

On Computing the Coefficients of Rational Formal Series

Paolo Massazza and Roberto Radicioni

Abstract. *In this work we study the problem of computing the coefficients of rational formal series in two commuting variables. Given a rational formal series $\phi(x, y) = \sum_{n, k \geq 0} c_{nk} x^n y^k = P(x, y)/Q(x, y)$ with $P, Q \in \mathbb{Q}\{x, y\}$ and $Q(0, 0) \neq 0$, 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.*

RÉSUMÉ. *Dans cet article, nous étudions le problème du calcul des coefficients de séries formelles rationnelles en deux variables commutatives. Étant donné une série formelle rationnelle $\phi(x, y) = \sum_{n, k \geq 0} c_{nk} x^n y^k = P(x, y)/Q(x, y)$ ou $P, Q \in \mathbb{Q}\{x, y\}$ et $Q(0, 0) \neq 0$, nous montrons que le coefficient $[x^i y^j] \phi(x, y)$ peut être calculé en un temps $O(i + j)$ sous le critère de coût uniforme.*

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 easily reduced to computing the coefficients of suitable formal power series: the property is codified into a weight function $w : S \rightarrow \mathbb{N}$ and the formal series $\sum_{s \in S} w(s) s$ is considered. Then, the counting problem associated with S and w consists of computing the function $f(n) = \#\{s \in S | w(s) = n\}$.

Another setting where the Coefficient Problem arises is the random generation of combinatorial structures (see, for instance, [10]). Efficient algorithms for the random generation of strings in a language can also be derived by exploiting the generating function associated with the language (see, for example, [7]).

A likewise important and (intuitively) more general version of the Coefficient Problem can be stated considering a 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 . When dealing with counting and random generation, this generalization appears whenever a multiple output weight function $w : S \rightarrow \mathbb{N}^k$ is considered. Some examples are the problem of counting and random sampling words with fixed occurrences of each letter of the alphabet ([2], [8]) or the random generation through object grammars ([9]).

In this paper we consider the Coefficient Problem for the class $\mathbb{Q}[[\{x, y\}]]_r$ of the rational formal series in two commuting variables. These are power series expansions of functions of the form $P(x, y)/Q(x, y)$ where P, Q are polynomials with rational coefficients and $Q(0, 0) \neq 0$.

We show that given a couple of integers (i, j) and a couple of polynomials $P, Q \in \mathbb{Q}\{x, y\}$, with $Q(0, 0) \neq 0$, the coefficient $[x^i y^j] \phi(x, y)$ of $\phi(x, y) = P(x, y)/Q(x, y)$ can be computed in time $O(i + j)$ under

Key words and phrases. holonomic functions, shift algebra.

This work has been supported by the project M.I.U.R. COFIN "Formal Languages and Automata: Theory and Applications".

the uniform cost criterion. Our method is based on the theory of holonomic power series. We derive suitable recurrence equations with polynomial coefficients from the holonomic system associated with $\phi(x, y)$, then we use them, together with a recurrence with constant coefficients, in order to compute $[x^i y^j] \phi(x, y)$ in an efficient way. This is a significant improvement on a more general algorithm presented in [11], that let us 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 $e \in \mathbb{Q}$: $e \cdot \{a_{nk}\} = \{ea_{nk}\}$, $e \in \mathbb{Q}$
- Shift : $E_n \{a_{nk}\} = \{a_{n-1k}\}$, $E_k \{a_{nk}\} = \{a_{nk-1}\}$
- Multiplication by n, k : $n \{a_{nk}\} = \{na_{nk}\}$, $k \{a_{nk}\} = \{ka_{nk}\}$

Then, the so called *shift algebra* $\mathbb{Q}\langle n, k, E_n, E_k \rangle$ is a particular Ore algebra (see, for instance, [6]) and can be interpreted as a (noncommutative) ring of linear operators on $\mathbb{Q}^{(2)}$, with pseudo-commutative rules given by:

$$\begin{aligned} nk &= kn, & nE_k &= E_k n, & kE_n &= E_n k, \\ nE_n &= E_n n + E_n, & kE_k &= E_k k + E_k. \end{aligned}$$

More simply, a polynomial in $\mathbb{Q}\langle n, k, E_n, E_k \rangle$ represents a linear recurrence with polynomial coefficients.

2.1. Rational formal series and Holonomic functions. Let Σ^c be the commutative free monoid generated by a finite alphabet Σ . Given a commutative ring \mathbb{K} , a *formal series* ψ in commutative variables Σ is a function $\psi : \Sigma^c \mapsto \mathbb{K}$, usually indicated by $\sum_{\underline{x} \in \Sigma^c} \psi(\underline{x}) \underline{x}$; the *support* of ψ is the set of monomials $\{\underline{x} \in \Sigma^c \mid \psi(\underline{x}) \neq 0\}$. We denote by $\mathbb{K}[[\Sigma]]$ the ring of commutative formal series with coefficients in \mathbb{K} equipped with the usual operations of sum (+) and product (\cdot). Formal series with finite support belong to the ring of polynomials $\mathbb{K}[\Sigma]$. The ring of rational formal series $\mathbb{K}[[\Sigma]]_r$ can be defined as the smallest subring of $\mathbb{K}[[\Sigma]]$ containing $\mathbb{K}[\Sigma]$ and rationally closed (i.e. closed with respect to $\star, +, \cdot$ and the two external products of \mathbb{K} on $\mathbb{K}[[\Sigma]]$ — where \star is the usual closure operation that is defined for proper series, i.e. series ψ s.t. $\psi(\epsilon) = 0$).

In the sequel we will consider the alphabet $X = \{x, y\}$ and $\mathbb{K} = \mathbb{Q}$. A rational formal series $\phi \in \mathbb{Q}[[X]]_r$ is then the power series expansion of a suitable rational function,

$$\phi(x, y) = \sum_{n, k \in \mathbb{N}} c_{nk} x^n y^k = \frac{P(x, y)}{Q(x, y)} \quad P, Q \in \mathbb{Q}[X], \quad Q(0, 0) \neq 0.$$

We often use the notation $[x^n y^k] \phi(x, y)$ to indicate the coefficient c_{nk} of a formal series ϕ . We refer to [3] for a detailed analysis of the class of the rational series.

It is well known (see, for example, [14]) that the class of the rational functions is properly contained in the class of the holonomic functions defined as follows.

Definition 2.1. A function $\phi(x, y)$ is *holonomic* iff there exist some polynomials

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

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 .

Holonomic systems were first introduced by I.N. Bernstein in the 1970s ([1]) and deeply investigated by Stanley, Lipshitz, Zeilberger et al. (see [5], [11], [13] and [14]). In this setting, we are interested in the following result:

Theorem 2.2. *Let $\phi(x, y) = \sum_{n,k \geq 0} c_{nk} x^n y^k$ be a holonomic function. Then the sequence of coefficients $\{c_{nk}\}$ satisfies a system of linear recurrence equations with polynomial coefficients*

$$(2.1) \quad S = \begin{cases} P(n, k, E_n)\{c_{nk}\} = 0 \\ Q(n, k, E_k)\{c_{nk}\} = 0 \end{cases}$$

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 $\mathbb{Q}\langle n, k, E_n, E_k \rangle$.

PROOF. See, for instance, [11]. □

A direct consequence of the previous theorem is that the sequence of the coefficients of a rational formal series satisfies a system of recurrence equations of type (2.1): we give here an outline of how to compute such a system.

In [14] it is shown how to compute a holonomic system $\{D_1, D_2\}$ associated with a rational function $\phi(x, y)$. Then, given $\{D_1, D_2\}$, we can obtain in two steps a system of recurrences $S = \{P, Q\}$ of type (2.1) satisfied by the sequence $\{c_{nk}\}$. First, we compute two operators $w_1, w_2 \in \mathbb{Q}\langle n, k, E_n, E_k \rangle$ such that $w_1(n, E_n, E_k)\{c_{n,k}\} = w_2(k, E_n, E_k)\{c_{n,k}\} = 0$. This is easily done by observing the following correspondence between operators

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

Then, we get the first recurrence $P(n, k, E_n)$ by solving an elimination problem in $\mathbb{Q}\langle n, k, E_n, E_k \rangle$. This can be done, for example, by computing the Gröbner basis associated with w_1, w_2 with respect to a suitable ordering on $\mathbb{Q}\langle n, k, E_n, E_k \rangle$. We proceed similarly in order to get the second recurrence $Q(n, k, E_k)$. Useful packages for such computations have been recently developed (see [6], [12]).

We recall that the coefficients of a rational series satisfy a linear recurrence with constant coefficients. More formally, we have the following:

Theorem 2.3. *Let be $\phi(x, y) = \sum_{n,k \geq 0} c_{nk} x^n y^k = P(x, y)/Q(x, y)$ with $P, Q \in \mathbb{Q}[X]$ and $Q(0, 0) \neq 0$. Then the sequence of coefficients $\{c_{nk}\}$ satisfies a linear recurrence equation with constant coefficients*

$$(2.2) \quad B(E_n, E_k)\{c_{nk}\} = 0$$

where $B(E_n, E_k) = \sum_{i,j=0}^{r,s} q_{ij} E_n^i E_k^j$, $q_{ij} \in \mathbb{Q}$ and $q_{00} \neq 0$.

PROOF. Let be $P(x, y) = \sum_{i,j=0}^{r_1,s_1} p_{ij} x^i y^j$ and $Q(x, y) = \sum_{i,j=0}^{r_2,s_2} q_{ij} x^i y^j$. Then we have

$$Q(x, y)\phi(x, y) = \sum_{n,k \geq 0} \left(\sum_{\substack{i_1+i_2=n \\ j_1+j_2=k}} q_{i_1 j_1} c_{i_2 j_2} \right) x^n y^k = \sum_{i,j=0}^{r_1,s_1} p_{ij} x^i y^j.$$

So, for $n > r_1$ and $k > s_1$ it holds $\sum_{i,j=0}^{r_2,s_2} q_{ij} c_{n-ik-j} = \sum_{i,j=0}^{r_2,s_2} q_{ij} E_n^i E_k^j c_{nk} = 0$. □

A naive method for computing the coefficient c_{ij} of a rational series in time $O(ij)$ can be obtained as an immediate application of the theorem above. Linear recurrences with constant coefficients have been deeply studied in [4], where it is shown a necessary and sufficient condition that let us to define a well ordering on the elements of the solution. Moreover, it is also shown that the solutions of such recurrences can have generating functions that are not rational (they can be algebraic, holonomic or even non-holonomic).

3. Computing the coefficient

Given a rational formal series $\phi = \sum_{n,k \in \mathbb{N}} c_{nk} x^n y^k$, by applying Theorem 2.3 we can easily obtain a linear recurrence equation with constant coefficients satisfied by $\{c_{nk}\}$. Then, we can use it for computing an arbitrary coefficient c_{ij} once a suitable set of initial conditions is known. On the other hand, because in general both E_n and E_k appear in the recurrence, this technique requires $O(ij)$ coefficients in order to determine c_{ij} .

As shown before, the theory of holonomic systems let us to obtain particular linear recurrence equations with polynomial coefficients that are more suitable for computing coefficients. More precisely, we get two operators in 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 $n = \bar{n}$ or $k = \bar{k}$ if the leading and the least coefficients of the recurrences do not vanish on that line.

Our approach takes advantage of both types of recurrences in order to get a method that efficiently computes the coefficient c_{ij} by starting with a suitable set of initial conditions and proceeding by choosing at each step the “right” recurrence to use.

More formally, we consider the Coefficient Problem for rational series defined as follows.

Problem: to determine the coefficient c_{ij} in the power series expansion of a rational series $\phi(x, y)$.

Input: A tuple $\langle \mathcal{N}, \mathcal{K}, \mathcal{B}, I, i, j \rangle$ where:

-: \mathcal{N}, \mathcal{K} and \mathcal{B} are three recurrence equations of type

$$(3.1) \quad \mathcal{N}(n, k, E_n) = \sum_{i=0}^r p_i(n, k) E_n^i \quad p_i(n, k) \in \mathbb{Q}[\{n, k\}], \quad p_r(n, k) \neq 0$$

$$(3.2) \quad \mathcal{K}(n, k, E_k) = \sum_{j=0}^s q_j(n, k) E_k^j \quad q_j(n, k) \in \mathbb{Q}[\{n, k\}], \quad q_s(n, k) \neq 0$$

$$(3.3) \quad \mathcal{B}(E_n, E_k) = \sum_{i,j=0}^{r_2, s_2} h_{ij} E_n^i E_k^j \quad h_{ij} \in \mathbb{Q}, \quad h_{00} \neq 0$$

satisfied by the sequence $\{c_{nk}\}$.

-: I is a suitable set of initial conditions for $\mathcal{N}, \mathcal{K}, \mathcal{B}$, i.e. the set of coefficients

$$I = \{c_{nk} \mid (0 \leq n \leq \alpha \wedge 0 \leq k \leq e) \vee (0 \leq n \leq e \wedge 0 \leq k \leq \beta)\}$$

with

- $e = \max\{r, s, r_2, s_2\}$
- $\alpha = \max\{n \in \mathbb{N} \mid p_r(n, k) = 0 \vee p_0(n, k) = 0, 0 \leq k \leq e\}$
- $\beta = \max\{k \in \mathbb{N} \mid q_s(n, k) = 0 \vee q_0(n, k) = 0, 0 \leq n \leq e\}$

-: $(i, j) \in \mathbb{N}^2$.

Output: the coefficient c_{ij} .

3.1. Clusters and coefficients. In this section we give some definitions and prove some basic results that are useful to describe the behaviour of the algorithm.

Definition 3.1 ($R^{(e)}(x, y)$). Let be $e, x, y \in \mathbb{N}$. The *square* $R^{(e)}(x, y)$ is the set of points

$$R^{(e)}(x, y) = \{(x', y') \mid (x', y') \in \mathbb{N}^2, x - e < x' \leq x \wedge y - e < y' \leq y\}.$$

Note that each square is identified by the point on the right upper corner.

Definition 3.2 ($\text{SQ}^{(e)}$). Let be $e \in \mathbb{N}$. Then

$$\text{SQ}^{(e)} = \{R^{(e)}(x, y) \mid (x, y) \in \mathbb{N}^2, x \geq e, y \geq e\}.$$

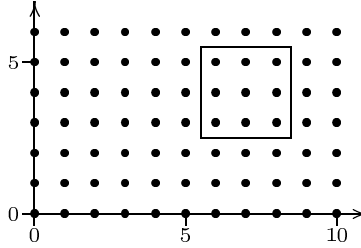


FIGURE 1. The square $R^{(3)}(8, 5)$.

We give a notion of neighbor of a square introducing the following partial functions from $\text{SQ}^{(e)}$ to $\text{SQ}^{(e)}$:

$$\begin{aligned} N(R^{(e)}(x, y)) &= R^{(e)}(x, y + e) \\ S(R^{(e)}(x, y)) &= R^{(e)}(x, y - e) \quad (\text{defined if } y \geq 2e) \\ E(R^{(e)}(x, y)) &= R^{(e)}(x + e, y) \\ W(R^{(e)}(x, y)) &= R^{(e)}(x - e, y) \quad (\text{defined if } x \geq 2e) \end{aligned}$$

Let be $T \in \{N, E, S, W\}$, then we often write $T^i(R)$ for $T(T^{i-1}(R))$, $T^0(R) = R$. Moreover, we also consider the shortcuts $SW(R) = S(W(R))$, $NW(R) = N(W(R))$, $SE(R) = S(E(R))$ and $NE(R) = N(E(R))$.

Definition 3.3 ($\text{SQ}^{(e)}(\bar{R})$, $\text{SQ}_V^{(e)}(\bar{R})$). Let be $e \in \mathbb{N}$ and $V \subseteq \mathbb{N}^2$. Given $\bar{R} \in \text{SQ}^{(e)}$ such that $\bar{R} = N^c(W^d(R^{(e)}(e, e)))$ ($c, d \in \mathbb{N}$) we define

$$\begin{aligned} \text{SQ}^{(e)}(\bar{R}) &= \left\{ R \in \text{SQ}^{(e)} \mid \exists u, v \in \mathbb{N}^2, R = W^u(S^v(\bar{R})) \right\} \\ \text{SQ}_V^{(e)}(\bar{R}) &= \{ R \in \text{SQ}^{(e)}(\bar{R}) \mid R \cap V \neq \emptyset \} \end{aligned}$$

We introduce a reflexive and symmetric relation $\diamond \subset \text{SQ}^{(e)} \times \text{SQ}^{(e)}$:

Definition 3.4 (\diamond). $R_1^{(e)}$ is a *neighbor* of $R_2^{(e)}$, $R_1^{(e)} \diamond R_2^{(e)}$, if and only if

$$\exists T \in \{N, NE, E, SE, S, SW, W, NW\} \quad \text{s.t.} \quad R_1^{(e)} = T(R_2^{(e)}).$$

Particular sequences of squares will be of interest when considering the behaviour of the algorithm.

Definition 3.5. Let $\text{Seq} = R_1, \dots, R_k$ be a sequence of squares in $\text{SQ}^{(e)}$. Then, Seq is

- *8-connected* iff for $1 \leq i < k$ it holds $R_i \diamond R_{i+1}$
- *4-connected* iff for $1 \leq i < k$ it holds $R_{i+1} = T_i(R_i)$ with $T_i \in \{N, E, S, W\}$
- *descending* iff Seq is 8-connected or 4-connected and for $1 \leq i < k$ it holds $R_{i+1} = T_i(R_i)$ with $T_i \in \{E, SE, S, SW, W\}$
- *ascending* iff Seq is 8-connected or 4-connected and for $1 \leq i < k$ it holds $R_{i+1} = T_i(R_i)$ with $T_i \in \{W, NW, N, NE, E\}$

Henceforward, we fix an instance $\langle \mathcal{N}, \mathcal{K}, \mathcal{B}, I, i, j \rangle$ of the Coefficient Problem for a series $\phi(x, y) \in \mathbb{Q}[[X]]_r$ and we associate with it the following values:

- $Z = Z(\mathcal{N}, \mathcal{K}) = \{(x, y) \in \mathbb{N}^2 \mid p_r(x, y) = 0 \vee p_0(x, y) = 0 \vee q_s(x, y) = 0 \vee q_0(x, y) = 0\}$.
- $e = e(\mathcal{N}, \mathcal{K}) = \max\{r, s, r_2, s_2\}$
- $R_0 = R^{(e)}(e - 1, e - 1)$
- $\bar{R} = \bar{R}(\bar{i}, \bar{j}) = N^c(E^d(R_0))$ with $c = \lfloor j/e \rfloor$ $d = \lfloor i/e \rfloor$.

Once we have fixed the values above, we can write R for $R^{(e)}$ and SQ for $\text{SQ}^{(e)}$ whenever the context is clear. Note that $\text{SQ}_Z(\bar{R})$ consists of those squares in $\text{SQ}(\bar{R})$ that contain at least one point (n, k) such that at least one of the following methods fails:

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

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

The next lemma will be frequently used in the sequel.

Lemma 3.6. *Let be $\overline{R} = \overline{R}(\overline{i}, \overline{j})$. Then the number of squares in $SQ(\overline{R})$ containing at least one point for which $\text{Compute}_{\mathcal{N}}(n, k)$ or $\text{Compute}_{\mathcal{K}}(n, k)$ fail is $\#SQ_Z(\overline{R}) = O(\overline{i} + \overline{j})$.*

PROOF. We first note that if $(x, y) \in R$ with $R \in SQ_Z(\overline{R})$ then $0 \leq x \leq \overline{i}$ and $0 \leq y \leq \overline{j}$. Then, consider the set $Z_{\overline{j}} = \{(x, y) \in Z \mid 0 \leq y \leq \overline{j}\}$ and observe that $\#Z_{\overline{j}} \leq [\deg_n(p_0(n, k)) + \deg_n(p_r(n, k)) + (\deg_n(q_0(n, k)) + \deg_n(q_s(n, k)))](\overline{j} + 1) = O(\overline{j}) = O(\overline{i} + \overline{j})$. Since each square in $SQ_Z(\overline{R})$ contains at least one point in $Z_{\overline{j}}$, we have $\#SQ_Z(\overline{R}) = O(\overline{i} + \overline{j})$. \square

In the sequel, we will denote by $\text{Coeff}_{\phi}(R)$ the set of the coefficients of ϕ associated with R , that is,

$$\text{Coeff}_{\phi}(R) = \{[x^a y^b] \phi(x, y) \mid (a, b) \in R\}.$$

The following lemmas tell us how to compute the coefficients in $\text{Coeff}_{\phi}(R)$ from the knowledge of the coefficients in the neighborhood.

Lemma 3.7. *Let be $R \in SQ(\overline{R}) \setminus SQ_Z(\overline{R})$. If there exists $T \in \{N, W, S, E\}$ such that $\text{Coeff}_{\phi}(T(R))$ is known, then $\text{Coeff}_{\phi}(R)$ can be computed in time $O(1)$.*

PROOF. Suppose that we know $\text{Coeff}_{\phi}(E(R))$, that is, the set $\text{Coeff}_{\phi}(R(l-e, m))$. Then it is immediate to obtain $\text{Coeff}_{\phi}(R(l-e+1, m))$ by computing only e coefficients in $\text{Coeff}_{\phi}(R(l-e+1, m)) \setminus \text{Coeff}_{\phi}(R(l-e, m))$: this can be easily done with e^2 arithmetical operations (in order to get $c_{l-e+1, m-i+1}$ we use the recurrence \mathcal{N} and the values of the i -th row of $\text{Coeff}_{\phi}(R(l-e, m))$). Then, for $i = 2 \dots e$, we compute $\text{Coeff}_{\phi}(R(l-e+i, m))$ from $\text{Coeff}_{\phi}(R(l-e+i-1, m))$. Since there are e steps with cost $O(e^2)$, the overall computation requires time $O(e^3) = O(1)$. The other cases are similar. \square

Lemma 3.8. *Let $R = R(l, m) \in SQ_Z(\overline{R})$. If $\text{Coeff}_{\phi}(W(R))$, $\text{Coeff}_{\phi}(SW(R))$ and $\text{Coeff}_{\phi}(S(R))$ are known, then $\text{Coeff}_{\phi}(R)$ can be computed in time $O(1)$.*

PROOF. We define an ordering \prec on the set $\text{Coeff}_{\phi}(R)$ as follows: $c_{\alpha\beta} \prec c_{\gamma\delta}$ iff $\beta < \delta$ or $\beta = \delta$ and $\alpha \leq \gamma$. Then, we can compute the coefficients according to the \prec ordering, starting with $\min(\text{Coeff}_{\phi}(R))$ and using equation \mathcal{B} of the instance (see Equation (3.3) in the Coefficient Problem definition). At each step we compute one coefficient with e^2 arithmetical operations. Since we have e^2 coefficients, the total time is $O(e^4) = O(1)$. \square

The transitive closure of \diamond defines an equivalence relation $\diamond^* \subseteq SQ_Z(\overline{R})^2$, i.e. it defines a partition of $SQ_Z(\overline{R})$ into equivalence classes that we call *clusters*. More precisely, let be

$$\diamond_{\overline{R}} = \diamond \cap (SQ_Z(\overline{R}) \times SQ_Z(\overline{R}))$$

and consider the following:

Definition 3.9 (Cluster). The cluster generated by $R \in SQ_Z(\overline{R})$ is

$$\text{Cl}_{\overline{R}}^R = [R]_{\diamond_{\overline{R}}^*} = \{Q \in SQ_Z(\overline{R}) \mid Q \diamond_{\overline{R}}^* R\}.$$

It is then immediate to observe that it holds the following partition

$$SQ_Z(\overline{R}) = \bigcup_{h=1}^k \text{Cl}_{\overline{R}}^{R_h}$$

with $R_h \in SQ_Z(\overline{R})$ and $R_{h_1} \not\phi_{\overline{R}}^* R_{h_2}$.

Example 3.10. Let us consider the function $\phi(x, y) = \frac{1}{1-x^2y-xy^3}$ and the recurrences

$$\begin{aligned} N &= (4n^3 - 4n^2k - 30n^2 - 7nk^2 + 50n - 5nk + 25k + 5k^2 - 2k^3) E_n^5 + \\ &\quad (27n^3 - 135n^2 - 27n^2k + 90nk + 150n + 9nk^2 - k^3 - 50k - 15k^2) E_n^0 \\ K &= (10n + nk + 6n^2 - k^2 + 5k) E_k^5 - (4k^2 + 4nk - 5n - n^2 + 10k) E_k^0 \end{aligned}$$

associated with it. Let be $\overline{R} = R^{(5)}(59, 59)$ and $R = R^{(5)}(4, 4)$. The graphical representation of the cluster $\text{Cl}_R^{\overline{R}}$ is given in Figure 2.

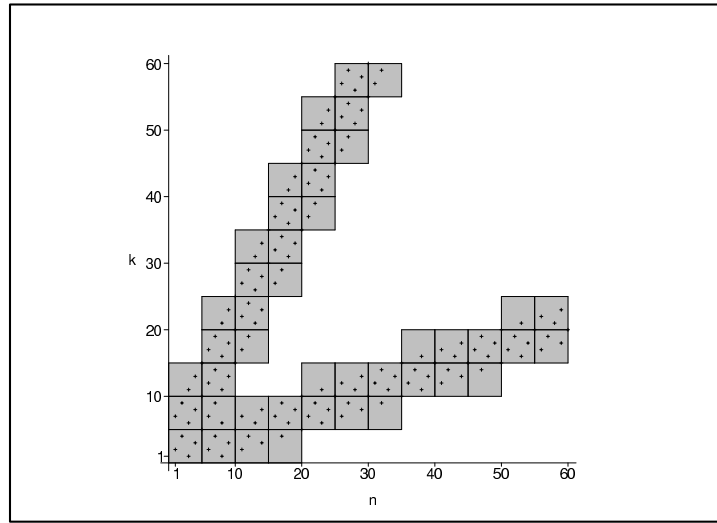


FIGURE 2. A cluster associated with $\phi(x, y) = \frac{1}{1-x^2y-xy^3}$. Dots are elements of $Z(\mathcal{N}, \mathcal{K})$.

Given a cluster $\text{Cl}_R^{\overline{R}}$ we define its *border* as the set

$$B(\text{Cl}_R^{\overline{R}}) = \{R' \in \text{SQ}(\overline{R}) \setminus \text{SQ}_Z(\overline{R}) \mid \exists R'' \in \text{Cl}_R^{\overline{R}} \text{ s.t. } R' \diamond_{\overline{R}} R''\}.$$

It is immediate to observe that $\sharp B(\text{Cl}_R^{\overline{R}}) = O(\sharp \text{Cl}_R^{\overline{R}}) = O(\overline{\tau} + \overline{\eta})$.

3.2. The algorithm. As shown in the previous section, an instance $\langle \mathcal{N}, \mathcal{K}, \mathcal{B}, I, i, j \rangle$ univocally identifies a set Z , an integer e , a square R_0 at the origin and a square $\overline{R} = N^{\lfloor j/e \rfloor}(E^{\lfloor i/e \rfloor}(R_0))$ containing the point (i, j) . We compute the coefficient $[x^i y^j] \phi(x, y)$ through a procedure that starts with $\text{Coeff}_\phi(R_0) \subseteq I$ and halts having computed $\text{Coeff}_\phi(\overline{R})$ after $O(i + j)$ steps.

Informally, the procedure works by computing a main sequence

$$\begin{aligned} &\text{Coeff}_\phi(E^0(R_0)), \text{Coeff}_\phi(E^1(R_0)), \dots, \text{Coeff}_\phi(E^{\lfloor i/e \rfloor}(R_0)), \\ &\text{Coeff}_\phi(N^1(E^{\lfloor i/e \rfloor}(R_0))), \text{Coeff}_\phi(N^2(E^{\lfloor i/e \rfloor}(R_0))) \dots, \text{Coeff}_\phi(\overline{R}). \end{aligned}$$

The first $\lfloor i/e \rfloor + 1$ sets of coefficients are easily computed in time $O(i)$. Then, we compute each set $\text{Coeff}_\phi(N^k(E^{\lfloor i/e \rfloor}(R_0)))$ having as input $\text{Coeff}_\phi(N^{k-1}(E^{\lfloor i/e \rfloor}(R_0)))$ according to the following rule: if $N^k(E^{\lfloor i/e \rfloor}(R_0)) \in \text{SQ}(\overline{R}) \setminus \text{SQ}_Z(\overline{R})$ then $\text{Coeff}_\phi(N^k(E^{\lfloor i/e \rfloor}(R_0)))$ is computed as shown in Lemma 3.7, otherwise all the coefficients associated with the cluster $\text{Cl}_{N^k(E^{\lfloor i/e \rfloor}(R_0))}^{\overline{R}}$ are computed in a suitable order.

In Figure 3 we define a procedure $\text{COEFF}(i, j)$ that has as input two positive integers i, j and returns the value $[x^i y^j] \phi(x, y)$. In the code a procedure $\text{COMPUTE}_{\diamond}(R_{out}, R_{in})$ is called. It takes as input two squares

such that $R_{out} = T(R_{in})$, with $T \in \{N, E, S, W\}$, and computes the set $\text{Coeff}_\phi(R_{out})$ under the assumption that $\text{Coeff}_\phi(R_{in})$ has been previously computed.

Procedure COEFF(i, j)

Begin

$R_0 := R(e-1, e-1)$; $c_1 := \lfloor i/e \rfloor$; $c_2 := \lfloor j/e \rfloor$;

For k **from** 1 **to** c_1 **do**

 compute $\text{Coeff}_\phi[E^k(R_0)]$ from $\text{Coeff}_\phi[E^{k-1}(R_0)]$

 by using the equation \mathcal{N} and the initial conditions I ;

For k **from** 1 **to** c_2 **do**

if $N^k(E^{c_1}(R_0)) \notin \text{SQ}_Z(\overline{R})$

then compute $\text{Coeff}_\phi[N^k(E^{c_1}(R_0))]$ by using equation \mathcal{K} and $\text{Coeff}_\phi[N^{k-1}(E^{c_1}(R_0))]$;

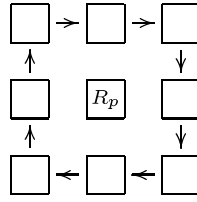
else COMPUTE $_{\circlearrowleft}(N^k(E^{c_1}(R_0)), N^{k-1}(E^{c_1}(R_0)))$;

return $[x^i y^j] \phi(x, y)$ from $\text{Coeff}_\phi[N^{c_2}(E^{c_1}(R_0))]$;

End;

FIGURE 3. Procedure COEFF

Both procedures are supposed to use two global variables: a suitable data structure for the sets $\text{Coeff}_\phi(R)$ and an integer variable e (the size of the edge of a square). As we note, the core of the algorithm consists of the procedure COMPUTE $_{\circlearrowleft}(R_{out}, R_{in})$. This procedure computes $\text{Coeff}_\phi(R_{out})$ starting from $\text{Coeff}_\phi(R_{in})$ and moving clockwise by using coefficients previously computed. In the code we find an indexed function $\text{next}_{R_p}(R)$: this is used to identify the square R' that is neighbor to R_p and follows R (clockwise). More formally:

$$\text{next}_{R_p}(R) \begin{cases} NE(R_p) & \text{if } R = N(R_p) \\ E(R_p) & \text{if } R = NE(R_p) \\ SE(R_p) & \text{if } R = E(R_p) \\ S(R_p) & \text{if } R = SE(R_p) \\ SW(R_p) & \text{if } R = S(R_p) \\ W(R_p) & \text{if } R = SW(R_p) \\ NW(R_p) & \text{if } R = W(R_p) \\ N(R_p) & \text{if } R = NW(R_p) \end{cases}$$


Function *next*

As an example, suppose we call COMPUTE $_{\circlearrowleft}(R, S(R))$, that is, we want to compute $\text{Coeff}_\phi(R)$ knowing $\text{Coeff}_\phi(S(R))$. So, if $R \in \text{SQ}_Z(\overline{R})$ then $\text{Coeff}_\phi(SW(R))$ and $\text{Coeff}_\phi(W(R))$ are needed, as shown in Lemma 3.8. Hence, the procedure advances clockwise around R , in order to get (recursively) $\text{Coeff}_\phi(SW(R))$ from $\text{Coeff}_\phi(S(R))$ and then $\text{Coeff}_\phi(W(R))$ from $\text{Coeff}_\phi(SW(R))$.

Figure 4 shows the procedure COMPUTE $_{\circlearrowleft}$; a simple example of computation is sketched in Figure 5, while in Figure 6 a real computation is shown.

4. Complexity

It is straightforward to see that COEFF(i, j) computes $[x^i y^j] \phi(x, y)$ if and only if every call COMPUTE $_{\circlearrowleft}(N^k(E^{c_1}(R_0)), N^k(E^{c_1}(R_0)))$ terminates and computes $\text{Coeff}_\phi[N^k(E^{c_1}(R_0))]$.

Hence, the problem is to analyse which sets of coefficients are computed by the recursive procedure COMPUTE $_{\circlearrowleft}$.

Note that a call COMPUTE $_{\circlearrowleft}(R_{out}, R_{in})$ recursively calls itself if and only if $R_{out} \in \text{SQ}_Z(\overline{R})$. So, let be $\text{Out}_0 = N^k(E^{c_1}(R_0))$, $\text{In}_0 = N^{k-1}(E^{c_1}(R_0))$ for a suitable integer $k \leq c_2$ and consider the sequence of calls

$$\text{COMPUTE}_{\circlearrowleft}(\text{Out}_0, \text{In}_0), \dots, \text{COMPUTE}_{\circlearrowleft}(\text{Out}_l, \text{In}_l)$$


```

Procedure COMPUTE $\circlearrowleft$ ( $R_{out}, R_{in}$ )
Begin
  While Undef(Coeff $\phi$ [ $S(R_{out})$ ]) or Undef(Coeff $\phi$ [ $SW(R_{out})$ ]) or Undef(Coeff $\phi$ [ $W(R_{out})$ ]) do
     $R' := next_{R_{out}}(R_{in});$ 
    if Undef(Coeff $\phi$ [ $R'$ ]) then
      if  $R' \notin SQ_Z(\bar{R})$  then compute Coeff $\phi$ [ $R'$ ] by using  $\mathcal{N}$  or  $\mathcal{K}$  and Coeff $\phi$ [ $R_{in}$ ];
      else COMPUTE $\circlearrowleft$ ( $R', R_{in}$ );

     $R_{in} := R';$ 
  EndWhile
  compute Coeff $\phi$ [ $R_{out}$ ] by using  $\mathcal{B}$  and Coeff $\phi$ [ $S(R_{out})$ ], Coeff $\phi$ [ $SW(R_{out})$ ], Coeff $\phi$ [ $W(R_{out})$ ];
End;

```

FIGURE 4. Procedure COMPUTE \circlearrowleft .

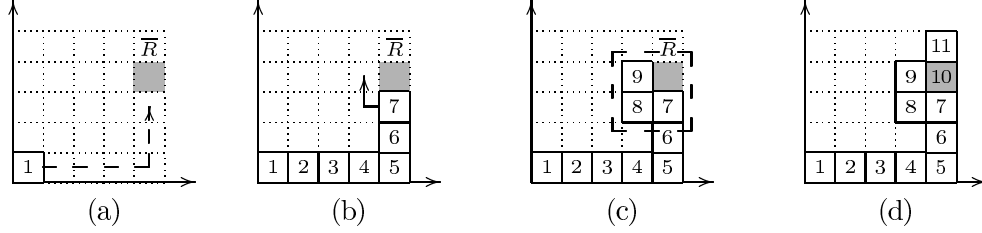


FIGURE 5. A run of COEFF. Squares are numbered with respect to the order of computation. The gray square is in $SQ_Z(\bar{R})$; in order to compute it, COMPUTE \circlearrowleft moves clockwise until South, West and South-West neighbors have been computed.

contained in the stack associated with the call COMPUTE \circlearrowleft (Out $_0$, In $_0$) (at the bottom).

For each $0 \leq p < l$, let Step $_p = R_{p_1}, \dots, R_{p_h}$ be the 4-connected sequence of squares adjacent to Out $_p$ such that

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

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

In the sequel, we consider three sequences Seq, $\widetilde{\text{Seq}}$ and $\widehat{\text{Seq}}$, associated with the stack and defined as follows:

- Seq = Out $_0, \dots, \text{Out}_l$ is the 8-connected sequence of squares in $SQ_Z(\bar{R})$ such that Coeff ϕ (Out $_i$) is not known ($0 \leq i \leq l$).
- $\widetilde{\text{Seq}} = R_0, E(R_0), \dots, E^{c_1}(R_0), N(E^{c_1}(R_0)), \dots, N^{k-1}(E^{c_1}(R_0))$ is the 4-connected ascending sequence of $c_1 + k$ squares such that Coeff ϕ (R) is known, $R \in \widetilde{\text{Seq}}$.
- $\widehat{\text{Seq}} = \text{Step}_0, \dots, \text{Step}_{l-1}, \text{In}_l$ is the 4-connected sequence such that for all $R \in \widehat{\text{Seq}}$, Coeff ϕ (R) has been computed by recursive calls to COMPUTE \circlearrowleft .

We analyse which sets of coefficients are computed by COMPUTE \circlearrowleft by proving the following:

Lemma 4.1. *Let COMPUTE \circlearrowleft ($N^k(E^{c_1}(R_0)), N^{k-1}(E^{c_1}(R_0))$) be a call occurring in COEFF. Then, for all the calls COMPUTE \circlearrowleft (R_{out}, R_{in}) that are pushed onto the stack we have*

$$R_{out} \in Ct_{N^k(E^{c_1}(R_0))}^{N^k(E^{c_1}(R_0))}.$$

PROOF. Let be Out $_0 = N^k(E^{c_1}(R_0))$ and let Seq, $\widetilde{\text{Seq}}$ and $\widehat{\text{Seq}}$ be the sequences associated with the stack having the call COMPUTE \circlearrowleft (Out $_0, S(\text{Out}_0))$ at the bottom. We show that for all Out $_i$ in Seq we

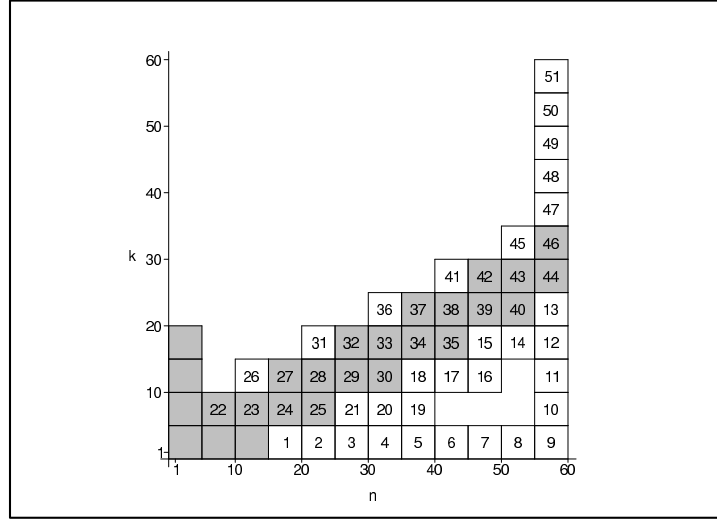


FIGURE 6. Running COEFF(59,59) for the function $\phi(x, y) = \frac{1}{1-x^2y-xy^3}$ (Example 3.10). Squares that are not numbered belong to the set of initial conditions.

have $\text{Out}_i \in \text{Cl}_{\text{Out}_0}^{\text{Out}_0}$, that is, $\text{Out}_i \diamond_{\text{Out}_0}^* \text{Out}_0$. Hence, since for $0 \leq i < l$ it holds $\text{Out}_i \diamond \text{Out}_{i+1}$, it is sufficient to prove that

$$(4.1) \quad \text{Out}_i \in \text{SQ}_Z(\text{Out}_0)$$

Observe that $\{\text{Seq}\} \cap \{\widehat{\text{Seq}}\} = \{\text{Seq}\} \cap \{\widetilde{\text{Seq}}\} = \emptyset$ and note that we can univocally identify g sequences Seq_i ($1 \leq i \leq g$) such that $\text{Seq} = \text{Seq}_1, \text{Seq}_2, \dots, \text{Seq}_g$ with

- Seq_1 is the longest descending sequence that appears at the beginning of Seq
- Seq_{2i} is the longest ascending sequence after $\text{Seq}_1, \text{Seq}_2, \dots, \text{Seq}_{2i-1}$, $1 < 2i \leq g$
- Seq_{2i+1} is the longest descending sequence after $\text{Seq}_1, \text{Seq}_2, \dots, \text{Seq}_{2i}$, $1 < 2i + 1 \leq g$

We prove Property (4.1) by induction on the number g of ascending or descending sequences in the decomposition of Seq shown above.

BASIS: Seq consists of one descending sequence $\text{Out}_0, \dots, \text{Out}_l$, where $\text{Out}_i = E^{w_i}(W^{v_i}(S^{u_i}(\text{Out}_0)))$ and either $\text{Out}_1 = W(\text{Out}_0)$ or $\text{Out}_1 = SW(\text{Out}_0)$. So, it trivially holds that $\text{Out}_i \in \text{SQ}_Z(\text{Out}_0)$ if $v_i > w_i$, $1 \leq i \leq l$. By absurd, let be $\bar{i} = \min\{i \mid v_i < w_i\}$ (note that it must be $v_{\bar{i}} \neq w_{\bar{i}}$ since $\{\text{Seq}\} \cap \{\widetilde{\text{Seq}}\} = \emptyset$). So, we would have $v_{\bar{i}-1} > w_{\bar{i}-1}$ and $v_{\bar{i}} < w_{\bar{i}}$: this would imply that $\text{Out}_{\bar{i}-1} \not\sim \text{Out}_{\bar{i}}$.

INDUCTION: $\text{Seq} = \text{Seq}_1, \text{Seq}_2, \dots, \text{Seq}_{n-1}, \text{Seq}_n$. By induction we know that all the squares in $\text{Seq}_1, \text{Seq}_2, \dots, \text{Seq}_{n-1}$ satisfy Property (4.1). Let be $\text{Seq}_n = \text{Out}_s, \dots, \text{Out}_l$ and let Out_{s-1} be the last square of Seq_{n-1} . We distinguish two cases.

n IS ODD: Seq_n is a descending sequence. By induction we know that $\text{Out}_{s-1} \in \text{SQ}_Z(\text{Out}_0)$, that is, $\text{Out}_{s-1} = W^{\alpha_{s-1}}(S^{u_{s-1}}(\text{Out}_0))$ with $\alpha_{s-1}, u_{s-1} \in \mathbb{N}$, $\alpha_{s-1} > 0$. Since $\text{Out}_s = T(\text{Out}_{s-1})$ with $T \in \{SW, S, SE\}$ it holds $\text{Out}_s = W^{\alpha_s}(S^{u_{s-1}+1}(\text{Out}_0))$ with $\alpha_s \geq 0$. Again, $\alpha_s \neq 0$ since $\{\text{Seq}\} \cap \{\widetilde{\text{Seq}}\} = \emptyset$. Now, the same analysis done for the basis shows that Property 4.1 holds for the squares of Seq_n .

n IS EVEN: Seq_n is an ascending sequence, that is, $\text{Out}_s = T(\text{Out}_{s-1})$, $T \in \{NW, N, NE\}$. We claim that $\text{Out}_s = NE(\text{Out}_{s-1})$. In fact, recall that each sequence Step_i of squares examined by $\text{COMPUTE}_{\circlearrowleft}(\text{Out}_i, \text{In}_i)$

before calling $\text{COMPUTE}_{\circ}(\text{Out}_{i+1}, \text{In}_{i+1})$ is 4-connected. This means that the sequence $\text{Out}_{s-1}, \text{In}_{s-1}$ is 4-connected, that is, $\text{In}_{s-1} = T(\text{Out}_{s-1})$ with $T = \{N, E, S, W\}$. In particular, note that $\text{In}_{s-1} = N(\text{Out}_{s-1})$ since in the other three cases the call $\text{COMPUTE}_{\circ}(\text{Out}_{s-1}, \text{In}_{s-1})$ would compute $\text{Coeff}_{\phi}(\text{Out}_{s-1})$ without any recursion.

Therefore, $\text{COMPUTE}_{\circ}(\text{Out}_{s-1}, \text{In}_{s-1})$ recursively calls $\text{COMPUTE}_{\circ}(NE(\text{Out}_{s-1}), \text{In}_s)$ with $\text{In}_s = N(\text{Out}_{s-1}) \in \widehat{\text{Seq}}$.

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

$$\widehat{\widehat{\text{Seq}}} = R_0, E(R_0), \dots, E^{c_1}(R_0), N(E^{c_1}(R_0)), \dots, N^{k-1}(E^{c_1}(R_0)), \text{Step}_0, \dots, \text{Step}_{s-1}, \text{In}_s.$$

We trivially have $\{\widehat{\widehat{\text{Seq}}}\} \cap \{\text{Seq}_n\} = \emptyset$. In fact, the value $\text{Coeff}_{\phi}[R]$ is defined if $R \in \widehat{\widehat{\text{Seq}}}$ and undefined if $R \in \text{Seq}_n$. Informally, this means that the squares of the ascending sequence Seq_n are restricted to lie in a closed area (delimited by $\widehat{\widehat{\text{Seq}}}$) consisting of squares that satisfy Property (4.1). □

An immediate consequence of the previous lemma is:

Corollary 4.2. *Let be $R_k = N^k(E^{c_1}(R_0)) \in \text{SQ}_Z(\overline{R})$. If $\text{Coeff}_{\phi}(R)$ is computed by a call $\text{COMPUTE}_{\circ}(R_k, S(R_k))$ occurring in COEFF then*

$$R \in B(\text{Cl}_{R_k}^{R_k}) \cup \text{Cl}_{R_k}^{R_k}.$$

PROOF. By inspecting the code of COMPUTE_{\circ} we note that for each computed set $\text{Coeff}_{\phi}(R)$, either $R \in \text{SQ}_Z(\overline{R})$ (and $\text{COMPUTE}_{\circ}(R, R_{in})$ is a call generated by $\text{COMPUTE}_{\circ}(R_k, S(R_k))$) or $R \notin \text{SQ}_Z(\overline{R})$ (and $\text{Coeff}_{\phi}(R)$ is computed by a recursive call $\text{COMPUTE}_{\circ}(R_{out}, Q)$ generated by $\text{COMPUTE}_{\circ}(R_k, S(R_k))$ such that $R \diamond R_{out}$).

In the first case Lemma 4.1 states that $R \in \text{Cl}_{R_k}^{R_k}$ while in the second we have $R \in B(\text{Cl}_{R_k}^{R_k})$. □

Lemma 4.3. *Let St be the stack associated with a call $\text{COMPUTE}_{\circ}(R, S(R))$ occurring in COEFF. Then, St does not contain two identical calls.*

PROOF. (By contradiction) Let $\text{COMPUTE}_{\circ}(\text{Out}_h, \text{In}_h)$ be the first repeated occurrence of a call, that is, $h = \min\{0 \leq i \leq l \mid \exists \delta > 0, \text{Out}_i = \text{Out}_{i-\delta} \wedge \text{In}_i = \text{In}_{i-\delta}\}$. Without loss of generality, we suppose that $\text{In}_h = W(\text{Out}_h)$. Consider the 8-connected sequence S that is a subsequence of Seq ,

$$S = \text{Out}_{h-\delta}, \text{Out}_{h-\delta+1}, \dots, \text{Out}_h,$$

together with the 4-connected subsequence of $\widehat{\text{Seq}}$,

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

We recall that for $R \in \widehat{S}$ the set $\text{Coeff}_{\phi}(R)$ is known and that for each $R \in \widehat{S}$ ($R \in S$) there exists $Q \in S$ ($Q \in \widehat{S}$) such that $R \diamond Q$. Note that both sequences are “closed”, that is, their first and last squares coincide. Moreover, we have $\{S\} \cap \{\widehat{S}\} = \emptyset$.

Let be $R_0 = R(e-1, e-1)$ and for each closed sequence S denote by $\text{Inside}(S)$ the set of all the squares in $\text{SQ}(\overline{R})$ that lie in the area surrounded by S . Then, it is immediate to observe that we have only two cases:

$\widehat{S} \subseteq \text{Inside}(S)$: This means that if $R \in \widehat{S}$ it is impossible to find a 4-connected sequence $T_R = R_0, \dots, R$ such that $\{T_R\} \cap \{S\} = \emptyset$. On the other hand, we know that for every $R \in \widehat{S}$ there exists a 4-connected sequence T_R from R_0 to R , consisting of squares in $\text{SQ}(\overline{R})$, such that for Q in T_R the set $\text{Coeff}_{\phi}(Q)$ has been computed (see the sequence $\widehat{\widehat{\text{Seq}}}$ in the proof of Lemma 4.1). Therefore, we have $\widehat{S} \not\subseteq \text{Inside}(S)$.

$S \subseteq \text{Inside}(\widehat{S})$: Let be $k_1, k_2 \in \mathbb{N}$ such that

$$\begin{cases} N^{k_1}(E^{k_2}(R_0)) \in S \\ N^{h_1}(E^{h_2}(R_0)) \in S \Rightarrow k_1 + k_2 \leq h_1 + h_2 \end{cases}$$

Let be $\text{Out}_{\bar{h}} = N^{k_1}(E^{k_2}(R_0))$. Since $S \subseteq \text{Inside}(\widehat{S})$, it is immediate to prove that $S(\text{Out}_{\bar{h}})$ and $W(\text{Out}_{\bar{h}})$ belong to \widehat{S} . More precisely, because \widehat{S} is 4-connected, it follows that

$$\widehat{S} = \text{In}_{\bar{h}}, \dots, S(\text{Out}_{\bar{h}}), SW(\text{Out}_{\bar{h}}), W(\text{Out}_{\bar{h}}), \dots, \text{In}_{\bar{h}}.$$

Then, the call that computes $\text{Coeff}_{\phi}(SW(\text{Out}_{\bar{h}}))$ must be $\text{COMPUTE}_{\circ}(\text{Out}_{\bar{h}}, \text{In}_{\bar{h}})$. By observing the code of COMPUTE_{\circ} , we note that if $\text{COMPUTE}_{\circ}(\text{Out}_{\bar{h}}, \text{In}_{\bar{h}})$ computes $\text{Coeff}_{\phi}(SW(\text{Out}_{\bar{h}}))$ then it has previously computed $\text{Coeff}_{\phi}(S(\text{Out}_{\bar{h}}))$ and it necessarily computes also $\text{Coeff}_{\phi}(W(\text{Out}_{\bar{h}}))$. So, this call would terminate without any recursion. \square

The following lemma states that we can develop a suitable data structure for storing all the coefficients that are computed by the algorithm. More precisely, we have:

Lemma 4.4. *The data structure $\text{Coeff}_{\phi}[]$ can be implemented in space $O(i + j)$ and accessed in time $O(1)$.*

PROOF. $\text{Coeff}_{\phi}[]$ can be easily implemented as a dynamic data structure representing a graph. We give here an outline for such implementation.

Let be

$$d = \max\{\deg_n(p_r(n, k)), \deg_n(p_0(n, k)), \deg_n(q_s(n, k)), \deg_n(q_0(n, k))\}$$

the integer univocally associated with an instance $\langle \mathcal{N}, \mathcal{K}, \mathcal{B}, I, i, j \rangle$ and let be $\zeta = 4d(e + 1)$. Note that for every integer k we have $\#\{N^k(E^h(R(e - 1, e - 1))) \in \text{SQ}_{\mathcal{Z}}(\bar{R})\} \leq \zeta$. Since we have to consider also squares that belong to the border of a cluster, for each k we have at most 9ζ squares $R_{kh} = N^k(E^h(R(e - 1, e - 1)))$ such that $\text{Coeff}_{\phi}(R_{kh})$ is computed. So, an immediate implementation for the sets $\text{Coeff}_{\phi}()$ is based on a list of lists. More precisely, we have a primary double linked list whose length is $\lfloor j/e \rfloor + 1$. The k -th node of this list contains a link to the list for the sets $\text{Coeff}_{\phi}(R_{kh})$: this is a list whose length is less or equal to 9ζ . Then, it is immediate to note that we access to $\text{Coeff}_{\phi}[R_{kh}]$ in constant time if the procedure $\text{COMPUTE}_{\circ}(R_{kh}, \text{In})$ is equipped with a suitable link to the k -th node of the main list. \square

Theorem 4.5. *The total number of calls to COMPUTE_{\circ} during the execution of $\text{COEFF}(i, j)$ is $O(i + j)$.*

PROOF. Recall that $\bar{R} = R(\bar{i}, \bar{j}) = N^{\lfloor j/e \rfloor}(E^{\lfloor i/e \rfloor}(R(e - 1, e - 1)))$ and let

$$\text{COMPUTE}_{\circ}(R_1, S(R_1)), \dots, \text{COMPUTE}_{\circ}(R_t, S(R_t))$$

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

$$\text{TOT} = \{\text{COMPUTE}_{\circ}(R, T(R)) \mid R \in \text{SQ}_{\mathcal{Z}}(\bar{R}), T \in \{N, E, S, W\}\},$$

and, for $1 \leq k \leq t$,

$$\text{TOT}_k = \{C \in \text{TOT} \mid C \text{ is a recursive call originated by } \text{COMPUTE}_{\circ}(R_k, S(R_k))\}.$$

Note that $\text{COMPUTE}_{\circ}(R_k, S(R_k))$ recursively generates calls of type $\text{COMPUTE}_{\circ}(R, Q)$ with $R \in \text{Cl}_{R_k}^{R_k}$, such that $\text{Coeff}_{\phi}(R)$ has not been previously computed by $\text{COMPUTE}_{\circ}(R_l, S(R_l))$ with $1 \leq l < k$. In other words, $\text{TOT}_l \cap \text{TOT}_m = \emptyset$, for $l \neq m$.

Lemma 4.3 guarantees that the number of recursive calls generated by $\text{COMPUTE}_{\circ}(R_k, S(R_k))$ is exactly $\#\text{TOT}_k$. Hence, recalling Lemma 3.6, the total number of calls is

$$\sum_{k=1}^t \#\text{TOT}_k = \#\bigcup_{k=1}^t \text{TOT}_k \leq \#\text{TOT} = 4 \cdot \#\text{SQ}_{\mathcal{Z}}(\bar{R}) = O(\bar{i} + \bar{j}) = O(i + j)$$

□

At last, we have:

Theorem 4.6. *COEFF(i, j) runs in time $O(i + j)$ and in space $O(i + j)$.*

PROOF. By Th. 5.4 we know that procedure COMPUTE_○ is called $O(i + j)$ times. By inspecting the code we note that each call consists of a constant number of operations because the cost of accessing $\text{Coeff}_\phi[]$ is $O(1)$ (see Lemma 5.3). Moreover, the space requirement is bounded by the sum of the maximum stack size and the size of the data structure for $\text{Coeff}_\phi[]$. So, we conclude that $\text{COEFF}(i, j)$ runs in time $O(i + j)$ using $O(i + j)$ space. □

5. Conclusions

In this paper we have presented an algorithm that computes the coefficient $[x^i y^j] \phi(x, y)$ of a rational formal series $\phi(x, y)$ working in time and space $O(i + j)$ under the uniform cost criterion. If we adopt the logarithmic cost criterion, we expect that the complexity of the algorithm becomes $O((i + j)^2)$, since the growing of the coefficients $[x^n y^k] \phi(x, y)$ is at most exponential (i.e. the cost of a single arithmetical operation is at most linear).

Two remarks are worthwhile with respect to such algorithm: the first is related to the computation of the recurrences, the second deals with the size e of the squares. We pointed out that the recurrences can be obtained through an elimination process in a noncommutative algebra. Actually, in order to compute a Gröbner Basis in the shift algebra $\mathbb{Q}\langle n, k, E_n, E_k \rangle$, we took advantage of the package ‘Mgfun’, running under Maple and implemented by Chyzak ([5]). This step can be quite expensive and so it would be interesting to look for a method that directly computes the recurrences from the linear representation of a rational series.

With respect to the size e , let us show an upper bound for its value. Consider a rational function $\phi = \frac{P(x, y)}{Q(x, y)}$ and let d_x (d_y) be the degree of $P \cdot Q$ in x (y). A system of independent recurrence equations is obtained by converting the holonomic system

$$\begin{cases} (\partial_x P Q - P \partial_x Q) \partial_x \phi & = 0 \\ (\partial_y P Q - P \partial_y Q) \partial_x \phi + (\partial_x P Q - P \partial_x Q) \partial_y \phi & = 0 \end{cases}$$

into operators of the shift algebra. The degree in E_n and E_k of such operators is respectively d_x and d_y . By applying the Zeilberger’s elimination algorithm ([14]), we obtain operators depending either on E_n or on E_k . Their degrees in E_n and E_k are at most quadratic with respect to d_x and d_y (see Section 5.3 in [14]). Then, we have $e = O((\max\{d_x, d_y\})^2)$.

Last but not least, it might be interesting to study whether the technique we have presented can be modified in order to deal with a number of variables greater than two. We point out that the straightforward extension of this method to the 3-D case does not work (in the 3-D case the set of the values that the algorithm considers as initial conditions has not size $O(1)$). Moreover, as our method is related to the theory of the holonomic series, it would be useful to generalize it in order to get an efficient algorithm for the Coefficient Problem for holonomic series.

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., vol. 5 (1971), pp. 1–16 (russian); pp. 89–101 (english).
- [2] A. Bertoni, P. Massazza, R. Radicioni, *Random generation of words in regular languages with fixed occurrences of symbols*. Proceedings of WORDS’03, 4th International Conference on Combinatorics of Words, Turku, Finland, 2003, pp. 332–343.
- [3] J. Berstel, C. Reutenauer, *Rational Series and Their Languages*. Springer-Verlag, EATCS Monographs on Theoretical Computer Science, vol. 12 (1984).
- [4] M. Bousquet-Mélou, M. Petkovšek, *Linear recurrences with constant coefficients: the multivariate case*. Discrete Mathematics **225** (2000), pp. 51–75.

- [5] F. Chyzak, *An Extension of Zeilberger's Fast Algorithm to General Holonomic Functions*. Proceedings of FPSAC'97, Universität Wien, 1997, pp. 172–183.
- [6] F. Chyzak, B. Salvy, *Non-commutative Elimination in Ore Algebras Proves Multivariate Identities*. Journal of Symbolic Computation **2**, vol.26 (1998), pp. 187–227.
- [7] A. Denise, *Génération aléatoire uniforme de mots de langages rationnels*. Theoretical Computer Science **159** (1996), pp. 43–63.
- [8] A. Denise, O. Roques, M. Termier, *Random generation of words of context-free languages according to the frequencies of letters*. Mathematics and Computer Science: Algorithms, Trees, Combinatorics and Probabilities, Trends in Mathematics, Birkhäuser (2000), pp. 113–125. (Proceedings of Colloquium on Mathematics and Computer Science, Versailles, September 2000).
- [9] I. Dutour, J. M. Fédou, *Object grammars and random generation*. Discrete Mathematics and Theoretical Computer Science **2** (1998), pp. 47–61.
- [10] P. Flajolet, P. Zimmermann and B. Van Cutsem, *A calculus for the random generation of labelled combinatorial structures*. Theoretical Computer Science **132** (1994), pp. 1–35.
- [11] L. Lipshitz, *D-Finite Power Series*. Journal of Algebra **122** (1989), pp. 353–373.
- [12] E. Mansfield, *Differential Gröbner Bases*. Ph.D Thesis, University of Sidney (1991).
- [13] R.P. Stanley, *Differentiably Finite Power Series*. European Journal of Combinatorics **1** (1980), pp. 175–188.
- [14] D. Zeilberger, *A holonomic systems approach to special functions identities*. Journal of Computational and Applied Mathematics **32** (1990), pp. 321–368.

DIPARTIMENTO DI INFORMATICA E COMUNICAZIONE, UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA, VIA MAZZINI 5, 21100 VARESE, ITALY
E-mail address: paolo.massazza@uninsubria.it

DIPARTIMENTO DI SCIENZE DELL'INFORMAZIONE, UNIVERSITÀ DEGLI STUDI DI MILANO, VIA COMELICO 39, 20135 MILANO, ITALY
E-mail address: radicioni@dsi.unimi.it

An immediate consequence of the previous lemma is:

Corollary 5.1. *Let be $R(\bar{\tau}, ke) \in SQ_Z(\bar{R})$ and $x, y \in \mathbb{N}$ such that the set $Coeff_\phi(x, y)$ is computed by a call $COMPUTE_\circ(\bar{\tau}, ke, \bar{\tau}, (k-1)e)$ occurring in COEFF. Then*

$$R(x, y) \in B(CI_{R(\bar{\tau}, ke)}^{\bar{\tau}, ke}) \cup CI_{R(\bar{\tau}, ke)}^{\bar{\tau}, ke}.$$

PROOF. By inspecting the code of $COMPUTE_\circ$ we note that each recursive call occurs on input (x, y, z, t) such that $R(x, y)$ is either in the equivalence class of $R(\bar{\tau}, ke)$ (with respect to relation \diamond^*) or in its border. Lemma 4.1 bounds x and y properly. \square

Lemma 5.2. *Let $COMPUTE_\circ(l, m, l, m-e)$ be a call in COEFF. Then, the stack that has $COMPUTE_\circ(l, m, l, m-e)$ at the bottom does not contain two identical calls.*

PROOF. (By contradiction) Let us denote by C_h the call at level h in the stack. Without loss of generality, let $C_{\hat{h}} = COMPUTE_\circ(\hat{x}, \hat{y}, \hat{x} - e, \hat{y})$ be the first occurrence of a call that appears twice in the stack. Then, there exists $\delta > 0$ such that $C_{\hat{h}-\delta} = COMPUTE_\circ(\hat{x}, \hat{y}, \hat{x} - e, \hat{y})$. The portion of stack $C_{\hat{h}-\delta}, C_{\hat{h}-\delta+1}, \dots, C_{\hat{h}}$ identifies an 8-connected sequence

$$Seq = R(x_0, y_0), \dots, R(x_\delta, y_\delta) \quad (\text{with } R(x_0, y_0) = R(x_\delta, y_\delta) = R(\hat{x}, \hat{y}))$$

that has an associated 4-connected sequence

$$\widetilde{Seq} = R(\tilde{x}_0, \tilde{y}_0), \dots, R(\tilde{x}_\gamma, \tilde{y}_\gamma) \quad (\text{with } R(\tilde{x}_0, \tilde{y}_0) = R(\tilde{x}_\gamma, \tilde{y}_\gamma) = R(\hat{x} - e, \hat{y}))$$

such that $Coeff(\tilde{x}_j, \tilde{y}_j)$ is known, $0 \leq j \leq \gamma$. Note that Seq and \widetilde{Seq} are closed path and that $\{Seq\} \cap \{\widetilde{Seq}\} = \emptyset$. Let $Inside(P)$ denote the set of all the squares of SQ that lie in the area surrounded by a closed path P . Then, it is immediate to observe that we have only two cases:

$\widetilde{\text{Seq}} \subseteq \text{Inside}(\text{Seq})$: This means that for each $R(x, y) \in \widetilde{\text{Seq}}$ it is impossible to find a 4-connected sequence $T_{(x,y)} = R(e, e), \dots, R(x, y)$ such that $\{T_{(x,y)}\} \cap \{\text{Seq}\} = \emptyset$. On the other hand, we know that for every $R(x, y) \in \widetilde{\text{Seq}}$ there is a 4-connected sequence of squares $R(e, e), \dots, R(x, y)$ whose associated coefficients have been computed (see the definition of the sequence $\widetilde{\text{Seq}}$ in the proof of Lemma 4.1). We have then the contradiction $\widetilde{\text{Seq}} \not\subseteq \text{Inside}(\text{Seq})$.

$\text{Seq} \subseteq \text{Inside}(\widetilde{\text{Seq}})$: Since $\widetilde{\text{Seq}}$ is a closed 4-connected sequence, there must be $R(a, b) \in \widetilde{\text{Seq}}$ such that $\widetilde{\text{Seq}} = R(\tilde{x}_0, \tilde{y}_0), \dots, R(a, b), R(a - e, b), R(a - e, b + e), \dots, R(\tilde{x}_\gamma, \tilde{y}_\gamma)$ and $R(a, b + e) \in \text{Seq}$, $R(a, b + e) \in \text{SQ}_S$. This is impossible, because there would be in the stack a call $\text{COMPUTE}_\circ(a, b + e, \alpha, \beta)$ that would halt the recursion. \square

The following lemma states that we can develop a suitable data structure for storing all the coefficients that are computed by the algorithm. More precisely, we have:

Lemma 5.3. *Let $\langle \mathcal{N}, \mathcal{K}, \mathcal{B}, I, i, j \rangle$ be an instance of the Coefficient Problem associated with a rational series $\phi(x, y)$. Then, there exists a data structure G for the sets $\text{Coeff}_\phi(x, y)$ that are computed by $\text{COMPUTE}(i, j)$ such that G can be implemented in space $O(i + j)$ and access time $O(1)$.*

PROOF. G can be easily implemented as a dynamic data structure representing a graph. We give here an outline for such implementation.

Let us consider the system of recurrences $\{N, K\}$ and let be

$$d = \max\{\deg_n(p_r(n, k)), \deg_n(p_0(n, k)), \deg_n(q_s(n, k)), \deg_n(q_0(n, k))\}.$$

Note that for every integer $k \geq e, k \equiv_e 0$, there exists at most $\zeta = 4d(e + 1)$ values x_{k_i} such that $R(x_{k_i}, k) \in \text{SQ}_S(i, j)$, and $\text{Coeff}_\phi(x_{k_i}, k)$ is computed by the algorithm. Since we have to consider also squares that belong to borders, we have at most $\xi = \zeta + 2\zeta + 3 \cdot 2\zeta$ squares $R(x, k)$ such that $\text{Coeff}_\phi(x, k)$ is computed. So, an immediate implementation for the sets $\text{Coeff}_\phi(x, y)$ is based on a list of lists. More precisely, we have a primary double linked list whose length is $\lceil j/e \rceil$. The h -th node of this list contain a link to the list for the sets $\text{Coeff}_\phi(x, he)$: this is a list whose length is bounded by the constant ξ . Then, it is immediate to note that the access to the sets $\text{Coeff}_\phi(x, be)$ requires constant time if the procedure $\text{COMPUTE}_\circ(a, b, c, d)$ is equipped with a suitable link to the b -th node of the main list. \square

Theorem 5.4. *The total number of calls to COMPUTE_\circ during the execution of $\text{COEFF}(i, j)$ is $O(i + j)$.*

PROOF. Let $\text{COMP}(x, y, z, t)$ be the number of recursive calls generated by $\text{COMPUTE}_\circ(x, y, z, t)$. Define $\text{CALL} = \{(x, y, z, t) \in \mathbb{N}^4 \mid \text{COEFF}(i, j) \text{ calls } \text{COMPUTE}_\circ(x, y, z, t)\}$. We obviously have $\#\text{CALL} = O(i + j)$. Moreover, let be $\text{CALL}_1 = \{(x, y, z, t) \in \text{CALL} \mid R(x, y) \in \text{SQ}_S(i, j)\}$ and $\text{CALL}_2 = \{(x, y, z, t) \in \text{CALL} \mid R(x, y) \notin \text{SQ}_S(i, j)\}$.

We need to estimate

$$\sum_{\alpha \in \text{CALL}} \text{COMP}(\alpha) = \sum_{\alpha \in \text{CALL}_1} \text{COMP}(\alpha) + \sum_{\alpha \in \text{CALL}_2} \text{COMP}(\alpha).$$

We first note that $\sum_{\alpha \in \text{CALL}_2} \text{COMP}(\alpha) = O(i + j)$, then we partition CALL_1 according to the partition of $\text{SQ}_S(i, j)$ in clusters, obtaining

$$\text{CALL}_1 = \bigcup_{h=1}^k \text{CALL}(x_h, y_h, z_h, t_h)$$

where $\text{CALL}(x_h, y_h, z_h, t_h) = \{(x, y, z, t) \in \text{CALL}_1 \mid R(x, y) \diamond_{(i,j)}^* R(x_h, y_h)\}$ and for each $1 \leq r < s \leq h$, $R(x_r, y_r) \not\phi_{(i,j)}^* R(x_s, y_s)$.

Since the sets $\text{Coeff}_\phi(a, b)$ are computed once, recalling Lemma 4.3 and Corollary 4.2 we have

$$\sum_{\alpha \in \text{CALL}_1} \text{COMP}(\alpha) = \sum_{h=1}^k \sum_{\alpha \in \text{CALL}(x_h, y_h, z_h, t_h)} \text{COMP}(\alpha) = O\left(\sum_{h=1}^k \#\text{Cl}_R^{(i,j)}(x_h, y_h)\right) = O(i+j).$$

Hence, we conclude that

$$\sum_{\alpha \in \text{CALL}} \text{COMP}(\alpha) = O(i+j).$$

□

At last, we have:

Theorem 5.5. *COEFF(i, j) runs in time $O(i+j)$ and in space $O(i+j)$.*

PROOF. By Th. 5.4 we know that there are $O(i+j)$ calls to the procedure COMPUTE_\circ . By inspecting the code we note that each instance consists of a constant number of operations if accessing $\text{Coeff}_\phi(x, y)$ costs $O(1)$. Moreover, the space requirement is bounded by the sum of the maximum stack size and the size of the data structure for $\text{Coeff}_\phi(x, y)$. Recalling Theorem ?? and Lemma 5.3 we conclude that $\text{COEFF}(i, j)$ runs in time $O(i+j)$ using $O(i+j)$ space. □