Recurrence relations versus succession rules

S. Bilotta[*]   E. Pergola[*]   R. Pinzani[*]   S. Rinaldi[†]

Abstract

In this paper we present a method to pass from a recurrence relation having constant coefficients (in short, a C-recurrence) to a finite succession rule defining the same number sequence. We recall that succession rules are a recently studied tool for the enumeration of combinatorial objects related to the ECO method. We also discuss the applicability of our method as a test for the positivity of a number sequence.

keywords: Positivity problem; recurrence relation; succession rule.

1   Introduction

Succession rules (sometimes called *ECO-systems*) have been proved an efficient tool to solve several combinatorial problems. A *succession rule* Ω is a system constituted by an *axiom* (a), with $a \in \mathbb{N}$, and a set of *productions* of the form:

$$(k) \rightsquigarrow (e_1(k))(e_2(k))\ldots(e_k(k)), \qquad k \in \mathbb{N}, \ e_i : \mathbb{N} \to \mathbb{N}.$$

A production constructs, for any given label (k), its *successors* $(e_1(k)), (e_2(k)), \ldots, (e_k(k))$. The rule Ω can be represented by means of a *generating tree* having (a) as the label of the root and each node labelled (k) at level n produces k sons labelled $(e_1(k)), (e_2(k)), \ldots, (e_k(k))$, respectively, at level $n + 1$.

A succession rule Ω defines a sequence of positive integers $\{f_n\}_{n \geq 0}$ where f_n is the number of the nodes at level n in the generating tree defined by Ω. By convention the root is at level 0, so $f_0 = 1$. The function $f_\Omega(x) = \sum_{n \geq 0} f_n x^n$ is the *generating function* determined by Ω.

The concept of a succession rule was introduced in [9] by Chung et al. to study reduced Baxter permutations, and was later applied to the enumeration of permutations with forbidden subsequences [25].

Only later this has been recognized as an extremely useful tool for the ECO method, a methodology applied for the enumeration [6], random generation [4], or exhaustive generation [1] of various combinatorial structures.

More recently, succession rules have been considered as a remarkable object to be studied independently of their applications, and they have been treated by several points of view. In [2], Banderier et al. explore in detail the relationship between the form and the generating function of a succession rule, and then provide a classification of rules as *rational*, *algebraic*, or *transcendental*, according to their generating function type; besides some algebraic properties of succession rules – represented in terms of a *rule operator* – have been determined and studied in [13].

---
[*]Dipartimento di Sistemi e Informatica, viale Morgagni 65, 50134 Firenze, Italy `bilotta@dsi.unifi.it` `elisa@dsi.unifi.it` `pinzani@dsi.unifi.it`

[†]Dipartimento di Scienze Matematiche ed Informatiche, Pian dei Mantellini, 44, 53100, Siena, Italy `rinaldi@unisi.it`

Furthermore, some extensions of the concept of succession rule have been proposed. In [14] the authors admit that a label produces sons at different levels of the generating tree, introducing the so called *jumping succession rules*, while in [10] the author presents *marked succession rules*, where labels can be marked or not. The main difference with *ordinary succession rules* is that marked labels annihilate non-marked labels having the same value and lying at the same level.

Finally, to explore the relationship between succession rules and other formal tools for the enumeration is a rather popular trend of research. A tentative in this direction has been made in [11] with the definition of the so called *production matrix*, which is just a way to represent a given succession rule in terms of an infinite matrix, and supplies the possibility to work with succession rules using some of the algebraic tools developed for matrices.

More recently, there have been some efforts in developing methods to pass from a recurrence relation defining an integer sequence to a succession rule defining the same sequence - in this case we say that the succession rule and the recurrence relation are *equivalent*.

Our work fits into this research line, and tries to deepen the relations between succession rules and recurrence relations.

It is worth mentioning that almost all studies realized until now on this topic have regarded linear recurrence relations with integer coefficients [8, 12]. Following Zeilberger [26], we will address to these as *C-finite recurrence relations*, and to the defined sequences as *C-finite sequences*.

Accordingly, our work will start considering C-finite recurrences. Compared with the methods presented in [8, 12], our approach is completely different.

To achieve this goal, we first translate the given C-finite recurrence relation into an *extended succession rule*, which differs from the *ordinary* succession rules since it admits both jumps and marked labels. Then we recursively eliminate jumps and marked labels from such an extended succession rule, thus obtaining a finite succession rule equivalent to the previous one. We need to point out that this translation is possible only if a certain condition – called *positivity condition* – is satisfied. Such a condition ensures that all the labels of the generating tree are non marked, hence the sequence defined by the succession rule has all positive terms.

If the recurrence relation has degree k with coefficients $a_1, \ldots, a_k$, such a condition can be expressed in terms of a set of k inequalities which can be obtained from a set of quotients and remainders given by the coefficients. To the authors' knowledge, such a condition is completely new in literature. It directly follows that our positive condition provides a sufficient condition for testing the positivity of a C-finite sequence then it relates to the so called *positivity problem*.

Positivity Problem: given a C-finite sequence $\{f_n\}_{n \geq 0}$, establish if all its terms are positive.

This problem was originally proposed as an open problem in [7], and then re-presented in [23] (Theorems 12.1-12.2, pages 73-74), but no general solution has been found yet.

It is worth mentioning that the positivity problem can be solved for a large class of C-finite sequences, precisely those whose generating function is a ℕ-rational series. We also recall that the class of ℕ-rational series is precisely the class of the generating functions of regular languages, and that Soittola's Theorem [24] states that the problem of establishing whether a rational generating function is ℕ-rational is decidable.

Soittola's Theorem has recently been proved in different ways in [8, 22], using different approaches and some algorithms to pass from an ℕ-rational series to a regular expression enumerated by such a series have been proposed [3, 18]. However, none of these techniques provides a method to face C-finite recurrence relations which are not ℕ-rational.

Following the attempt of enlightening some questions on positive sequences, some researches have recently focused on determining sufficient conditions to establish the eventual positivity of a given C-finite recurrence relation, as interestingly described in [16]. As a matter of fact, up to now, we only know that the positivity problem is decidable for C-finite recurrences of two [15]

or three terms [19]. Another possible approach to tackle the positivity problem is to develop algorithms to test eventual positivity of recursively defined sequences (and, in particular, C-finite sequences) by means computer algebra, as in [17].

Our work fits into this research line, since the positive condition we propose is a sufficient condition for testing the positivity of a C-finite sequence.

In the last section we consider the more general case of holonomic integer sequences, i.e., those satisfying a linear recurrence with polynomial coefficients. We show that also in this case we can easily translate the recurrence relation into an infinite succession rule (possibly having marked labels and jumps). A further goal is to find a way to convert such a rule into an ordinary succession rule, and find a more general criterion for proving the positivity of an holonomic sequence. Here we present an example in which we show a possible strategy to perform the desired conversion.

2 Basic definitions and notations

In this section we present some basic definitions and notations related to the concept of succession rule. For further definitions and examples we address the reader to [9].

An *(ordinary) succession rule* Ω can be written in the compact notation:

$$\begin{cases} (a) \\ (k) & \rightsquigarrow (e_1(k))(e_2(k))\dots(e_k(k)) \end{cases} \tag{1}$$

The rule Ω can be represented by means of a *generating tree*, that is a rooted tree whose vertices are the labels of Ω; where (a) is the label of the root and each node labelled (k) produces k sons labelled $(e_1(k)), (e_2(k)), \dots, (e_k(k))$, respectively. As usual, the root lies at level 0, and a node lies at level n if its parent lies at level $n-1$. If a succession rule describes the growth of a class of combinatorial objects, then a given object can be coded by the sequence of labels in the unique path from the root of the generating tree to the object itself.

A succession rule Ω defines a sequence of positive integers $\{f_n\}_{n\geq 0}$ where f_n is the number of the nodes at level n in the generating tree defined by Ω. By convention the root is at level 0, so $f_0 = 1$. The function $f_\Omega(x) = \sum_{n\geq 0} f_n x^n$ is the *generating function* determined by Ω.

Two succession rules are *equivalent* if they have the same generating function. A succession rule is *finite* if it has a finite number of labels and productions.

For example, the two succession rules:

$$\begin{cases} (2) \\ (2) \rightsquigarrow (2)(2) \end{cases} \qquad\qquad \begin{cases} (2) \\ (k) \rightsquigarrow (1)^{k-1}(k+1) \end{cases}$$

are equivalent rules, and define the sequence $f_n = 2^n$. The one on the left is a finite rule, since it uses only the label (2), while the one on the right is an infinite rule.

A slight generalization of the concept of ordinary succession rule is provided by the so called *jumping succession rule*. Roughly speaking, the idea is to consider a set of succession rules acting on the objects of a class and producing sons at different levels.

The usual notation to indicate a jumping succession rule Ω is the following:

$$\begin{cases} (a) \\ (k) & \overset{j_1}{\rightsquigarrow} (e_{11}(k))(e_{12}(k))\dots(e_{1k}(k)) \\ (k) & \overset{j_2}{\rightsquigarrow} (e_{21}(k))(e_{22}(k))\dots(e_{2k}(k)) \\ \vdots \\ (k) & \overset{j_m}{\rightsquigarrow} (e_{m1}(k))(e_{m2}(k))\dots(e_{mk}(k)) \end{cases} \tag{2}$$

The generating tree associated with Ω has the property that each node labelled (k) lying at level n produces k sets of sons at level $n + j_1$, $n + j_2$, ..., $n + j_m$, respectively and each of such set has labels $(e_{i1}(k))$, $(e_{i2}(k))$, ..., $(e_{ik}(k))$ respectively, $1 \leq i \leq m$.

A jumping succession rule Ω defines a sequence of positive integers $\{f_n\}_{n \geq 0}$ where, as usual, f_n is the number of the nodes at level n in the generating tree of Ω. The function $f_\Omega(x) = \sum_{n \geq 0} f_n x^n$ is the *generating function* determined by Ω.

We need to point out that, according to the above definitions, a node labelled (k) has precisely k sons. A rule having this property is said to be *consistent*. However, in many cases we can relax this constraint and consider rules (as in [10]), where the number of sons is a function of the label k.

For example, the jumping succession rule (3) counts the number of *2-generalized Motzkin paths* and Figure 1 shows some levels of the associated generating tree. For more details about these topics, see [14].

$$
\begin{cases}
(1) & \\
(k) & \overset{1}{\rightsquigarrow} (1)(2)\cdots(k-1)(k+1) \\
(k) & \overset{2}{\rightsquigarrow} (k)
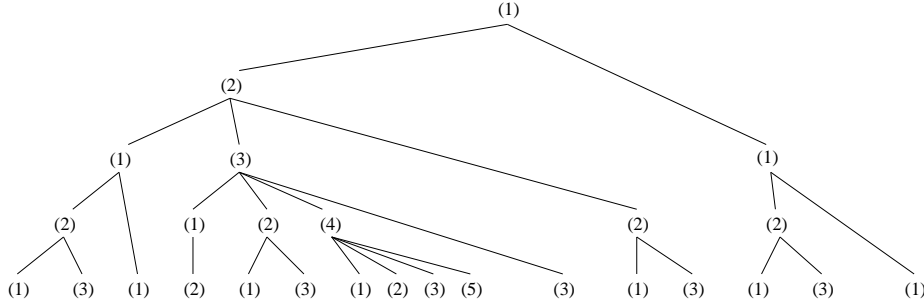\end{cases}
\tag{3}
$$

Figure 1: Four levels of the generating tree associated with the succession rule (3)

Another generalization of succession rules is introduced in [21], where the authors deal with *marked succession rules*. In this case the labels appearing in a succession rule can be marked or not, therefore *marked* are considered together with usual labels. In this way a generating tree can support negative values if we consider a node labelled $(\overline{k})$ as opposed to a node labelled (k) lying on the same level.

A *marked generating tree* is a rooted labelled tree where marked or non-marked labels appear according to the corresponding succession rule. The main property is that, on the same level, marked labels kill or annihilate the non-marked ones with the same label value, in particular the enumeration of the combinatorial objects in a class is the difference between the number of non-marked and marked labels lying on a given level.

For any label (k), we introduce the following notation for generating tree specifications:

$(\overline{\overline{k}}) = (k);$

$(k)^n = \underbrace{(k)\dots(k)}_{n} \quad n > 0;$

$(k)^{-n} = \underbrace{(\overline{k})\dots(\overline{k})}_{n} \quad n > 0.$

For example, the classical succession rule for Catalan numbers can be rewritten in the form

(4) and Figure 2 shows some levels of the associated generating tree.

$$\begin{cases} (2) \\ (k) & \overset{1}{\rightsquigarrow} (2)(3)\dots(k)(k+1)(k) \\ (k) & \overset{1}{\rightsquigarrow} (\overline{k}) \end{cases} \tag{4}$$
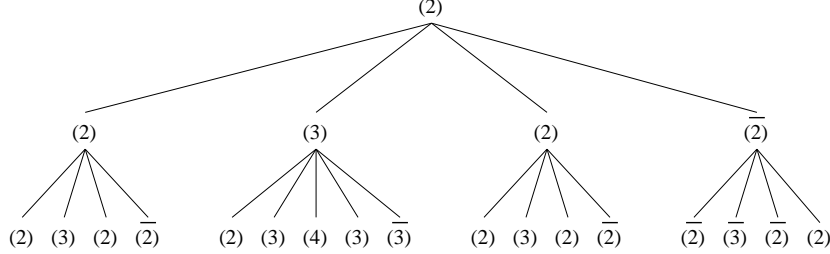
Figure 2: Three levels of the generating tree associated with the succession rule (4)

The concept of marked labels has been implicity used for the first time in [20], then in [10] in relation with the introduction of signed ECO-systems.

3 A general method to translate C-sequences into succession rules

The main purpose of our research is to develop a general formal method to translate a given recurrence relation into a succession rule defining the same number sequence. By abuse of notation, in this case we will say that the recurrence relation and the succession rules are *equivalent*.

As a first step we deal with linear recurrence relations with integer coefficients [8, 12]. Following Zeilberger [26], we will address to these as *C-finite recurrence relations*, and to the defined sequences as *C-finite sequences*.

This section is organized as follows.

i) First we deal with C-sequences of the form

$$f_n = a_1 f_{n-1} + a_2 f_{n-2} + \cdots + a_k f_{n-k} \qquad a_i \in \mathbb{Z}, 1 \le i \le k \tag{5}$$

with *default* initial conditions, i.e. $f_0 = 1$ and $f_h = 0$ for all $h < 0$. We translate the given C-finite recurrence relation into an extended succession rule, possibly using both jumps and marked labels (Section 3.1).

ii) Then, we recursively eliminate jumps and marked labels from such an extended succession rule, thus obtaining a finite succession rule equivalent to the previous one (Section 3.2). We remark that steps i) and ii) can be applied independently of the positivity of $\{f_n\}_{n\ge0}$, but at this step we cannot be sure that all the labels of the obtained rule are nonnegative integers.

iii) We state a condition to ensure that the labels of the obtained succession rule are all nonnegative. If such a condition holds, then the sequence $\{f_n\}_{n\ge0}$ has all positive terms, thus we refer to this as *positivity condition* (Section 3.3).

iv) We show how our method can be extended to *C*-sequences with generic initial conditions (Section 3.4).

3.1 C-sequences with default initial conditions

Let us consider a C-finite recurrence relation expressed as in (5), with default initial conditions and the related C-sequence $\{f_n\}_{n\geq 0}$. We recall that the generating function of $\{f_n\}_{n\geq 0}$ is rational, and precisely it is

$$f(x) = \sum_{n\geq 0} f_n x^n = \frac{1}{1 - a_1 x - a_2 x^2 - \cdots - a_k x^k}\,. \tag{6}$$

The first step of our method consists into translating the C-finite recurrence relation (5) into an extended succession rule. The translation is rather straightforward, since in practice it is just an equivalent way to represent the recurrence relation.

Proposition 3.1 *The recurrence relation (5) with default initial conditions is equivalent to the following extended succession rule:*

$$\begin{cases} (a_1) & \\ (a_1) & \overset{1}{\rightsquigarrow} (a_1)^{a_1} \\ (a_1) & \overset{2}{\rightsquigarrow} (a_1)^{a_2} \\ \vdots & \\ (a_1) & \overset{k}{\rightsquigarrow} (a_1)^{a_k} \end{cases} \tag{7}$$

For example, the recurrence relation $f_n = 3f_{n-1} + 2f_{n-2} - f_{n-3}$ with default initial conditions, defines the sequence $1, 3, 11, 38, 133, 464, 1620, 5655, \ldots$, and it is equivalent to the following extended succession rule:

$$\begin{cases} (3) & \\ (3) & \overset{1}{\rightsquigarrow} (3)^3 \\ (3) & \overset{2}{\rightsquigarrow} (3)^2 \\ (3) & \overset{3}{\rightsquigarrow} (\overline{3}) \end{cases} \tag{8}$$

3.2 Elimination of jumps and marked labels

The successive step of our method consists into recursively eliminating jumps from the extended succession rule (7) in order to obtain a finite succession rule which is equivalent to the previous one. Once jumps have been eliminated we will deal with marked labels.

Proposition 3.2 *The succession rule:*

$$\begin{cases} (a_1) & \\ (a_1) & \rightsquigarrow (a_1 + a_2)(a_1)^{a_1 - 1} \\ (a_1 + a_2) & \rightsquigarrow (a_1 + a_2 + a_3)(a_1)^{a_1 + a_2 - 1} \\ \vdots & \\ (\sum_{l=1}^{k-1} a_l) & \rightsquigarrow (\sum_{l=1}^{k} a_l)(a_1)^{(\sum_{l=1}^{k-1} a_l) - 1} \\ (\sum_{l=1}^{k} a_l) & \rightsquigarrow (\sum_{l=1}^{k} a_l)(a_1)^{(\sum_{l=1}^{k} a_l) - 1} \end{cases} \tag{9}$$

is equivalent to the recurrence relation $f_n = a_1 f_{n-1} + a_2 f_{n-2} + \cdots + a_k f_{n-k}$, $a_i \in \mathbb{Z}, 1 \leq i \leq k$, with default initial conditions.

Proof. Let $A_k(x)$ be the generating function of the label $(\sum_{l=1}^{k} a_l)$ related to the succession rule (9). We have:

$$A_1(x) = 1 + (a_1 - 1)xA_1(x) + (a_1 + a_2 - 1)xA_2(x) + \cdots + (a_1 + a_2 + \cdots + a_k - 1)xA_k(x);$$

$$A_2(x) = xA_1(x);$$

$$A_3(x) = xA_2(x) = x^2 A_1(x);$$

$$\vdots$$

$$A_{k-1}(x) = xA_{k-2}(x) = x^{k-2} A_1(x);$$

$$A_k(x) = xA_{k-1}(x) + xA_k(x) = \frac{x^{k-1}}{1-x} A_1(x).$$

Therefore,

$$A_1(x) = 1 + x(a_1 - 1)A_1(x) + x^2(a_1 + a_2 - 1)A_1(x) + \cdots + \frac{x^k}{1-x}(a_1 + a_2 + \cdots + a_k - 1)A_1(x),$$

and we obtain the generating function $A_1(x) = \frac{1-x}{1 - a_1 x - a_2 x^2 - \cdots - a_k x^k}$.

At this point we can consider the generating function determined by the succession rule (9) as following:

$$A_1(x) + A_2(x) + \cdots + A_{k-1}(x) + A_k(x) = A_1(x) + xA_1(x) + \cdots + x^{k-2} A_1(x) + \frac{x^{k-1}}{1-x} A_1(x) =$$

$$= \frac{(1-x) + x(1-x) + \cdots + x^{k-2}(1-x) + x^{k-1}}{1 - a_1 x - a_2 x^2 - \cdots - a_k x^k} = \frac{1}{1 - a_1 x - a_2 x^2 - \cdots - a_k x^k} \ .$$

∎

Following the previous statement, the extended succession rule (8) – determined in the previous section – can be translated into the following succession rule:

$$\begin{cases} (3) \\ (3) & \rightsquigarrow (5)(3)^2 \\ (5) & \rightsquigarrow (4)(3)^4 \\ (4) & \rightsquigarrow (4)(3)^3 \end{cases} \tag{10}$$

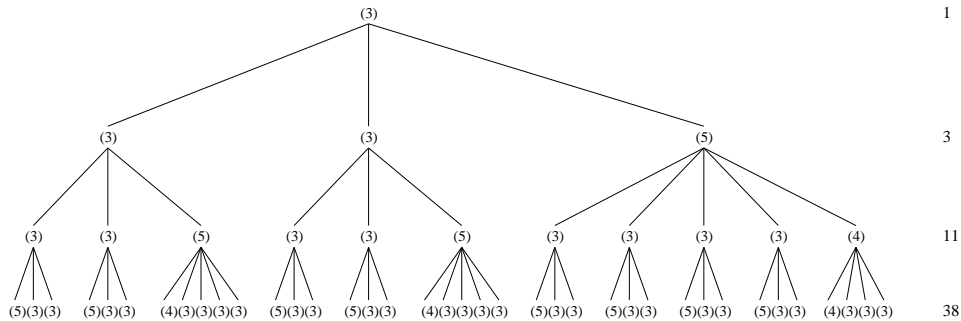Figure 3 shows some levels of the generating tree associated with (10).

Figure 3: Four levels of the generating tree associated with the succession rule (10)

We observe that the previously obtained succession rule is an ordinary finite succession rule, but it may happen that the value of the label $(\sum_{l=1}^{i} a_l)$ is negative, for some i with $i \leq k$, then the succession rule (9) contains marked labels.

For example, the recurrence relation $f_n = 5f_{n-1} - 6f_{n-2} + 2f_{n-3}$, with default initial conditions, which defines the sequence 1,5,19,67,231,791,2703, ..., (sequence A035344 in the The On-Line Encyclopedia of Integer Sequences) is equivalent to the following succession rule:

$$\begin{cases} (5) & \\ (5) & \rightsquigarrow (-1)(5)^4 \\ (-1) & \rightsquigarrow (1)(\overline{5})^2 \\ (1) & \rightsquigarrow (1) \end{cases} \tag{11}$$

and Figure 4 shows some levels of the associated generating tree represented using a "compact notation", i.e., by convention, the number of nodes at a given level n is obtained by means of the algebraic sum of the exponents of the labels lying at level n.
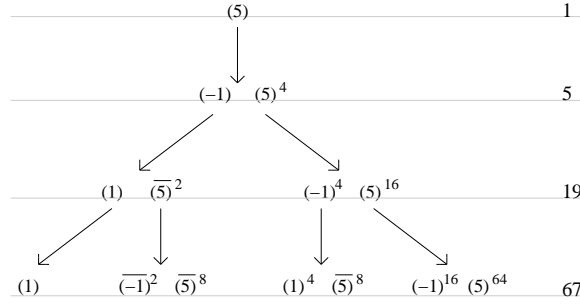

Figure 4: Compact notation for the generating tree associated with the succession rule (11)

Therefore our next goal is to remove all possible marked labels from the succession rule. We observe that in order to obtain this goal, the recurrence relation $f_n = a_1 f_{n-1} + a_2 f_{n-2} + \cdots + a_k f_{n-k}$ with default initial conditions needs $a_1 > 0$. We assume that this condition holds throughout the rest of the present section.

In order to furnish a clearer description of our method, we start considering the case $k = 2$.

Proposition 3.3 *The C-finite recurrence $f_n = a_1 f_{n-1} + a_2 f_{n-2}$, with default initial conditions, and having $a_1 > 0$, is equivalent to*

$$\begin{cases} (a_1) & \\ (a_1) & \rightsquigarrow (0)^{q_2}(r_2)(a_1)^{a_1-(q_2+1)} \\ (r_2) & \rightsquigarrow \left((0)^{q_2}(r_2)\right)^{q_2} (0)^{q_2}(r_2)(a_1)^{r_2-(q_2+1)^2} \end{cases} \tag{12}$$

where, by convention, the label (0) does not produce any son, and q_2, r_2 are defined as follows:

- if $a_1 + a_2 \le 0$ then $q_2, r_2 > 0$ such that $|a_1 + a_2| = q_2 a_1 - r_2$;

- otherwise $q_2 = 0$, $r_2 = a_1 + a_2$.

Proof. We have to distinguish two cases: in the first one $a_1 + a_2 \le 0$ and in the second one $a_1 + a_2 > 0$.

If $a_1 + a_2 \le 0$, we have to prove that the generating tree associated to the succession rule (12) is obtained by performing some actions on the generating tree associated to the extended succession rule (13) which is obviously equivalent to the recurrence $f_n = a_1 f_{n-1} + a_2 f_{n-2}$ having $a_1 > 0$ and $a_2 < 0$, with $f_0 = 1$ and $f_h = 0$ for each $h < 0$.

$$\begin{cases} (a_1) & \\ (a_1) & \overset{1}{\rightsquigarrow} (a_1)^{a_1} \\ (a_1) & \overset{2}{\rightsquigarrow} (a_1)^{a_2} \end{cases} \tag{13}$$

The proof consists in eliminating jumps and marked labels at each level of the generating tree associated with succession rule (13), sketched in Figure 5, by modifying the structure of the generating tree, still maintaining f_n nodes at level n, for each n.

Let (a_1) be a label at a given level n. We denote by B_1 the set of a_1 labels (a_1) at level $n+1$ and by B_2 the set of a_2 labels (a_1) at level $n+2$, see Figure 5. We remark that $(a_1)^{a_2} = \underbrace{\overline{(a_1)}\ldots\overline{(a_1)}}_{-a_2}$.
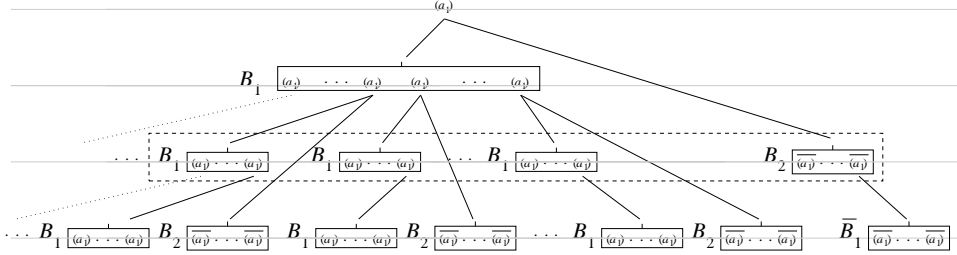

Figure 5: Step 1

In order to eliminate both jumps and marked labels in B_2 at level 2 produced by the root (a_1) at level 0, we have to consider the set of a_1 labels (a_1) in B_1 at level 2 obtained by (a_1) which lie at level 1. At level 2, each label (a_1) in a given set B_1 kills one and only one marked label $\overline{(a_1)}$ in B_2. At this point $|a_1 + a_2|$ labels $\overline{(a_1)}$ in B_2 always exist at level 2.

In order to eliminate such marked labels we have to consider more than a single set B_1 of label (a_1) at level 2. Let q_2 be a sufficient number of sets B_1 at level 2 able to kill all the labels $\overline{(a_1)}$ in B_2 at level 2. Therefore $|a_1 + a_2| = q_2 a_1 - r_2$ with $q_2, r_2 > 0$.

By setting q_2 labels (a_1) at level 1 equal to (0) and one more label (a_1) to (r_2), we have the desired number of labels (a_1) at level 2. Note that the marked labels at level 2 are not generated and the labels (a_1) at level 1 are revised in order to have the right number of labels at level 2, see Figure 6.
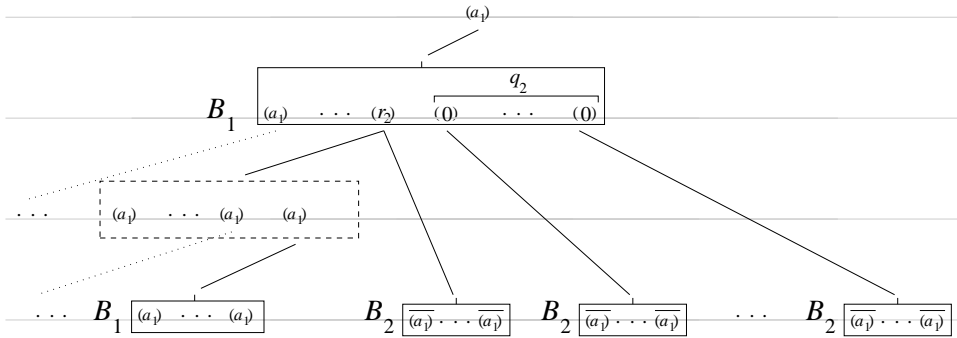

Figure 6: Step 2

Note that, when a label (a_1) kills a marked label $\overline{(a_1)}$ at a given level n, then the subtree, having such label (a_1) as its root, kills the subtree having $\overline{(a_1)}$ as its root. So, at level 2 when a label (a_1) of B_1 kills a label $\overline{(a_1)}$ of B_2 then the two subtrees having such labels as their roots are eliminated too, see Figure 6.

On the other hand, the $q_2 + 1$ sets B_2 at level 3 obtained by the $q_2 + 1$ labels at level 1, once labelled with (a_1) and now having value $r_2, 0, \ldots, 0$, respectively, are always present in the tree, see Figure 6. In order to eliminate such undesired marked labels we can only set the production of (r_2). As a set B_2 at a given level is eliminated by using $q_2 + 1$ labels at previous level then

(r_2) must give $(r_2)\underbrace{(0)\dots(0)}_{q_2}$ exactly $q_2 + 1$ times. This explains the first part of the production rule of the label (r_2) in rule (12). Since (r_2) has r_2 sons then the remaining $r_2 - (q_2+1)^2$ labels are set to be equal to (a_1) as in the previous case, see Figure 7.
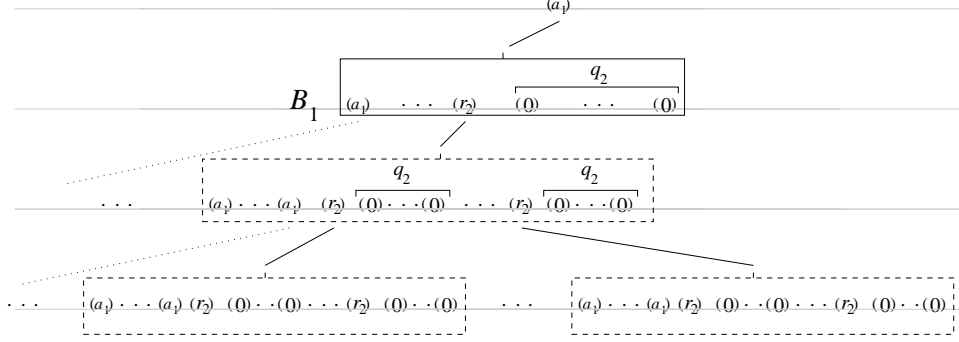

Figure 7: Step 3

By the way, the modified $q_2 + 1$ labels having value $r_2, 0, \dots, 0$, respectively, at a given level n, produce the labels $\left((0)^{q_2}(r_2)\right)^{q_2+1}(a_1)^{r_2-(q_2+1)^2}$ at level $n + 1$. Just as obtained for levels 1 and 2, the labels $\left((0)^{q_2}(r_2)\right)^{q_2+1}$ automatically annihilate the remaining $q_2 + 1$ sets B_2 of marked labels at level $n + 2$, once obtained by the modified $q_2 + 1$ labels at level n, see Figure 7.

Till now we have modified a portion P of the total generating tree in a way that it does not contain any marked label. Note that, the remaining labels (a_1) will be the roots of subtrees which are all isomorphic to P.

The value f_n defined by the tree associated to the extended succession rule (13), is given by the difference between the number of non-marked and marked labels. The just described algorithm modifies the number of generated non-marked labels and sets to 0 the number of marked ones in a way that f_n is unchanged, for each n, so the succession rule (12) is equivalent to the recurrence $f_n = a_1 f_{n-1} + a_2 f_{n-2}$.

In the case $a_1 + a_2 > 0$ we have marked labels only if $a_2 < 0$. In this case a single set B_1 is sufficient to kill all the marked labels in B_2 at level 2. By the way, both in the case $a_2 < 0$ and $a_2 > 0$ we have that $q_2 = 0$ and $r_2 = a_1 + a_2$, and the succession rule (12) has the same form of the rule (9) which is equivalent to the recurrence $f_n = a_1 f_{n-1} + a_2 f_{n-2}$ having $a_1 > 0$ and $a_2 \in \mathbb{Z}$, with $f_0 = 1$ and $f_h = 0$ for each $h < 0$. ∎

The statement of Proposition 3.3 can be naturally extended to the general case $k > 2$.

Proposition 3.4 *The C-sequence $\{f_n\}_n$ satisfying $f_n = a_1 f_{n-1} + a_2 f_{n-2} + \dots + a_k f_{n-k}$, with default initial conditions and $a_1 > 0$ is equivalent to*

$$
\begin{cases}
(a_1) \\
(a_1) & \rightsquigarrow (0)^{q_2}(r_2)(a_1)^{a_1-(q_2+1)} \\
(r_2) & \rightsquigarrow \left((0)^{q_2}(r_2)\right)^{q_2}(0)^{q_3}(r_3)(a_1)^{r_2-(q_2(q_2+1)+q_3+1)} \\
\vdots \\
(r_i) & \rightsquigarrow \left((0)^{q_2}(r_2)\right)^{q_i}(0)^{q_{i+1}}(r_{i+1})(a_1)^{r_i-(q_i(q_2+1)+q_{i+1}+1)}, \quad 3 \le i \le k-1 \\
\vdots \\
(r_k) & \rightsquigarrow \left((0)^{q_2}(r_2)\right)^{q_k}(0)^{q_k}(r_k)(a_1)^{r_k-(q_k(q_2+1)+q_k+1)}
\end{cases}
\tag{14}
$$

where the parameters q_i and r_i, with $2 \leq i \leq k$, can be determined in the following way:

- if $\sum_{l=1}^{i} a_l \leq 0$ then $q_i, r_i > 0$ such that $|\sum_{l=1}^{i} a_l| = q_i a_1 - r_i$,

- otherwise $q_i = 0$ and $r_i = \sum_{l=1}^{i} a_l$.

The proof of the Proposition 3.4 is quite similar to the proof of Proposition 3.3. It has the same level of difficulty but it is more cumbersome, so it is omitted for brevity.

Using Proposition 3.4, we can translate the previously considered recurrence relation $f_n = 5f_{n-1} - 6f_{n-2} + 2f_{n-3}$, with default initial conditions, into the following ordinary succession rule:

$$\begin{cases} (5) & \\ (5) & \rightsquigarrow (0)(4)(5)^3 \\ (4) & \rightsquigarrow (0)(4)(1)(5) \\ (1) & \rightsquigarrow (1) \end{cases} \tag{15}$$

being $q_2 = 1$, $r_2 = 4$, $q_3 = 0$ and $r_3 = 1$.

3.3 Positivity condition

The statement of Proposition 3.4 is indeed a tool to translate C-recurrences into finite succession rules. However this property turns out to be effectively applicable only when the labels of the succession rule are all positive, and the reader can easily observe that Proposition 3.4 does not give us an instrument to test whether this happens or not.

In particular, if the labels of the succession rule are all positive then the terms of the C-sequence are all positive. It is then interesting to relate our problem with the so called *positivity problem*, which we have already mentioned in the Introduction.

Positivity Problem: given a C-finite sequence $\{f_n\}_{n \geq 0}$, establish if all its terms are positive.

We recall that the problem was originally proposed as an open problem in [7], and then re-presented in [23] (Theorems 12.1-12.2, pages 73-74), but no general solution has been found yet.

Moreover, the positivity problem can be solved for a large class of C-finite sequences, precisely for $\mathbb{N}$-rational sequences. We recall that the class of $\mathbb{N}$-rational series is precisely the class of the generating functions of regular languages, and that Soittola's Theorem [24] states that the problem of establishing whether a rational generating function is $\mathbb{N}$-rational is decidable.

Let us start examining the case of C-recurrences of degree 2. So, let $f_n = a_1 f_{n-1} + a_2 f_{n-2}$ be a recurrence relation, with $a_1 > 0$ and $a_2 \in \mathbb{Z}$.

By referring to the succession rule (12), precisely to the case $a_1 + a_2 \leq 0$, we observe that the succession rule equivalent to the recurrence relation is an ordinary rule (i.e., it has all positive labels) if and only if the following condition is verified:

$$\begin{cases} a_1 - (q_2 + 1) \geq 0 \\ r_2 - (q_2 + 1)^2 \geq 0 \end{cases} \tag{16}$$

As $r_2 = q_2 a_1 - |a_1 + a_2| = q_2 a_1 + a_1 + a_2$ then $r_2 - (q_2 + 1)^2 \geq 0$ means $q_2^2 + (2 - a_1)q_2 + 1 - a_1 - a_2 \leq 0$. This inequality has solution if and only if $a_1^2 + 4a_2 \geq 0$, and this is clearly a necessary and sufficient condition to ensure the positivity of all the terms of f_n [8] .

Let us now consider a generic C-recurrence of degree k. Using a similar reasoning, and following Proposition 3.4 we can prove:

Corollary 3.1 *Let us consider the recurrence relation $f_n = a_1 f_{n-1} + a_2 f_{n-2} + \cdots + a_k f_{n-k}$ having $a_1 > 0$ and $a_i \in \mathbb{Z}$, $2 \leq i \leq k$, with $f_0 = 1$ and $f_h = 0$ for each $h < 0$. If*

$$
\begin{cases}
a_1 - (q_2 + 1) \geq 0 \\
r_2 - (q_2(q_2 + 1) + q_3 + 1) \geq 0 \\
\vdots \\
r_i - (q_i(q_2 + 1) + q_{i+1} + 1) \geq 0 \quad , \quad 3 \leq i \leq k - 1 \\
\vdots \\
r_k - (q_k(q_2 + 1) + q_k + 1) \geq 0
\end{cases}
\tag{17}
$$

then $f_n > 0$ for all n.

As $r_i = \sum_{l=1}^{i} a_l + q_i a_1$, $2 \leq i \leq k$, then the system (17) can be rewritten as

$$
\begin{cases}
a_1 - (q_2 + 1) \geq 0 \\
\vdots \\
\sum_{l=1}^{i} a_l + q_i a_1 - (q_i(q_2 + 1) + q_{i+1} + 1) \geq 0 \quad , \quad 2 \leq i \leq k - 1 \\
\vdots \\
\sum_{l=1}^{k} a_l + q_k a_1 - (q_k(q_2 + 1) + q_k + 1) \geq 0.
\end{cases}
\tag{18}
$$

As previously mentioned, condition (18) ensures that all the labels of the succession rules equivalent to the given C-recurrence relation are positive, hence all the terms f_n are positive. Thus it can be viewed as a sufficient condition to test the positivity of a given C-recurrence relation.

Unfortunately, this is not a necessary condition to test positivity, then there are cases of positive C-sequences for which our method fails to prove positivity. A simple example is given by any positive non $\mathbb{N}$-rational C-sequence. The reader can find an instance of such sequences in [18]. It would be more interesting to give an example of a $\mathbb{N}$-rational C-sequence for which our method is not able to prove positivity, but we have not been able to find any such example.

Clearly, any C-sequence satisfying the positivity condition has a $\mathbb{N}$-rational generating function (in fact, any finite succession rule may be regarded as a finite state automaton), thus our method can be suitably used to test the $\mathbb{N}$-rationality of a sequence. Though it is not our intention to deepen the computational complexity of our test, we remark that, despite the methods presented in [8, 18, 22], our method does not deal with calculating polynomial roots.

In order to give an idea of the computational cost to solve the system (18) we consider the worst case that is when $\sum_{l=1}^{i} a_l \leq 0$, $2 \leq i \leq k$, and the system itself has no solution.

In this case all the possible values for each q_i, $2 \leq i \leq k$, must be checked in order to conclude that the system (18) does not admit any solution.

As q_2 can range in the close set $[1, a_1 - 1]$ and q_{i+1} in $[1, \sum_{l=1}^{i} a_l + q_i a_1 - (q_i(q_2 + 1) - 1]$ then we have

$$
1 + (a_1 - 1) \prod_{i=2}^{k-1} \left(\sum_{l=1}^{i} a_l + a_1 q_i - q_i(q_2 + 1) - 1 \right)
$$

where the first 1 accounts the check to verify $\sum_{l=1}^{k} a_l + q_k a_1 - (q_k(q_2 + 1) + q_k + 1) \geq 0$.

An average complexity study of our test is a further development. Anyway, the referred experimental results give a sufficiently short computational time to test condition (18).

3.4 Generic initial conditions

Now it is possible to use the statement of Proposition 3.4 to treat the case of C-recurrence relations with generic initial conditions. The following result is obtained by simply adapting the productions of the labels in the first levels of the generating tree to the given initial conditions, then using the productions of Proposition 3.4. So, we have two sets of productions: the ones stating the initial conditions, and the remaining ones defining all the other levels.

Proposition 3.5 *Let us consider the C-finite recurrence relation $f_n = a_1 f_{n-1} + a_2 f_{n-2} + \cdots + a_k f_{n-k}$, $a_i \in \mathbb{Z}$, $1 \le i \le k$, and let us assume that the initial conditions are $f_0 = 1$ and $f_i = h_i$, with $h_i \in \mathbb{Z}$, $1 \le i < k$, then it can be translated into the following extended succession rule:*

$$
\begin{cases}
(h_1) & \\
(h_1) & \overset{1}{\leadsto} (a_1)^{h_1} \\
\quad \vdots & \\
(h_1) & \overset{i}{\leadsto} (a_1)^{h_i - \sum_{j=1}^{i-1} h_j a_{i-j}} \quad , \quad 1 < i < k \\
\quad \vdots & \\
(h_1) & \overset{k}{\leadsto} (a_1)^{a_k} \\
& \\
(a_1) & \overset{1}{\leadsto} (a_1)^{a_1} \\
(a_1) & \overset{2}{\leadsto} (a_1)^{a_2} \\
\quad \vdots & \\
(a_1) & \overset{k}{\leadsto} (a_1)^{a_k}
\end{cases}
\tag{19}
$$

For example, the recurrence relation $f_n = 3f_{n-1} + 2f_{n-2} - f_{n-3}$ with $f_0 = 1$, $f_1 = 2$ and $f_2 = 3$, which defines the sequence $1, 2, 3, 12, 40, 141, 491, 1715, \ldots$, is equivalent to the following extended succession rule:

$$
\begin{cases}
(2) & \\
(2) & \overset{1}{\leadsto} (3)^2 \\
(2) & \overset{2}{\leadsto} (\overline{3})^3 \\
(2) & \overset{3}{\leadsto} (\overline{3}) \\
& \\
(3) & \overset{1}{\leadsto} (3)^3 \\
(3) & \overset{2}{\leadsto} (3)^2 \\
(3) & \overset{3}{\leadsto} (\overline{3})
\end{cases}
\tag{20}
$$

and Figure 8 shows some levels of the generating tree associated to it.

Following the described method in Section 3.2 to eliminate jumps and marked labels, we can translate the extended succession rule (20) into the ordinary succession rule (21), where the labels (3), $(3)_1$ and $(3)_2$ are different labels with different productions.

$$
\begin{cases}
(2) & \\
(2) & \leadsto (0)(3)_1 \\
(3)_1 & \leadsto (6)(3)^2 \\
(6) & \leadsto (3)^5(3)_2 \\
(3)_2 & \leadsto (4)(3)^2 \\
& \\
(3) & \leadsto (5)(3)^2 \\
(5) & \leadsto (4)(3)^4 \\
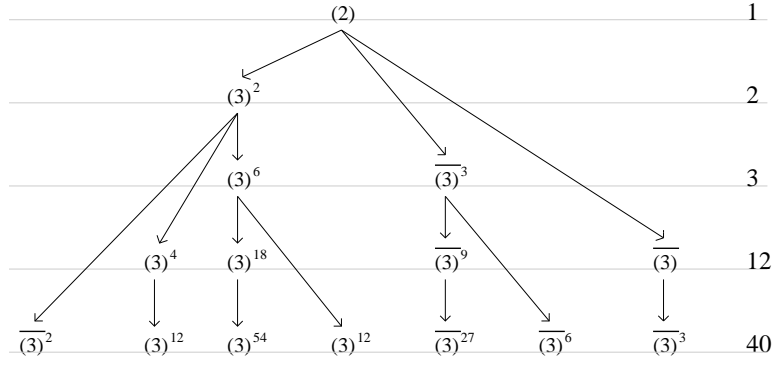(4) & \leadsto (4)(3)^3
\end{cases}
\tag{21}
$$

Figure 8: Compact notation for the generating tree associated with the succession rule (20)

4 Concluding remarks

In this paper we have presented a general method to translate a given C-finite recurrence relation into an ordinary succession rule and we have proposed a sufficient condition for testing the positivity of a given C-finite sequence.

A further development could take into consideration the average complexity necessary to prove the positivity of a given C-finite sequence.

Afterwards, it should be interesting to develop the study concerning the C-recurrence relations with generic initial conditions in order to examine in depth the potentiality of our method.

Finally, we would like to show that some of our ideas can be applied to the case of holonomic integer sequences, i.e., those satisfying a linear recurrence relation with polynomial coefficients.

Just to have a simple example, let us consider the *involutions* of n, enumerated by the sequence $\{f_n\}$ defined by the holonomic recurrence relation

$$f_n = f_{n-1} + (n-1)f_{n-2}, \tag{22}$$

with $f_0 = 1$, $f_1 = 1$ (sequence A000085 in the The On-Line Encyclopedia of Integer Sequences).

We easily observe that, using the same argument of Proposition 3.2, we can translate the recurrence relation (22) into an infinite succession rule (possibly having marked labels and jumps), where now we adopt the convention that a generic label (k) is placed at the level k of the generating tree:

$$\begin{cases} (0) \\ (k) \overset{1}{\rightsquigarrow} (k+1) \\ (k) \overset{2}{\rightsquigarrow} (k+2)^{k+1} \end{cases} \tag{23}$$

The successive step is to find a way how to convert such a rule into an ordinary succession rule. Referring to (23), this can be done by eliminating "by hand" marked labels and jumps, then re-writing the (ordinary) rule as follows:

$$\begin{cases} (1) \\ (k) \rightsquigarrow (k-1)^{k-1}(k+1) \end{cases} \tag{24}$$

We believe that such a method should be formalized in some further work, and then applied to automatically convert the obtained rule into an ordinary succession rule.

Moreover, from this method, we could also develop a more general criterion for proving the positivity of an holonomic sequence.

14

References

[1] S. Bacchelli, E. Barcucci, E Grazzini, E. Pergola. Exhaustive generation of combinatorial objects using ECO. *Acta Informatica* 40 (8) (2004) 585-602.

[2] C. Banderier, M. Bousquet-Mélou, A. Denise, P. Flajolet, D. Gardy, D. Gouyou-Beauchamps. Generating functions for generating trees. *Discrete Mathematics* 246 (2002) 29–55.

[3] E. Barcucci, A. Del Lungo, A. Frosini, S. Rinaldi. A technology for reverse-engineering a combinatorial problem from a rational generating function. *Advances in Applied Mathematics* 26 (2) (2001) 129-153.

[4] E. Barcucci, A. Del Lungo, E. Pergola, R. Pinzani. Random generation of trees and other combinatorial objects. *Theoretical Computer Science* 218 (1999) 219-232.

[5] E. Barcucci, S. Rinaldi. Some linear recurrences and their combinatorial interpretation by means of regular languages. *Theoretical Computer Science* 255 (2001) 679-686.

[6] E. Barcucci, A. Del Lungo, E. Pergola, R. Pinzani. ECO: a methodology for the Enumeration of Combinatorial Objects. *Journal of Difference Equations and Applications* 5 (1999) 435-490.

[7] J. Berstel, M. Mignotte. Deux propriétés décidables des suites récurrentes linéaires. *Bulletin de la Société Mathématique de France* 104 (2) (1976) 175–184.

[8] J. Berstel, C. Reutenauer. Another proof of Soittola's Theorem. *Theoretical Computer Science* 393 (2008) 196-203.

[9] F. R. K. Chung, R. L. Graham, V. E. Hoggatt, M. Kleimann. The number of Baxter permutations. *Journal of Combinatorial Theory Series A* 24 (1978) 382–394.

[10] S. Corteel. Series generatrices exponentielles pour les eco-systemes signes. *Proceedings of the 12-th International Conference on Formal Power Series and Algebraic Combinatorics*, Moscow (2000).

[11] E. Deutsch, L. Ferrari, S. Rinaldi. Production matrices. *Advances in Applied Mathematics* 34 (2005) 101–122.

[12] E. Duchi, A. Frosini, R. Pinzani, S. Rinaldi. A note on rational succession rules. *Journal of Integer Sequences* 6 (2003) Article 03.1.7.

[13] L. Ferrari, E. Pergola, R. Pinzani, S. Rinaldi. An algebraic characterization of the set of succession rules. *Theoretical Computer Science* 281 (2002) 351–367.

[14] L. Ferrari, E. Pergola, R. Pinzani, S. Rinaldi. Jumping succession rules and their generating functions. *Discrete Mathematics* 271 (2003) 29–50.

[15] V. Halava, T. Harju, M. Hirvensalo. Positivity of second order linear recurrent sequences. *Discrete Applied Mathematics* 154 (2006) 447–451.

[16] I. Gessel. Rational functions with nonnegative integer coefficients. In *The 50th seminaire Lotharingien de Combinatoire*, page Domaine Saint-Jacques, March (2003). Unpublished, available at Gessels homepage.

[17] S. Gerhold. Sequences: Non-Holonomicity and Inequalities. *Ph.D. Thesis.*

[18] C. Koutschan. Regular languages and their generating functions: The inverse problem. *Theoretical Computer Science* 391 (2008) 65–74.

[19] V. Laohakosol, P. Tangsupphathawat. Positivity of third order linear recurrence sequences. *Discrete Applied Mathematics* 157 (2009) 3239–3248.

[20] D. Merlini, R. Sprugnoli, M. C. Verri. An Algebra for proper generating tree. In *Algorithms, trees, combinatorics and probabilities*. Trends in Mathematics, Mathematics and Computer Science (2000) 127–139.

[21] D. Merlini, M. C. Verri. Generating trees and proper Riordan Arrays. *Discrete Mathematics* 218 (2003) 167–183.

[22] D. Perrin. On positive matrices. *Theoretical Computer Science* 94 (2) (1992) 357–366.

[23] A. Salomaa, M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*, Springer-Verlag, New York, 1978.

[24] M. Soittola. Positive rational sequences. *Theoretical Computer Science* 2 (3) (1976) 317-322.

[25] J. West. Generating trees and the Catalan and Schröder numbers. *Discrete Mathematics* 146 (1995) 247–262.

[26] D. Zeilberger. A holonomic systems approach to special functions identities. *Journal of Computational and Applied Mathematics* 32 (1990) 321–368.