_h$ and we associate with it the following values:

- $Z = Z(U_n, U_k) = \{(x, y) \in \mathbb{N}^2 \mid a_r(x, y) = 0 \vee a_0(x, y) = 0 \vee b_s(x, y) = 0 \vee b_0(x, y) = 0\}$;
- $a = a(U_n, B_n, B_k) = \max\{r, h\}$;
- $b = b(U_k, B_n, B_k) = \max\{s, h + 1\}$;
- $P_0 = P^{(ab)}_{\mathrm{Start}}$;
- $\overline{P} \in \mathrm{PAR}^{(ab)}$ univocally identified by the integers

$$c_1 = \lceil (i + (j - b)a)/((b + 1)a)\rceil - 1, \quad c_2 = \lceil j/b\rceil - 1$$

such that $(i, j) \in \overline{P} = \overline{P}^{(ab)}(\bar{\imath}, \bar{\jmath}) = N^{c_2}(E^{c_1}(P_0))$.

Given an instance, we will write P and PAR instead of $P^{(ab)}$ and $\mathrm{PAR}^{(ab)}$ whenever the context is clear.

Definition 12 ($\mathrm{PAR}(P), \mathrm{PAR}_{\mathtt{V}}(P)$). Let $P = N^c(E^d(P_0)) \in \mathrm{PAR}$ $(c, d \in \mathbb{N})$. Given $V \subseteq \mathbb{N}^2$ we have

$$\mathrm{PAR}(P) = \Big\{Q \in \mathrm{PAR} \mid \exists u, v \in \mathbb{N}, u \leqslant d, v \leqslant c,$$
$$Q = W^u(S^v(P)) = N^{c-v}(E^{d-u}(P_0))\Big\},$$
$$\mathrm{PAR}_V(P) = \{Q \in \mathrm{PAR}(P) \mid Q \cap V \neq \emptyset\}.$$

Note that $\mathrm{PAR}_Z(\overline{P})$ consists of those parallelograms in $\mathrm{PAR}(\overline{P})$ that contain at least one point (n, k) such that at least one of the following methods fails:

Compute$_{\mathtt{N}}(n, k)$: use U_n to compute $[x^n y^k]\phi$ from the values $[x^{n-l} y^k]\phi$ or $[x^{n+l} y^k]\phi$, $1 \leqslant l \leqslant r$;

Compute$_{\mathtt{K}}(n, k)$: use U_k to compute $[x^n y^k]\phi$ from the values $[x^n y^{k-l}]\phi$ or $[x^n y^{k+l}]\phi$, $1 \leqslant l \leqslant s$.

The elements of $\mathrm{PAR}_Z(\overline{P})$ can be considered as "obstacles", in the sense that their associated coefficients can be computed only by using B_n or B_k. Henceforward, we call them *singular parallelograms*.

The size of $\mathrm{PAR}_Z(\overline{P})$ will be of interest in the analysis of the algorithm we propose, in a way that will become clear later. So we state the following:

Lemma 13. $\sharp PAR_Z(\overline{P}) = \mathrm{O}(\bar{\imath} + \bar{\jmath})$.

Proof. We first note that if $(x, y) \in Q$ with $Q \in \mathrm{PAR}_Z(\overline{P})$ then $0 \leqslant y \leqslant \bar{\jmath}$. Note that if $(k - y)$ is not a factor of $a_0(n, k)$, $a_r(n, k)$ (in U_n) or $b_0(n, k)$, $b_s(n, k)$ (in U_k), then the univariate polynomials $a_0(n, y)$, $a_r(n, y)$, $b_0(n, y)$ and $b_s(n, y)$ admit at most

$$D_{\max_n} = \max\{\deg_n(a_0(n, k)), \deg_n(a_r(n, k)), \deg_n(b_0(n, k)), \deg_n(b_s(n, k))\}$$

zeroes. Let $Z_{\bar{\imath}\bar{\jmath}} = \{(x, y) \in Z \mid 0 \leqslant x \leqslant \bar{\imath}, 0 \leqslant y \leqslant \bar{\jmath}\}$. Since the number of factors of type $(k - y)$ and the associated multiplicities are bounded by

$$D_{\max_k} = \max\{\deg_k(a_0(n, k)), \deg_k(a_r(n, k)), \deg_k(b_0(n, k)), \deg_k(b_s(n, k))\},$$

we have

$$\sharp Z_{\bar{\imath}\bar{\jmath}} \leqslant 4D_{\max_n}(\bar{\jmath} + 1) + 4D_{\max_k}(\bar{\imath} + a\bar{\jmath}) = \mathrm{O}(\bar{\imath} + \bar{\jmath}).$$

Since each parallelogram in $\mathrm{PAR}_Z(\overline{P})$ contains at least one point in $Z_{\bar{\imath}\bar{\jmath}}$ and each point in $Z_{\bar{\imath}\bar{\jmath}}$ belongs to at most four singular parallelograms, we have $\sharp\mathrm{PAR}_Z(\overline{P}) \leqslant 4\sharp Z_{\bar{\imath}\bar{\jmath}}$ and so $\sharp\mathrm{PAR}_Z(\overline{P}) = \mathrm{O}(\bar{\imath} + \bar{\jmath})$. □

We indicate by $\mathrm{Coeff}_\phi(P)$ the family of coefficients of ϕ associated with $P \in \mathrm{PAR}$, that is,

$$\mathrm{Coeff}_\phi(P) = \{[x^n y^k]\phi(x, y) \mid (n, k) \in P\}.$$

The following lemmas show how to compute the coefficients in $\mathrm{Coeff}_\phi(P)$ from the knowledge of the coefficients associated with neighbours of P.

Lemma 14. *Let P be a nonsingular parallelogram. If there exists $T \in \{N, W, S, E\}$ such that* $\mathrm{Coeff}_\phi(T(P))$ *is known, then* $\mathrm{Coeff}_\phi(P)$ *can be computed in time* $\mathrm{O}(1)$.

Proof. Let $P = P(l, m)$. We consider two cases:

- ($T \in \{E, W\}$). Without loss of generality let us consider $T = E$, that is, suppose that $\mathrm{Coeff}_\phi(E(P)) = \mathrm{Coeff}_\phi(P(l + (b + 1)a, m))$ is known. It is immediate to obtain $\mathrm{Coeff}_\phi(P(l + (b + 1)a - 1, m))$ by computing $b + 1$ coefficients in $\mathrm{Coeff}_\phi(P(l + (b + 1)a - 1, m)) \setminus \mathrm{Coeff}_\phi(P(l + (b + 1)a, m))$: this is done by using U_n and the values of the $b + 1$ "rows" of $\mathrm{Coeff}_\phi(E(P))$. We obtain $\mathrm{Coeff}_\phi(P(l, m))$ by iterating this computation $(b + 1)a$ times.
- ($T \in \{N, S\}$). Without loss of generality we suppose that $\mathrm{Coeff}_\phi(N(P)) = \mathrm{Coeff}_\phi(P(l - ba, m + b))$ is known. We obtain the coefficients in $\mathrm{Coeff}_\phi(P(l - (b - 1)a, m + b - 1))$ in two steps: first, we apply U_k to the "columns" $l - ba + 1, \ldots, l - ba + a$ of $\mathrm{Coeff}_\phi(P(l - ba, m + b))$ in order to get the leftmost a coefficients in $\mathrm{Coeff}_\phi(P(l - (b - 1)a, m + b - 1)) \setminus \mathrm{Coeff}_\phi(P(l - ba, m + b))$. Second, from these a values we compute the remaining ba values on the right by applying U_n ba times. We obtain $\mathrm{Coeff}_\phi(P(l, m))$ by iterating this computation $b + 1$ times.

Therefore, $\mathrm{Coeff}_\phi(P(l,m))$ is computed in time O(1) (independent of l or m) since the computation of one coefficient requires $\max(a,b)$ arithmetical operations and we compute $((b+1)a+1)\cdot(b+1)$ coefficients (a,b constants). □

The next lemma refers to the integer constant d that appears in the formal definition of the coefficient problem for a series ϕ (see the definition of I in the tuple $\langle B_n, B_k, \mathrm{Sec}, U_n, U_k, I, i, j\rangle$).

Lemma 15. *Let P be a singular parallelogram such that if $(a,b)\in P$ then $a>d$ and $b>d$. If* $\mathrm{Coeff}_\phi(W(P))$, $\mathrm{Coeff}_\phi(SW(P))$ *and* $\mathrm{Coeff}_\phi(S(P))$ *are known, then* $\mathrm{Coeff}_\phi(P)$ *can be computed in time* O(1).

Proof. We define an ordering $\prec$ on P as follows: $(\alpha,\beta)\prec(\gamma,\delta)$ iff $\beta<\delta$ or $\beta=\delta$ and $\alpha\leqslant\gamma$. Then, we compute the coefficients in $\mathrm{Coeff}_\phi(P)$ according to the $\prec$, starting with $[x^{\alpha_0}y^{\beta_0}]\phi(x,y)$ ($(\alpha_0,\beta_0)=\min(P)$) and using B_n or B_k. At each step we compute one coefficient using at most ab arithmetical operations. Since the number of coefficients is $(b+1)\cdot(a(b+1)+1)$, $\mathrm{Coeff}_\phi(P)$ can be computed in time $\mathrm{O}((b+1)\cdot(a(b+1)+1)\cdot ab)=\mathrm{O}(1)$ (independent of l or m). □

The transitive closure $\diamond^\star$ defines an equivalence relation on $\mathrm{PAR}_Z(\overline{P})$, i.e. it defines a partition of $\mathrm{PAR}_Z(\overline{P})$ into equivalence classes that we call *clusters*. Informally, a cluster can be seen as a group of singular parallelograms that form a connected figure. More precisely, let $\diamond_{\overline{P}}=\diamond\cap\mathrm{PAR}_Z(\overline{P})\times\mathrm{PAR}_Z(\overline{P})$ and consider the following:

Definition 16 (Cluster). The cluster generated by $P\in\mathrm{PAR}_Z(\overline{P})$ is

$$Cl_P^{\overline{P}}=[P]_{\diamond^\star_{\overline{P}}}=\left\{Q\in\mathrm{PAR}_Z(\overline{P})\mid Q\diamond^\star_{\overline{P}}P\right\}.$$

It is immediate to observe that we have the partition

$$\mathrm{PAR}_Z(\overline{P})=\biguplus_{h=1}^{k}\ \mathrm{Cl}_{P_h}^{\overline{P}}$$

with $P_h\in\mathrm{PAR}_Z(\overline{P})$ and $P_{h_1}\not\diamond^\star_{\overline{P}}P_{h_2}$ if $h_1\neq h_2$.

Example 17. Let us consider the function $\phi(x,y)=(x+y)e^{y(x+y)}$ and the recurrences

$$B_n=-E_nE_k^2-E_n^2E_k-2E_n+nE_k+nE_n,$$
$$B_k=-2E_k-E_n^2E_k-2E_k^3-3E_nE_k^2+kE_k+kE_n,$$

$$U_n=(14n-45+4k+k^2-n^2)En^2+40n^2-84k-2n^2k+26nk-266n$$
$$+588-2n^3,$$

$$U_k=(2n+2k-2)Ek^2-2n+4k-3+n^2-k^2,$$

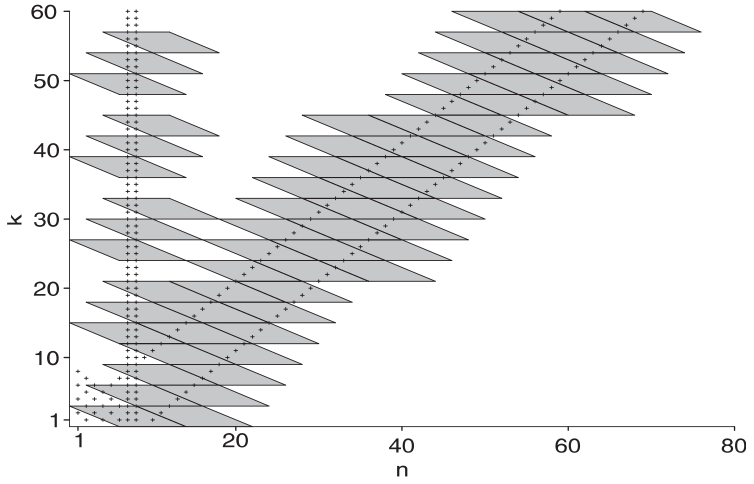


Fig. 4. The cluster $\mathrm{Cl}_{P^{(2\ 3)}(8,3)}^{P^{(2\ 3)}(70,60)}$ associated with $(x+y)e^{y(x+y)}$. Elements of $Z(U_n, U_k)$ are indicated by crosses.

associated with it. Let $\overline{P} = P^{(2\ 3)}(70, 60)$ and $P = P^{(2\ 3)}(8, 3)$. The graphical representation of the cluster $Cl_P^{\overline{P}}$ is given in Fig. 4.

Given a cluster $\mathrm{Cl}_Q^{\overline{P}}$ we define its *border* as the set $B(Cl_Q^{\overline{P}})$ of nonsingular parallelograms that are 8-adjacent to parallelograms in $\mathrm{Cl}_Q^{\overline{P}}$,

$$B(Cl_Q^{\overline{P}}) = \{P \in \mathrm{PAR}(\overline{P}) \setminus \mathrm{PAR}_Z(\overline{P}) \mid \exists P' \in Cl_Q^{\overline{P}} \text{ s.t. } P \diamond P'\}.$$

It is immediate to verify that $\sharp B(Cl_Q^{\overline{P}}) = \mathrm{O}(\sharp Cl_Q^{\overline{P}}) = \mathrm{O}(\bar{\imath} + \bar{\jmath})$.

3.2. The algorithm

As shown in the previous section, an instance $\langle B_n, B_k, \mathrm{Sec}, U_n, U_k, I, i, j\rangle$ univocally identifies a set Z, two integers a, b and two parallelograms P_0, $\overline{P}$. So, we compute the coefficient $[x^i y^j]\phi(x, y)$ through a procedure that starts with the computation of $\mathrm{Coeff}_\phi(P_0)$ and halts having computed the family $\mathrm{Coeff}_\phi(\overline{P})$, with $[x^i y^j]\phi(x, y) \in \mathrm{Coeff}_\phi(\overline{P})$, after $\mathrm{O}(i + j)$ steps.

The basic idea is quite simple: we would like to compute the sequence of families $\mathrm{Coeff}_\phi(P_0)$, $\mathrm{Coeff}_\phi(E(P_0)), \ldots, \mathrm{Coeff}_\phi(E^{c_1}(P_0))$, $\mathrm{Coeff}_\phi(N(E^{c_1}(P_0))), \ldots, \mathrm{Coeff}_\phi(N^{c_2}(E^{c_1}(P_0)))$ by going first eastwards and then northwards, until $\mathrm{Coeff}_\phi(N^{c_2}(E^{c_1}(P_0))) = \mathrm{Coeff}_\phi(\overline{P})$ is computed.

If no singular parallelograms are found, the pth family is easily computed by the $(p-1)$th family using Lemma 14. Otherwise, we eventually go through a singular parallelogram P_p and then the only knowledge of $\mathrm{Coeff}_\phi(P_{p-1})$, is not sufficient to proceed. To deal with this situation, we go round P_p (clockwise), starting from P_{p-1} and computing

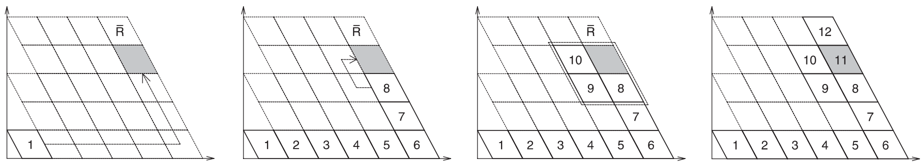


Fig. 5. An example of computation. Parallelograms are numbered with respect to the order of computation. The grey parallelogram is singular.

the associated families of coefficients until $\mathrm{Coeff}_\phi(S(P_p))$, $\mathrm{Coeff}_\phi(SW(P_p))$ and $\mathrm{Coeff}_\phi \times (W(P_p))$ are known. Then we can apply Lemma 15 and compute $\mathrm{Coeff}_\phi(P_p)$. If another singular parallelogram is found while going round P_p we recursively compute the associated family in the same way. Fig. 5 illustrates a sequence of families of coefficients that are computed according to this method. More precisely, we design a procedure that works in two steps. The first step has to be considered as a sort of precomputation: it computes $\mathrm{O}(i+j)$ suitable coefficients in time $\mathrm{O}(i+j)$. Such coefficients act as halting condition for the recursive computations that might occur in the second step. The core of the algorithm is indeed the second step.

Step 1: Compute all the coefficients $[x^n y^k]\phi(x, y)$ with either $n \leqslant \max(d, 2(b+1)a + ba + 1)$ and $d < k \leqslant \bar{\jmath}$ or $d < n \leqslant \bar{\imath} + ba(c_2 + 1)$ and $k \leqslant d$.

Step 2: For $0 < j \leqslant c_2$, compute the family $\mathrm{Coeff}_\phi(N^j(E^{c_1}(P_0)))$ having as input $\mathrm{Coeff}_\phi(N^{j-1}(E^{c_1}(P_0)))$ according to the following rule: if $N^j(E^{c_1}(P_0))$ is nonsingular then compute $\mathrm{Coeff}_\phi(N^j(E^{c_1}(P_0)))$ as shown in Lemma 14, otherwise apply Lemma 15 and compute all the coefficients associated with the cluster $\mathrm{Cl}^{N^j(E^{c_1}(P_0))}_{N^j(E^{c_1}(P_0))}$ (in a suitable order that derives from the recursive method illustrated before).

We give here an outline of how to compute the coefficients involved in Step 1. First, we compute the coefficients $[x^n y^k]\phi(x, y)$ with $d < n \leqslant \bar{\imath} + bac_2$ and $0 \leqslant k < h$ by using $T_k(n, E_n)$ in Sec and the initial conditions I. Then, we compute the coefficients $[x^n y^k]\phi(x, y)$, with $d < n \leqslant \bar{\imath} + bac_2$ and $h \leqslant k \leqslant d$, according to the following order: $[x^\alpha y^\beta]\phi(x, y)$ is computed before $[x^\gamma y^\delta]\phi(x, y)$ iff $\beta < \delta$ or $\beta = \delta$ and $\alpha < \gamma$. This is done by using B_n and I.

We then proceed by computing the coefficients $[x^n y^k]\phi(x, y)$ with $0 \leqslant n < h$ and $d < k \leqslant \bar{\jmath}$ (we use the recurrence $V_n(k, E_k)$ in Sec and the initial conditions I). At last, the coefficients $[x^n y^k]\phi(x, y)$, with $h \leqslant n \leqslant \max(d, 2(b+1)a + ba + 1))$ and $d < k \leqslant \bar{\jmath}$, are computed by using B_k and I in a way such that $[x^\alpha y^\beta]\phi(x, y)$ is computed before $[x^\gamma y^\delta]\phi(x, y)$ iff $\alpha < \gamma$ or $\alpha = \gamma$ and $\beta < \delta$.

Fig. 6 shows where the different recurrences are used.

Note that after Step 1 we know all the families $\mathrm{Coeff}_\phi(P(l, m))$ with either $(b+1)a \leqslant l \leqslant \bar{\imath} + bac_2$ and $b \leqslant m \leqslant d$ or $(b+1)a \leqslant l \leqslant \max(d - ba, 2(b+1)a + 1)$ and $b \leqslant m \leqslant \bar{\jmath}$. In particular, this means that the sequence

$$\mathrm{Coeff}_\phi(P_0), \mathrm{Coeff}_\phi(E(P_0)), \ldots, \mathrm{Coeff}_\phi(E^{c_1}(P_0))$$

has been computed.

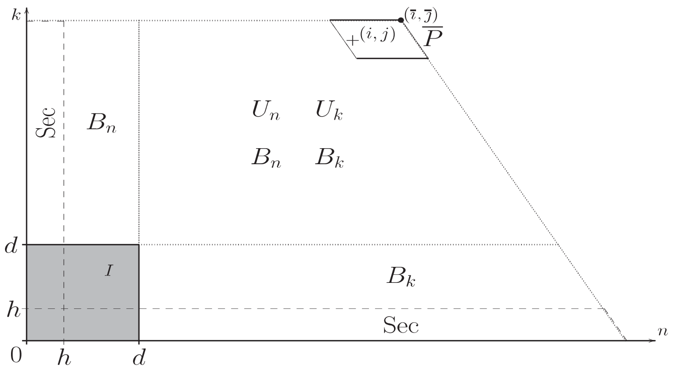


Fig. 6. Recurrences, initial conditions, and the target parallelogram $\overline{P}$.

```
**Procedure** COEFF($i, j$)
**Begin**
  $P_0 := P^{(ab)}_{\text{Start}}$;
  $c_2 := \lceil j/b \rceil - 1$;
  $c_1 := \lceil (i + (j - b)a)/((b + 1)a) \rceil - 1$;      /* $(i, j) \in N^{c_2}(E^{c_1}(P_0))$ */
  PRECOMPUTE(Sec,$B_n$,$B_k$,$I$,$i$,$j$);                    /* Step 1 */
  **For** c **from** 1 **to** $c_2$ **do**                          /* Step 2 */
    **if** $N^c(E^{c_1}(P_0)) \notin \mathrm{PAR}_Z(\overline{P})$
      **then** compute $\mathrm{Coeff}_\phi[N^c(E^{c_1}(P_0))]$ by using Lemma 14 and
           $\mathrm{Coeff}_\phi[N^{c-1}(E^{c_1}(P_0))]$;
      **else** COMPUTE$\circlearrowright$($N^c(E^{c_1}(P_0)), N^{c-1}(E^{c_1}(P_0))$);
  **return** $[x^i y^j]\phi(x, y)$ from $\mathrm{Coeff}_\phi[N^{c_2}(E^{c_1}(P_0))]$;
**End**;
```

Fig. 7. Procedure COEFF.

In Fig. 7 we define a procedure COEFF(i, j) that takes two positive integers i, j in input and returns the value $[x^i y^j]\phi(x, y)$. In the code, two procedures COMPUTE$\circlearrowright$ and PRECOMPUTE are called. All the procedures use a suitable data structure $\mathrm{Coeff}_\phi[]$ for the families $\mathrm{Coeff}_\phi(P)$. Moreover, we suppose that all the items given in the instance of the problem are available as global variables, as well as those values univocally associated with the instance (e.g. the integers a and b, defining the size of the parallelogram, independent of i or j).

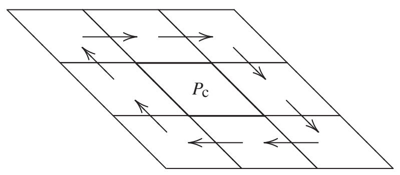
Procedure PRECOMPUTE(Sec, B_n, B_k, I, i, j) corresponds to Step 1. Procedure COMPUTE$\circlearrowright$(P_{out}, P_{in}) takes two parallelograms P_{out}, P_{in} such that $P_{\text{out}} \square P_{\text{in}}$ and computes $\mathrm{Coeff}_\phi(P_{\text{out}})$ under the assumption that $\mathrm{Coeff}_\phi(P_{\text{in}})$ has been computed. It moves clockwise and uses the coefficients previously computed. In the code we find an indexed

```
Procedure COMPUTE$\circlearrowright$($P_{\mathrm{out}}$, $P_{\mathrm{in}}$)
Begin
  While $\mathrm{Coeff}_\phi[S(P_{\mathrm{out}})]$, $\mathrm{Coeff}_\phi[SW(P_{\mathrm{out}})]$, $\mathrm{Coeff}_\phi[W(P_{\mathrm{out}})]$
        are not known do
    $P' := next_{P_{\mathrm{out}}}(P_{\mathrm{in}})$;
    if $\mathrm{Coeff}_\phi[P']$ is not known then
      if $P' \notin \mathrm{PAR}_Z(\overline{P})$ then compute $\mathrm{Coeff}_\phi[P']$ by using Lemma 14 and
                                    $\mathrm{Coeff}_\phi[P_{\mathrm{in}}]$;
                                else COMPUTE$\circlearrowright$($P'$, $P_{\mathrm{in}}$);
    $P_{\mathrm{in}} := P'$;
  EndWhile
  compute $\mathrm{Coeff}_\phi[P_{\mathrm{out}}]$ by using Lemma 15 and
  $\mathrm{Coeff}_\phi[S(P_{\mathrm{out}})]$, $\mathrm{Coeff}_\phi[SW(P_{\mathrm{out}})]$, $\mathrm{Coeff}_\phi[W(P_{\mathrm{out}})]$;
End;
```

Fig. 8. Procedure COMPUTE$\circlearrowright$.

function $\mathrm{next}_{P_c}(P)$: this is used to identify the parallelogram P' that is 8-adjacent to P_c and follows P (clockwise). More formally:

$$\mathrm{next}_{P_c}(P) = \begin{cases} NE(P_c) & \text{if} \quad P = N(P_c) \\ E(P_c) & \text{if} \quad P = NE(P_c) \\ SE(P_c) & \text{if} \quad P = E(P_c) \\ S(P_c) & \text{if} \quad P = SE(P_c) \\ SW(P_c) & \text{if} \quad P = S(P_c) \\ W(P_c) & \text{if} \quad P = SW(P_c) \\ NW(P_c) & \text{if} \quad P = W(P_c) \\ N(P_c) & \text{if} \quad P = NW(P_c) \end{cases}$$



Function next

Let us consider, for example, the call COMPUTE$\circlearrowright$(P, $S(P)$). This call knows the value $\mathrm{Coeff}_\phi[S(P)]$ and assigns a value to $\mathrm{Coeff}_\phi[P]$. So, if P is singular then $\mathrm{Coeff}_\phi[SW(P)]$ and $\mathrm{Coeff}_\phi[W(P)]$ are needed, as shown in Lemma 15. Hence, the procedure advances clockwise around P, in order to compute (recursively) $\mathrm{Coeff}_\phi[SW(P)]$ (with $\mathrm{Coeff}_\phi[S(P)]$ known) and $\mathrm{Coeff}_\phi[W(P)]$ (with $\mathrm{Coeff}_\phi[SW(P)]$ known). Fig. 8 shows COMPUTE$\circlearrowright$, while Fig. 9 shows a run of COEFF for the function $(x + y)e^{y(x+y)}$ in Example 17.

4. Complexity

It is easy to see that COEFF(i, j) computes $[x^i y^j]\phi(x, y)$ if and only if every call COMPUTE$\circlearrowright$($N^k(E_1^c(P_0))$, $N^{k-1}(E_1^c(P_0))$) terminates and computes the family Coeff_ϕ $(N^k(E_1^c(P_0)))$. Hence, the problem is to analyse the families of coefficients computed by the recursive procedure COMPUTE$\circlearrowright$.

A call COMPUTE$\circlearrowright$(P_{out}, P_{in}) recursively calls itself if and only if P_{out} is singular. So, let $\mathrm{Out}_0 = N^k(E_1^c(P_0))$ and $\mathrm{In}_0 = N^{k-1}(E_1^c(P_0))$ for a suitable integer $k \leqslant c_2$, and consider the sequence of calls

$$\text{COMPUTE}\circlearrowright(\mathrm{Out}_0, \mathrm{In}_0), \ldots, \text{COMPUTE}\circlearrowright(\mathrm{Out}_l, \mathrm{In}_l)$$

contained in the stack associated with the call COMPUTE$\circlearrowright$(Out_0,In_0) (at the bottom).

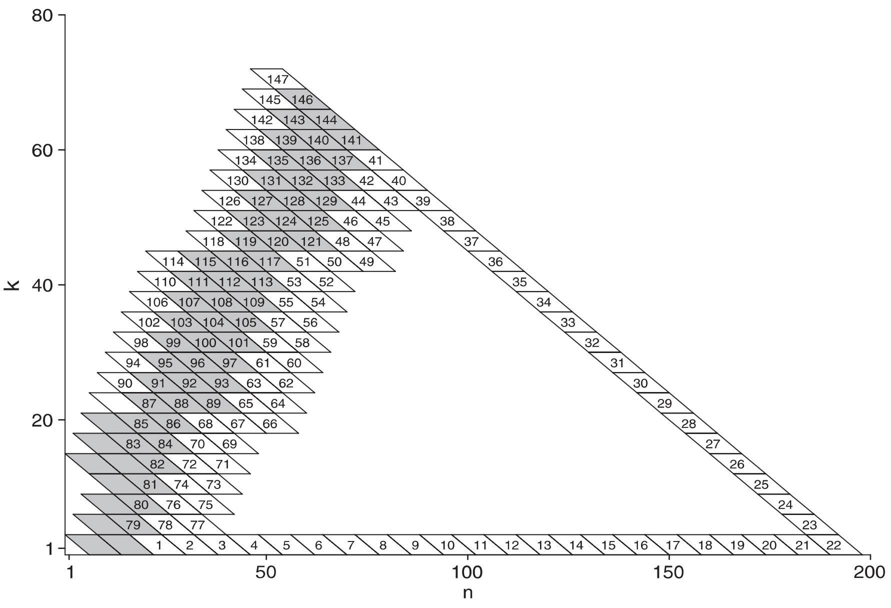


Fig. 9. Running COEFF(55,70) for the function $(x+y)e^{y(x+y)}$ (Example 17). Parallelograms that are not numbered belong to the set I of initial conditions or are computed by PRECOMPUTE.

For each $0 \leqslant p < l$, let $\text{Step}_p = P_{p_1}, \ldots, P_{p_h}$ be the 4-connected sequence of parallelograms 8-adjacent to Out_p such that

$$P_{p_i} = \begin{cases} \text{In}_p & \text{if } i = 1 \\ \text{next}_{\text{Out}_p}(P_{p_{i-1}}) & \text{if } i > 1 \end{cases}$$

and $h = \min\{j \mid \text{next}_{\text{Out}_p}(P_{p_j}) = \text{In}_{p+1}\}$.

We consider three sequences of parallelograms, Seq, $\widetilde{\text{Seq}}$ and $\widehat{\text{Seq}}$, associated with the stack and defined as follows:

- $\text{Seq} = \text{Out}_0, \ldots, \text{Out}_l$ is the 8-connected sequence of singular parallelograms such that $\text{Coeff}_\phi(\text{Out}_i)$ is not known $(0 \leqslant i \leqslant l)$;
- $\widetilde{\text{Seq}} = P_0, E(P_0), \ldots, E^{c_1}(P_0), N(E^{c_1}(P_0)), \ldots, N^{k-1}(E^{c_1}(P_0))$ is the 4-connected ascending sequence of $c_1 + k$ parallelograms such that $\text{Coeff}_\phi(P)$ is known, $P \in \widetilde{\text{Seq}}$;
- $\widehat{\text{Seq}} = \text{Step}_0, \ldots, \text{Step}_{l-1}, \text{In}_l$ is the 4-connected sequence such that for all $P \in \widehat{\text{Seq}}$, $\text{Coeff}_\phi(P)$ has been computed by recursive calls to COMPUTE$\circlearrowright$.

The families of coefficients computed by COMPUTE$\circlearrowright$ are identified by the following:

Lemma 18. *Let* COMPUTE$\circlearrowright(N^k(E^{c_1}(P_0)), N^{k-1}(E^{c_1}(P_0)))$ *be a call occurring in* COEFF. *Then, for all the calls* COMPUTE$\circlearrowright(P_{\text{out}}, P_{\text{in}})$ *that are pushed onto the stack we have*

$$P_{\text{out}} \in Cl^{N^k(E^{c_1}(P_0))}_{N^k(E^{c_1}(P_0))}.$$

Proof. Let $\mathrm{Out}_0 = N^k(E^{c_1}(P_0))$ and let Seq, $\widetilde{\mathrm{Seq}}$ and $\widehat{\mathrm{Seq}}$ be the sequences associated with the stack having the call $\mathrm{COMPUTE}{\circlearrowleft}(\mathrm{Out}_0, S(\mathrm{Out}_0))$ at the bottom. We show that for all Out_i in Seq we have $\mathrm{Out}_i \diamond Cl^{\mathrm{Out}_0}_{\mathrm{Out}_0}$, that is, $\mathrm{Out}_i \diamond^*_{\mathrm{Out}_0} \mathrm{Out}_0$. Hence, since $\mathrm{Out}_i \diamond \mathrm{Out}_{i+1}$ for $0 \leqslant i < l$, it is sufficient to prove that

$$\mathrm{Out}_i \in \mathrm{PAR}_Z(\mathrm{Out}_0). \tag{3}$$

Observe that $\{\mathrm{Seq}\} \cap \{\widehat{\mathrm{Seq}}\} = \{\mathrm{Seq}\} \cap \{\widetilde{\mathrm{Seq}}\} = \emptyset$ and note that we can univocally identify g sequences Seq_i $(1 \leqslant i \leqslant g)$ such that $\mathrm{Seq} = \mathrm{Seq}_1, \mathrm{Seq}_2, \ldots, \mathrm{Seq}_g$, where

- Seq_1 is the maximal descending sequence appearing at the beginning of Seq;
- Seq_{2i} is the maximal ascending sequence after $\mathrm{Seq}_1, \ldots, \mathrm{Seq}_{2i-1}$, $1 < 2i \leqslant g$;
- Seq_{2i+1} is the maximal descending sequence after $\mathrm{Seq}_1, \ldots, \mathrm{Seq}_{2i}$, $1 < 2i+1 \leqslant g$.

We prove (3) by induction on the number g of ascending or descending sequences in the decomposition of Seq above considered.

(*Basis*). Seq consists of one descending sequence $\mathrm{Out}_0, \ldots, \mathrm{Out}_l$ where $\mathrm{Out}_i = E^{w_i}(W^{v_i}(S^{u_i}(\mathrm{Out}_0)))$ and either $\mathrm{Out}_1 = W(\mathrm{Out}_0)$ or $\mathrm{Out}_1 = SW(\mathrm{Out}_0)$. We state that $v_i > w_i$ for $1 \leqslant i \leqslant l$ (and then $\mathrm{Out}_i \in \mathrm{PAR}_Z(\mathrm{Out}_0)$). In fact, let $\hat{\imath} = \min\{i \mid v_i < w_i\}$ (note that $v_{\hat{\imath}} \neq w_{\hat{\imath}}$ since $\{\mathrm{Seq}\} \cap \{\widetilde{\mathrm{Seq}}\} = \emptyset$). Then, we would have $v_{\hat{\imath}-1} > w_{\hat{\imath}-1}$ and $v_{\hat{\imath}} < w_{\hat{\imath}}$, that is, $\mathrm{Out}_{\hat{\imath}-1} \not\diamond \mathrm{Out}_{\hat{\imath}}$.

(*Induction*) $\mathrm{Seq} = \mathrm{Seq}_1, \ldots, \mathrm{Seq}_{n-1}, \mathrm{Seq}_n$. By induction we know that all parallelograms in $\mathrm{Seq}_1, \ldots, \mathrm{Seq}_{n-1}$ satisfy (3). Let $\mathrm{Seq}_n = \mathrm{Out}_s, \ldots, \mathrm{Out}_l$ and let Out_{s-1} be the last parallelogram of Seq_{n-1}. We distinguish two cases.

(*n is odd*) Seq_n is a descending sequence. By induction we have $\mathrm{Out}_{s-1} \in \mathrm{PAR}_Z(\mathrm{Out}_0)$, that is, $\mathrm{Out}_{s-1} = W^{\alpha_{s-1}}(S^{u_{s-1}}(\hat{P}))$ with $\alpha_{s-1}, u_{s-1} \in \mathbb{N}$, $\alpha_{s-1} > 0$. Since $\mathrm{Out}_s = T(\mathrm{Out}_{s-1})$, with $T \in \{SW, S, SE\}$, we have $\mathrm{Out}_s = W^{\alpha_s}(S^{u_s}(\mathrm{Out}_0))$ with $\alpha_s, u_s \in \mathbb{N}$, $\alpha_s \geqslant 0$. Again, $\alpha_s \neq 0$ since $\{\mathrm{Seq}\} \cap \{\widetilde{\mathrm{Seq}}\} = \emptyset$. Now, the same analysis done for the basis shows that parallelograms in Seq_n satisfy (3).

(*n is even*) Seq_n is an ascending sequence, that is, $\mathrm{Out}_s = T(\mathrm{Out}_{s-1})$ with $T \in \{NW, N, NE\}$. We claim that $\mathrm{Out}_{s-1} = NE(\mathrm{Out}_{s-1})$. In fact, each sequence Step_i of parallelograms examined by $\mathrm{COMPUTE}{\circlearrowleft}(\mathrm{Out}_i, \mathrm{In}_i)$ before calling $\mathrm{COMPUTE}{\circlearrowleft}(\mathrm{Out}_{i+1}, \mathrm{In}_{i+1})$ is 4-connected. This means that $\mathrm{In}_{s-1} \square \mathrm{Out}_{s-1}$. In particular, $\mathrm{In}_{s-1} = N(\mathrm{Out}_{s-1})$ since in the other three cases the call $\mathrm{COMPUTE}{\circlearrowleft}(\mathrm{Out}_{s-1}, \mathrm{In}_{s-1})$ would compute $\mathrm{Coeff}_\phi(\mathrm{Out}_{s-1})$ without any recursion. Therefore, $\mathrm{COMPUTE}{\circlearrowleft}(\mathrm{Out}_{s-1}, \mathrm{In}_{s-1})$ recursively calls $\mathrm{COMPUTE}{\circlearrowleft}(NE(\mathrm{Out}_{s-1}), \mathrm{In}_s)$ with $\mathrm{In}_s = \mathrm{In}_{s-1} = N(\mathrm{Out}_{s-1}) \in \widehat{\mathrm{Seq}}$.

Now, consider the 4-connected sequence $\widehat{\widehat{\mathrm{Seq}}}$ obtained by joining $\widetilde{\mathrm{Seq}}$ to $\widehat{\mathrm{Seq}}$,

$$\begin{aligned}\widehat{\widehat{\mathrm{Seq}}} = {} & P_0, E(P_0), \ldots, E^{c_1}(P_0), N(E^{c_1}(P_0)), \ldots, N^{k-1}(E^{c_1}(P_0)),\\ & \mathrm{Step}_0, \ldots, \mathrm{Step}_{s-1}, \mathrm{In}_s.\end{aligned}$$

We have $\{\widehat{\widehat{\mathrm{Seq}}}\} \cap \{\mathrm{Seq}_n\} = \emptyset$. In fact, $\mathrm{Coeff}_\phi(P)$ is known if $P \in \widehat{\widehat{\mathrm{Seq}}}$ and unknown otherwise $(P \in \mathrm{Seq}_n)$. Informally, this means that the parallelograms in the ascending sequence Seq_n are restricted to lie in a closed area (delimited by $\widehat{\widehat{\mathrm{Seq}}}$) consisting of parallelograms that satisfy (3). $\square$

An immediate consequence of the previous lemma is:

Corollary 19. *Let* $P_k = N^k(E^{c_1}(P_0))$ *be a singular parallelogram. If a call* COMPUTE$\circlearrowright$ $(P_k, S(P_k))$ *occurring in* COEFF *computes* $Coeff_\phi(P)$, *then*

$$P \in B(Cl^{P_k}_{P_k}) \cup Cl^{P_k}_{P_k}.$$

Proof. By inspecting the code of COMPUTE$\circlearrowright$, we note that if it computes $\mathrm{Coeff}_\phi(P)$, either P is singular (and COMPUTE$\circlearrowright(P, P_{\mathrm{in}})$ is a call generated by COMPUTE$\circlearrowright(P_k, S(P_k))$) or P is nonsingular and 8-adjacent to a singular parallelogram P_s such that there exists a recursive call COMPUTE$\circlearrowright(P_s, Q)$, generated by COMPUTE$\circlearrowright(P_k, S(P_k))$, that computes $\mathrm{Coeff}_\phi(P)$. In the first case Lemma 18 states that $P \in Cl^{P_k}_{P_k}$ while in the second we have $P \in B(Cl^{P_k}_{P_k})$. □

Lemma 20. *The stack associated with a call* COMPUTE$\circlearrowright(P, S(P))$ *occurring in* COEFF *does not contain two identical calls.*

Proof (*by contradiction*). Let COMPUTE$\circlearrowright(\mathrm{Out}_h, \mathrm{In}_h)$ be the first repeated occurrence of a call, that is, $h = \min\{0 \leqslant i \leqslant l \mid \exists\, \delta > 0,\ \mathrm{Out}_i = \mathrm{Out}_{i-\delta} \wedge \mathrm{In}_i = \mathrm{In}_{i-\delta}\}$. Without loss of generality, we suppose that $\mathrm{In}_h = W(\mathrm{Out}_h)$. Consider the 8-connected subsequence of Seq given by

$$\mathrm{S} = \mathrm{Out}_{h-\delta}, \mathrm{Out}_{h-\delta+1}, \ldots, \mathrm{Out}_h,$$

and the 4-connected subsequence of $\widehat{\mathrm{Seq}}$,

$$\widehat{\mathrm{S}} = \mathrm{Step}_{h-\delta}, \mathrm{Step}_{h-\delta+1}, \ldots, \mathrm{Step}_{h-1}, \mathrm{In}_h.$$

For each $P \in \widehat{\mathrm{S}}$, the family $\mathrm{Coeff}_\phi(P)$ is known. Moreover, we have that for each $P \in \widehat{\mathrm{S}}$ ($P \in \mathrm{S}$) there exists $Q \in \mathrm{S}$ ($Q \in \widehat{\mathrm{S}}$) such that $P \diamond Q$. Note that both sequences are "closed", that is, their first and last parallelograms coincide. Moreover, we have $\{\mathrm{S}\} \cap \{\widehat{\mathrm{S}}\} = \emptyset$.

For each closed sequence S, denote by Inside(S) the set of all parallelograms in PAR($\overline{P}$) that lie in the area surrounded by S. Then, it is immediate to observe that we have only two cases:

$\widehat{\mathrm{S}} \subseteq$ *Inside*(S). This means that if $P \in \widehat{\mathrm{S}}$ it is impossible to find a 4-connected sequence $T_P = P_0, \ldots, P$ such that $\{T_P\} \cap \{\mathrm{S}\} = \emptyset$. On the other hand, we know that for every $P \in \widehat{\mathrm{S}}$ there exists a 4-connected sequence T_P from P_0 to P, consisting of parallelograms in PAR($\overline{P}$), such that if $Q \in T_P$ the family $\mathrm{Coeff}_\phi(Q)$ has been computed (see the sequence $\widehat{\widehat{\mathrm{Seq}}}$ in the proof of Lemma 18). Therefore, we have $\widehat{\mathrm{S}} \nsubseteq \mathrm{Inside}(\mathrm{S})$.

$\mathrm{S} \subseteq$ *Inside*($\widehat{\mathrm{S}}$). Let $k_1, k_2 \in \mathbb{N}$ such that

$$N^{k_1}(E^{k_2}(P_0)) \in \mathrm{S} \quad \text{and} \quad N^{h_1}(E^{h_2}(P_0)) \in \mathrm{S} \Rightarrow k_1 + k_2 \leqslant h_1 + h_2.$$

Let $\mathrm{Out}_{\overline{h}} = N^{k_1}(E^{k_2}(P_0))$. Since $\mathrm{S} \subseteq \mathrm{Inside}(\widehat{\mathrm{S}})$, it is immediate to prove that $S(\mathrm{Out}_{\overline{h}})$ and $W(\mathrm{Out}_{\overline{h}})$ belong to $\widehat{\mathrm{S}}$. More precisely, because $\widehat{\mathrm{S}}$ is 4-connected, it follows that

$$\widehat{\mathrm{S}} = \mathrm{In}_h, \ldots, S(\mathrm{Out}_{\overline{h}}), SW(\mathrm{Out}_{\overline{h}}), W(\mathrm{Out}_{\overline{h}}), \ldots, \mathrm{In}_h.$$

Then, the call computing $\mathrm{Coeff}_\phi(SW(\mathrm{Out}_{\overline{h}}))$ is $\mathrm{COMPUTE}\circlearrowleft(\mathrm{Out}_{\overline{h}}, \mathrm{In}_{\overline{h}})$. By observing the code, we note that if $\mathrm{COMPUTE}\circlearrowleft(\mathrm{Out}_{\overline{h}}, \mathrm{In}_{\overline{h}})$ computes $\mathrm{Coeff}_\phi(SW(\mathrm{Out}_{\overline{h}}))$ then it has previously computed $\mathrm{Coeff}_\phi(S(\mathrm{Out}_{\overline{h}}))$ and it necessarily computes $\mathrm{Coeff}_\phi(W(\mathrm{Out}_{\overline{h}}))$. So, this call would terminate without recursion. □

The following lemma states that we can develop an efficient data structure for storing all the coefficients computed by the algorithm.

Lemma 21. *The data structure* $\mathrm{Coeff}_\phi[]$ *can be implemented in space* $O(i+j)$ *and accessed in time* $O(1)$.

Proof. $\mathrm{Coeff}_\phi[]$ could be easily implemented as a matrix $M_{c_1+1\times c_2+1}$ that requires $O(i \cdot j)$ space and admits $O(1)$ access time. We give here an outline of how to lower the space requirement to $O(i + j)$. Let us consider two integers

$$D_{\max_n} = \max\{\deg_n(a_r(n,k)), \deg_n(a_0(n,k)), \deg_n(b_s(n,k)), \deg_n(b_0(n,k))\},$$

$$D_{\max_k} = \max\{\deg_k(a_r(n,k)), \deg_k(a_0(n,k)), \deg_k(b_s(n,k)), \deg_k(b_0(n,k))\},$$

univocally associated with an instance $\langle B_n, B_k, \mathrm{Sec}, U_n, U_k, I, i, j\rangle$ (see also Lemma 13). Consider now the subset $L \subset \{0, 1, \ldots, c_2\}$ such that, for $c \in L$, we have $\sharp\{N^c(E^h(P_0)) \in \mathrm{PAR}_Z(\overline{P})\} = \Theta(i + j)$. It is easily shown that $\sharp L \leqslant 4D_{\max_k}$, since for each $c \in L$ there is necessarily a factor of type $(k-k_1)^{\alpha_1}$ in one of the polynomials a_0, a_r, b_0, b_s. Moreover, for $c \in \{0, 1, \ldots, c_2\} \setminus L$ we have $\sharp\{N^c(E^h(P_0)) \in \mathrm{PAR}_Z(\overline{P})\} < \zeta = 4D_{\max_n}(b+1) = O(1)$. In other words, we have to develop an efficient implementation for a sparse matrix having $O(c_1 + c_2)$ entries different from 0.

So, we define an array of $c_2 + 1$ links to tables of size 9ζ (for each singular parallelogram we have to consider at most eight 8-adjacent parallelograms). The ith element of the array allows us to access the families $\mathrm{Coeff}_\phi(N^i(E^h(P_0)))$. As long as a table is not full it supports insertion operations in time $O(9\zeta) = O(1)$. Suppose that the cth table is full and we want to insert a new element: this means that c belongs to L, so we dynamically replace the cth table with an array of size $c_1 = O(i + j)$ in order to maintain $O(1)$ access time. Note that the number of replacements is at most $3 \cdot \sharp L$. Therefore, $\mathrm{Coeff}_\phi[]$ can be implemented in space $O(i + j)$ and accessed in time $O(1)$. □

Theorem 22. *The total number of calls to* $\mathrm{COMPUTE}\circlearrowleft$ *during the execution of* $\mathrm{COEFF}(i, j)$ *is* $O(i + j)$.

Proof. Recall that $(i, j) \in \overline{P} = P(\bar{\imath}, \bar{\jmath}) = N^{c_2}(E^{c_1}(P_0))$, with $c_2 = O(j)$ and $c_1 = O(i + j)$. Let

$$\mathrm{COMPUTE}\circlearrowleft(P_1, S(P_1)), \ldots, \mathrm{COMPUTE}\circlearrowleft(P_t, S(P_t))$$

be the sequence of calls observed in $\mathrm{COEFF}(i, j)$ $(t = O(j))$. Moreover, let

$$\mathrm{TOT} = \left\{\mathrm{COMPUTE}\circlearrowleft(P_1, P_2) \mid P_1 \in \mathrm{PAR}_Z(\overline{P}),\ P_1 \square P_2\right\}$$

and, for $1 \leqslant k \leqslant t$, let

$$\mathrm{TOT}_k = \{\mathrm{C} \in \mathrm{TOT} \mid \mathrm{C} \text{ is a call originated by COMPUTE}\circlearrowleft(P_k, S(P_k))\}\,.$$

For $1 \leqslant k \leqslant t$, each call COMPUTE$\circlearrowleft(P_k, S(P_k))$ recursively generates calls of type COMPUTE$\circlearrowleft(P, Q)$, with $P \in \mathrm{Cl}_{P_k}^{P_k}$, such that $\mathrm{Coeff}_\phi(P)$ has not previously been computed by COMPUTE$\circlearrowleft(P_l, S(P_l))$ with $1 \leqslant l < k$. In other words, $\mathrm{TOT}_l \cap \mathrm{TOT}_m = \emptyset$, for $l \neq m$.

Lemma 20 guarantees that COMPUTE$\circlearrowleft(P_k, S(P_k))$ generates exactly $\sharp\mathrm{TOT}_k$ recursive calls. Hence, recalling Lemma 13, the total number of calls is

$$\sum_{k=1}^{t} \sharp\mathrm{TOT}_k = \sharp \bigcup_{k=1}^{t} \mathrm{TOT}_k \leqslant \sharp\mathrm{TOT} = 4 \cdot \sharp\mathrm{PAR}_Z(\overline{P}) = \mathrm{O}(\bar{\imath} + \bar{\jmath})$$
$$= \mathrm{O}(i + j). \qquad \square$$

At last, we have:

Theorem 23. COEFF(i, j) *runs in time* $\mathrm{O}(i + j)$ *and in space* $\mathrm{O}(i + j)$.

Proof. We have already noted that PRECOMPUTE(Sec, B_n, B_k, I, i, j) can be easily implemented in time $\mathrm{O}(i + j)$ and space $\mathrm{O}(i + j)$. Then, by Theorem 22, we know that COMPUTE$\circlearrowleft$ is called $\mathrm{O}(i + j)$ times. By inspecting the code, we note that each call consists of a constant number of operations because the cost of accessing $\mathrm{Coeff}_\phi[]$ is $\mathrm{O}(1)$ (see Lemma 21). Moreover, the space requirement is bounded by the sum of the maximum stack size and the size of the data structure $\mathrm{Coeff}_\phi[]$. So, we conclude that COEFF(i, j) runs in time $\mathrm{O}(i + j)$ using $\mathrm{O}(i + j)$ space. $\square$

5. Conclusions

In this paper we have presented an algorithm that computes the coefficient $[x^i y^j]\phi(x, y)$ of a bivariate holonomic formal series $\phi(x, y)$ in time and space $\mathrm{O}(i + j)$ (under the uniform cost criterion).

We recall that this is an improvement with respect to the algorithm presented in [9] that runs in time $\mathrm{O}(i \cdot j)$ in the bivariate case. So, it would be interesting to study whether the technique we have presented can be modified in order to deal with more than two variables. More precisely, we would like to answer to the question: is there an algorithm that solves the coefficient problem for $\phi \in \mathbb{Q}[[X]]_h$ with $X = \{x_1, \ldots, x_n\}, n > 2$, in time $\mathrm{O}(i_1 + \cdots + i_n)$? In case of a positive answer, we would have a dramatic improvement of the $\mathrm{O}(i_1 \cdot \ldots \cdot i_n)$ upper bound given by Lipshitz's algorithm. Actually, it is quite natural to extend our method to holonomic power series in three variables but this leads to a quadratic algorithm (versus the cubic Lipshitz's algorithm). Therefore, a deeper investigation is needed.

Last but not least, we would like to develop a robust implementation of the algorithm for a computer algebra system. For testing purposes, we took advantage of the package 'Mgfun' (implemented by Chyzak in [3]) to compute the recurrences needed by the algorithm and to develop a prototypical implementation under Maple.

Acknowledgements

The authors wish to thank the anonymous referees for their accurate suggestions.

References

[1] I.N. Bernstein, Modules over a ring of differential operators, study of the fundamental solutions of equations with constant coefficients, Functional Anal. Appl. 5 (1971) 1–16 (Russian), pages 89–101 (English).

[2] A. Bertoni, P. Massazza, R. Radicioni, Random generation of words in regular languages with fixed occurrences of symbols, in: Proc. of WORDS'03, Fourth Internat. Conf. on Combinatorics of Words, Turku, Finland, 2003, pp. 332–343.

[3] F. Chyzak, An extension of Zeilberger's fast algorithm to general holonomic functions, in: Proc. of FPSAC 1997, Universität Wien, 1997, pp. 172–183.

[4] F. Chyzak, B. Salvy, Non-commutative elimination in ore algebras proves multivariate identities, J. Symbolic Comput. 26 (1998) 187–227.

[5] A. Denise, Génération aléatoire uniforme de mots de langages rationnels, Theoret. Comput. Sci. 159 (1996) 43–63.

[6] I. Dutour, J.M. Fédou, Object grammars and random generation, Discrete Math. and Theoret. Comput. Sci. 2 (1998) 47–61.

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

[8] L. Lipshitz, The diagonal of a D-Finite power series is D-Finite, J. Algebra 113 (1988) 373–378.

[9] L. Lipshitz, D-Finite power series, J. Algebra 122 (1989) 353–373.

[10] P. Massazza, R. Radicioni, On computing the coefficients of rational formal series, in: Proc. of FPSAC'04, 16th Annu. Internat. Conf. on Formal Power Series and Algebraic Combinatorics, Vancouver, Canada, 2004, pp. 211–226, available at <http://fpsac.labri.fr/FPSAC04/fpsac04.html>.

[11] R.P. Stanley, Differentiably Finite Power Series, European J. Combin. 1 (1980) 175–188.

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