

# On Rational Recursive Sequences

$$\begin{cases} b_{n+1} = x+1 \\ c_{n+1} = y(x+1) \end{cases}$$

**Abstract**—We study the class of rational recursive sequences (ratrec) over the rational numbers. A ratrec sequence is defined via a system of sequences using mutually recursive equations of depth 1, where next values are computed as rational functions of the previous values. An alternative class is that of simple ratrec sequences, where one uses a single, but depth  $k$  recursive equation, and thus the next value is a rational function of  $k$  previous values. Our main result is that the classes ratrec and simple ratrec coincide. The class ratrec strictly generalises a well-known class of polynomial recursive sequences (polyrec), defined like ratrec, where polynomial functions instead of rational ones are used in the recursion. As a corollary of our main result, we prove that the zeroness and equivalence problems for polyrec are in **coREXPTIME**. We complement this result with a **PSPACE** lower bound for both problems for polyrec. The upper and the lower bound for zeroness and equivalence also hold for polynomial automata over a unary alphabet, since this class is known to be equivalent to polyrec.

## I. INTRODUCTION

The topic of this paper are recursively defined sequences of rational numbers  $\mathbb{N} \rightarrow \mathbb{Q}$ . There are two natural ways to define such sequences. In a *simple recursion of depth  $k$*  one fixes  $k$  initial values and defines the next value as a function of the previous  $k$  values. This is how the Fibonacci sequence is usually defined (with  $k = 2$ ):  $f_0 = 0$ ,  $f_1 = 1$ , and  $f_{n+2} = f_{n+1} + f_n$ . In a *mutual recursion of width  $k$*  one defines a system of  $k$  sequences such that every sequence has its initial value and the update function can access the immediately previous value of all  $k$  sequences, but no older value. For example, we can define  $a_n = n^2$  with an extra sequence  $b_n = n$  as follows:  $a_0 = b_0 = 0$  and  $a_{n+1} = a_n + 2b_n + 1$ ,  $b_{n+1} = b_n + 1$ . Both styles allow to define various classes of sequences depending on what operations are allowed in the equations, and in general mutual recursion of width  $k$  can simulate simple recursion of depth  $k$  (by adding sufficiently many auxiliary sequences).

One of the most well-known classes of sequences is the class of *linear recursive sequences*, which is obtained by allowing the update function to use addition and multiplication with constants. These are usually defined with a simple recursion, like in the Fibonacci example, but in fact, as a consequence of the Cayley-Hamilton theorem, one obtains the same class when using mutual recursion [20, Lemma 1.1]. In particular, all the example sequences  $f_n$ ,  $a_n$  and  $b_n$  are linear recursive. Another natural class of sequences are the *polynomial recursive sequences (polyrec)*, which are defined with mutual recursion and updates from the ring of polynomial functions  $\mathbb{Q}[x_1, \dots, x_k]$ . An example sequence from this class is  $c_n = n!$ , where one can use the already defined sequence  $b_n$  and define  $c_0 = 1$  and  $c_{n+1} = c_n \cdot (b_n + 1)$ . To see the polynomials behind this definition, let  $x$  and  $y$  be variables

corresponding to  $b_n$  and  $c_n$ , respectively. The polynomial to define  $b_{n+1}$  is  $P_b(x, y) = x + 1$ , and the polynomial to define  $c_{n+1}$  is  $P_c(x, y) = y(x + 1)$ . The class of *simple polynomial recursive sequences* is obtained by using polynomial updates and a simple recursion (instead of mutual recursion) and it is known to be strictly included in the class of all polyrec sequences. In particular, the sequence  $c_n$  is polyrec but not simple polyrec [13, Theorem 5].

The definition via mutual recursion appears in the area of control theory (under the name *implicit representation* of the space of states), and, in computer science, in the context of *weighted automata* over  $\mathbb{Q}$ . Such automata output a rational number for every word over a finite alphabet  $\Sigma$ , and they are defined by linear updates [18]. Linear recursive sequences are thus equivalent to weighted automata with a 1-letter alphabet  $\Sigma = \{a\}$  [4]. Similarly, polyrec sequences are equivalent to *polynomial automata* [5] (also known as *cost-register automata* [2]) with a 1-letter alphabet [13].

We are interested in two classical decision problems for such automata. *Equivalence*: Given two automata  $\mathcal{A}$  and  $\mathcal{B}$  do they output the same number for every word, and *zeroness*: Does the input automaton  $\mathcal{A}$  output 0 for every word. These problems are well-known to be efficiently equivalent to each other: Zeroness is clearly a special case of equivalence (just take  $\mathcal{B}$  to output zero for every word), and equivalence of  $\mathcal{A}, \mathcal{B}$  reduces to zeroness of the difference automaton  $\mathcal{A} - \mathcal{B}$  with the expected semantics. Therefore, we will consider only the zeroness problem. From the seminal work of Schützenberger on minimisation of weighted automata it follows that the zeroness problem for weighted automata is in **PTIME** [32]. For polynomial automata, zeroness is known to be Ackermann-complete [5]. Using the connection between sequences and automata one immediately obtains **PTIME** and Ackermann upper bounds for the zeroness problem of linear recursive sequences, resp., polyrec sequences.

Let us take a closer look at the zeroness problem for recursive sequences, i.e., given a sequence  $u_n$  is it the case that  $u_n = 0$  for all  $n \in \mathbb{N}$ ? The zeroness problem is a fundamental problem for number sequences. It is a basic building block in computer algebra, e.g., in proving identities involving recursively defined sequences. It is also important from a theoretical point of view as a yardstick of the well-behavedness of classes of number sequences, i.e., interesting classes of sequences should at least have a decidable zeroness problem. The difficulty of solving the zeroness problem in general depends on how the sequence is presented. If the sequence is defined with a simple recursion of depth  $k$  such as  $u_{n+k} = f(u_{n+k-1}, \dots, u_n)$ , then zeroness trivially reduces to checking that the first  $k$  values are 0 and that the recursive

update  $f$  needs to output 0 when the previous values are 0, i.e.,  $f(0, \dots, 0) = 0$ . However, this simple reasoning is flawed in the case of mutual recursion, because the auxiliary sequences employed in the mutual recursion need not be zero. However, for linear recursive sequences the zeroness problem is easily solved even in the case of mutual recursion, because the reduction to simple recursion [20, Lemma 1.1] implies that  $a_n$  is zero if, and only if, its first  $k+1$  values  $a_0 = \dots = a_k$  are zero. For polyrec sequences we cannot apply this argument since mutual recursion cannot be simulated by simple recursion in the case of polynomial updates.

*Our results:* In this paper we introduce the class of *rational recursive sequences (ratrec)*. This class is defined with mutual recursion and updates from the field of rational functions  $\mathbb{Q}(x_1, \dots, x_k)$ . For example, the Catalan numbers  $C_{n+1} = \frac{2(2n+1)}{n+2} C_n$  can be defined using  $b_n$  as an auxiliary sequence. Namely,  $C_0 = 1$  and  $C_{n+1} = \frac{2(2b_n+1)}{b_n+2} C_n$ , where the rational function used to define  $C_{n+1}$  is  $R(x, y) = \frac{2(2x+1)}{x+2} y$ . By definition, the class of polyrec sequences is included in the class of ratrec sequences, and in fact the inclusion is strict as witnessed by the fact that the Catalan numbers  $C_n$  are not polynomially recursive [13, Corollary 8]. Moreover, ratrec sequences also include the well-known and wide-spread *P-recursive sequences*<sup>1</sup> [22], which according to a 2005 estimate comprise at least 25% of the OEIS archive [30].

The natural question is whether the class of ratrec sequences semantically collapses to the class of *simple rational recursive sequences* obtained by adopting simple recursion. Unlike in the case of polynomial updates, in our main result we show that for rational updates we do have such a collapse.

**Theorem 1.** *TRUE FOR SEQ OF RAT. FUN. The class of rational recursive sequences coincides with the class of simple rational recursive sequences.*

To see the power of ratrec sequences recall that  $c_n = n!$  is not a simple polyrec sequence. However, when in the recursion we allow rational functions, then  $c_n$  can be defined with a simple recursion, namely:  $c_{n+2} = \frac{(c_{n+1})^2}{c_n} + c_{n+1}$ . Thus  $c_n$  is simple ratrec.

Our main tool in the proof of Theorem 1 comes from commutative algebra. Instead of looking at the elements of a ratrec sequence as numbers in the field of rationals  $\mathbb{Q}$ , we symbolically view them as elements of the field of rational functions  $\mathbb{Q}(x_1, \dots, x_k)$ . From these functions we build a sequence of field extensions

$$\mathbb{Q} \subseteq \mathbb{F}_0 \subseteq \mathbb{F}_1 \subseteq \mathbb{F}_2 \subseteq \dots \subseteq \mathbb{Q}(x_1, \dots, x_k)$$

and translate the problem of belonging to the class of simple ratrec sequences to the question of whether this sequence of field extensions eventually stabilises. In order to estimate at which level the stabilisation occurs we use certain results on basic algorithms for rational function fields [24].

<sup>1</sup> Sometimes P-recursive sequences are also called *holonomic sequences*, due to a connection with holonomic generating functions.

Our second result is a complexity upper bound for the zeroness problem of polyrec sequences.

**Theorem 2.** *The zeroness problem for polynomial recursive sequences over  $\mathbb{Q}$  is in coREXPTIME.*

Here, coREXPTIME is the complement of randomised exponential time, and it is contained in coNEXPTIME. Let us remark that for the purpose of computation we assume that polyrec sequences are represented on input by stating the initial condition and providing relevant polynomials using arithmetic circuits, but we also show that various possible representations are equivalent w.r.t. polynomial-time reductions. See Sec. V for a broader discussion.

The proof of Theorem 2 relies on a careful analysis of the complexity of the translation of a polyrec sequence to a simple ratrec one from Theorem 1: We establish that a simple ratrec sequence with an exponential recursion depth suffices to simulate a polyrec one. This implies that a polyrec sequence is zero if, and only if, the first exponentially many elements thereof are zero. The latter condition can be checked in coREXPTIME using randomisation techniques in order to keep the magnitude of the numbers involved under control.

The coREXPTIME upper bound from Theorem 2 should be contrasted with the (worse) Ackerman upper bound inherited from polynomial automata from [5]. This suggests that for polyrec sequences the natural object of study are rational function fields, which are of more algebraic nature and provide better complexity bounds than the order-theoretic techniques based on sequences of polynomial ideals and Hilbert's finite basis theorem [5].

Our final result is a complexity lower bound for the zeroness problem of polyrec sequences.

**Theorem 3.** *The zeroness problem for polynomial recursive sequences is PSPACE-hard.*

As far as we know, prior to this work nothing was known about the complexity of zeroness for polyrec sequences, except for the Ackermann upper bound following from polynomial automata [5]. The lower bound is proved by reducing from the QBF validity problem, and as a consequence of the construction it follows that the exponential blow-up of the recursion depth in the translation from Theorem 1 is unavoidable.

Given Theorems 1 and 2 it seems natural to investigate the zeroness problem for ratrec sequences. The issue is that it is not clear what would be the input for such a decision problem. Recall that to define ratrec sequences we allow for rational functions in the recursion, which means that we have to deal with division in order to compute the elements of the sequences. Then either one would require that the input sequence comes with a promise that all elements are well-defined and no division by 0 occurs; or one would need to verify whether division by 0 occurs in the input sequence. We find the former solution unnatural, and the latter is at least as hard as the so-called Skolem problem (c.f. below), which is not known to be decidable even for linear recursive sequences.

$$C_0 = x \quad C_1 = xy \quad C_2 = \frac{xy^2}{x} + xy = xy(y+1) \quad \dots \quad C_n = n! \quad \left\{ \begin{array}{l} C_0 = x, C_1 = xy, C_2 = xy(y+1) \\ C_{n+1} = (n+1) \cdot C_n = d_n \cdot C_n \\ d_0 = y, d_1 = 1+y, d_2 = 2+y \\ d_{n+1} = 1+d_n \end{array} \right.$$

*Related work:* The zeroness problem has been extensively studied. In the field of automata theory, we can mention applications to the equivalence problem of several classes of automata and grammars, starting from the weighted finite automata [32] and polynomial automata [5] already mentioned above, and including context-free grammars [14], multiplicity equivalence of finite automata [36] and multitape finite automata [21], [37], unambiguous context-free grammars [29, Theorem 5.5] (c.f. [15] for a PSPACE upper bound), polynomial grammars (which generalise polynomial automata) [9, Chapter 11], deterministic top-down tree-to-string transducers [33], MSO transductions on unordered forests [7], [8], MSO transductions of bounded treewidth under a certain equivalence relation [10], Parikh automata [11], and unambiguous register automata [3]. By replacing (pointwise) multiplication with convolution in the definition of polyrec sequences we obtain the so-called *convolution recursive sequences*, for which the zeroness problem can be solved in PSPACE [15, Theorem 4].

The zeroness problem of D-finite [38] and, more generally, D-algebraic power series [17], [35] is known to be decidable, but its computational complexity has not been investigated.

A natural problem related to the zeroness problem is the so-called *Skolem problem*, which asks whether a given sequence  $a_n$  has a zero, i.e., whether for some  $n$  we have  $a_n = 0$ . As a corollary of the constructions used to prove Theorem 3, it follows that the Skolem problem for polyrec sequences is PSPACE-hard. Only NP-hardness was formerly known, and already for linear recursive sequences [6, Corollary 2.1]. Decidability of the Skolem problem for linear recursive sequences is a long-standing open problem (c.f. the survey paper [26]). It is interesting to notice that those lower bounds are obtained already on the fixed field with two elements  $\{0, 1\}$ , and are thus of a combinatorial rather numerical nature. The Skolem problem for weighted automata over  $\mathbb{Q}$  (that generalise linear recursive sequences) is undecidable [28].

## II. PRELIMINARIES

By  $\mathbb{N}$  we denote the set of nonnegative integers. We denote an arbitrary field by  $\mathbb{F}$ , and we use 0 and 1 to denote the zero, resp., one elements thereof. Example fields of interest in this paper are: rationals  $\mathbb{Q}$ ; and the two-element field  $\mathbb{F}_2$ . A *sequence* over a *domain*  $\mathbb{D}$  is a function  $u: \mathbb{N} \rightarrow \mathbb{D}$ . The sequences considered in this work are over domains that have a field structure, like rationals  $\mathbb{Q}$ . We use bold-face letters as a short-hand for sequences, e.g.,  $\mathbf{u} = \langle u_n \rangle_{n \in \mathbb{N}}$ .

In this paper we work with multivariate polynomials and rational functions. The (combined) *degree* of a monomial  $x_1^{d_1} \cdots x_k^{d_k}$  is  $d_1 + \cdots + d_k$  and the degree of a polynomial  $P \in \mathbb{Q}(x_1, \dots, x_k)$ , written  $\deg P$ , is the maximum degree of monomials appearing in it. A *rational function* is a formal fraction of two polynomials, where the denominator is required to be non-zero. The degree of a rational function is the maximum of the degrees of the numerator and the denominator. Recall that for any field  $\mathbb{F}$  and a set of variables  $x_1, \dots, x_n$ , polynomials over  $x_1, \dots, x_n$  form the ring

$\mathbb{F}[x_1, \dots, x_n]$ , while rational functions over  $x_1, \dots, x_n$  form the field  $\mathbb{F}(x_1, \dots, x_n)$ .

The *value* of a rational function  $P/Q$  at a point  $\mathbf{x} = (x_1, \dots, x_k)$  is the quotient of values  $P(\mathbf{x})$  and  $Q(\mathbf{x})$ . It is well-defined if  $Q(\mathbf{x}) \neq 0$ , and if  $Q(\mathbf{x}) = 0$  we say that  $P/Q$  has a *singularity* at  $\mathbf{x}$ . However, it may happen that at a singular point  $P/Q$  still can be assigned a value in a unique reasonable way. Namely, if  $P(\mathbf{x}) = Q(\mathbf{x}) = 0$  and the limit  $\lim_{x \rightarrow \mathbf{x}} P(x)/Q(x)$  exists then the singularity at  $\mathbf{x}$  is *removable* and one can define  $(P/Q)(\mathbf{x}) = \lim_{x \rightarrow \mathbf{x}} P(x)/Q(x)$ . Note that this can be done if a field allows to take limits, e.g., over  $\mathbb{Q}$ .

The value of  $P/Q$  at  $\mathbf{x}$  where we have a removable singularity can be computed effectively. One may restrict to a line  $L$  through  $\mathbf{x}$  and an arbitrarily chosen point  $\mathbf{y}$  such that  $Q(\mathbf{y}) \neq 0$ . By substituting the equations of  $L$  to  $P$  and  $Q$  one obtains univariate polynomials. The value of their quotient at  $\mathbf{x}$  exists since we assume that the singularity is removable and it can be computed e.g. using the de l'Hospital's rule, formally differentiating both polynomials unless the substitution of  $\mathbf{x}$  gives non-zero value of the denominator.

The computational aspects of multivariate polynomials, in particular their representation on input to algorithms, are explained in Sec. V, as they will be of no concern in Sections III and IV.

## III. RATIONAL RECURSIVE SEQUENCES

We start with the central definitions, which were already discussed in Section I.

**Definition 4.** A sequence  $\mathbf{u}^{(1)}$  over a field  $\mathbb{F}$  is *rational recursive* (or *ratrec* for short) of *dimension*  $k$  and *degree*  $D$  if there exist auxiliary sequences  $\mathbf{u}^{(2)}, \dots, \mathbf{u}^{(k)}$  over  $\mathbb{F}$  and rational functions  $P_1, \dots, P_k \in \mathbb{F}(x_1, \dots, x_k)$  of degree at most  $D$  such that for all  $n \in \mathbb{N}$ , we have

$$\begin{cases} u_{n+1}^{(1)} &= P_1(u_n^{(1)}, \dots, u_n^{(k)}), \\ &\vdots \\ u_{n+1}^{(k)} &= P_k(u_n^{(1)}, \dots, u_n^{(k)}). \end{cases} \quad (1)$$

A sequence  $\mathbf{u}$  over a field  $\mathbb{F}$  is *polynomially recursive* (or *polyrec* for short) if it satisfies the same definition above where  $P_1, \dots, P_k$  are taken as polynomials in  $\mathbb{F}[x_1, \dots, x_k]$ .

In what follows we assume that all singularities of rational functions which occur in the process of determining the values of a ratrec sequence for chosen initial conditions are *removable*.

For instance, the sequence of *Catalan numbers*  $C_n = \frac{1}{n+1} \cdot \binom{2n}{n}$  is ratrec. This can be seen in several ways. For example, they satisfy the recurrence  $C_{n+1} = \frac{2(2n+1)}{n+2} \cdot C_n$ , giving rise to the following ratrec system:

$$\begin{cases} u_{n+1} &= \frac{2(2v_n+1)}{v_n+2} \cdot u_n, \\ v_{n+1} &= v_n + 1. \end{cases}$$

$$\frac{P}{Q} = \frac{R}{S} \text{ iff } PS = RQ$$

More generally, any P-recursive sequence  $a_n$  is ratrec. A sequence is P-recursive [34, Sec. 6.4] if it satisfies a single recursion of the form

$$P_0(n) \cdot a_n + P_1(n) \cdot a_{n+1} + \dots + P_d(n) \cdot a_{n+d} = 0, \quad (2)$$

where  $P_0, \dots, P_d \in \mathbb{Q}[n]$  are polynomials of the index variable  $n$ . This is readily transformed into the ratrec system

$$\begin{cases} u_{n+1}^{(d)} &= -\frac{P_0(v_n)}{P_d(v_n)} \cdot u_n^{(0)} - \dots - \frac{P_{d-1}(v_n)}{P_d(v_n)} \cdot u_n^{(d-1)}, \\ u_{n+1}^{(d-1)} &= u_n^{(d)}, \\ &\vdots \\ u_{n+1}^{(0)} &= u_n^{(1)}, \\ v_{n+1} &= v_n + 1. \end{cases}$$

may need to take a larger  $d' > d$  in order to ensure  $P_d(v_n) \neq 0$  for all  $n \geq d'$ .

Assuming  $v_0 = 0$  and  $u_0^{(0)} = a_0, \dots, u_0^{(d)} = a_d$ , it is immediate to verify  $v_n = n$  and  $u_n^{(0)} = a_n, \dots, u_n^{(d)} = a_{n+d}$  for every  $n \in \mathbb{N}$ .

**F:** I would consider using the name holonomic sequences. I know it's incorrect but I think most people heard about this class from Joel+Ben who write holonomic. We could explain in the introduction that this is equivalent.

The family of ratrec sequences strictly includes both P-recursive sequences and polyrec sequences. As an example consider the sequence  $u_n = 2^{2^n} + C_n$ . On the one hand, this sequence is certainly ratrec because it is the sum of a polyrec and a P-recursive sequence (which are ratrec) and ratrec sequences are closed under sum. On the other hand,  $u_n$  is not P-recursive since it grows asymptotically faster than any P-recursive sequence (every P-recursive sequence is in  $O((n!)^\gamma)$  for some constant  $\gamma \in \mathbb{R}$  [23, Proposition 3.11]). Further,  $u_n$  is also not polyrec, because  $2^{2^n}$  is polyrec,  $C_n$  is not [13, Corollary 8], and polyrec sequences are closed under sum.

In [13, Theorem 11], the following property of polyrec sequences is proved: if  $\mathbf{u}$  is polyrec, then there exists  $m \in \mathbb{N}$  and a *cancelling polynomial*  $P \in \mathbb{Q}[y_0, \dots, y_m]$ , that is, a non-zero polynomial such that

$$P(u_n, u_{n+1}, \dots, u_{n+m}) = 0 \quad \text{for all } n \in \mathbb{N}.$$

The proof of [13, Theorem 11] in fact operates on a field of rational functions over  $\mathbb{Q}$  and thus readily works for ratrec sequences as well. Thus we obtain:

**Theorem 5** (follows from [13]). *Every ratrec sequence admits a cancelling polynomial.*

In [13, Theorem 16] it is shown that the sequence  $u_n = n^n$  has no cancelling polynomial, and hence is not polyrec. By Theorem 5, it is also not ratrec.

#### IV. TRANSCENDENCE DEGREES

In this section we prove the main result of the paper, previously announced in the introduction as Theorem 1. More precisely, we show that a ratrec definition can be translated to a simple ratrec one with an exponential recursion depth:

$F_n^{(i)}$  becomes less and less free ... poles may appear

univariate so eventually non-zero

whenever  $R$  is defined

**Theorem 6.** Let  $\mathbf{u}^{(1)}$  be a rational-recursive sequence. There exists a rational function  $R \in \mathbb{Q}(y_0, \dots, y_m)$  s.t.

$$u_{n+m+1}^{(1)} = R(u_n^{(1)}, u_{n+1}^{(1)}, \dots, u_{n+m}^{(1)}), \text{ for all } n \in \mathbb{N}.$$

Moreover, if  $\mathbf{u}^{(1)}$  is of dimension  $k$  and degree  $D$ , then  $m$  can be bounded from above by  $k + (kD)^{k^3}$ .

Before we proceed to the proof, let us note that Theorem 6 can be regarded as a strengthening of Theorem 5 in the following sense. If we write  $R(y_0, \dots, y_m) = \frac{A(y_0, \dots, y_m)}{B(y_0, \dots, y_m)}$ , where  $A, B \in \mathbb{Q}[y_0, \dots, y_m]$ , then Theorem 6 shows that the following polynomial is cancelling for  $\mathbf{u}^{(1)}$ :

$$P(y_0, \dots, y_m, y_{m+1}) = y_{m+1} \cdot B(y_0, \dots, y_m) - A(y_0, \dots, y_m).$$

Thus, Theorem 6 shows (and in fact, is equivalent to) that every ratrec sequence admits a cancelling polynomial that is linear in the last variable.

The remainder of this section is devoted to the proof of Theorem 6. In particular, the first part of the theorem (existence) will be proved in Sec. IV-A and the concrete bound on the depth  $m$  will be proved in Sec. IV-B. Let us fix a ratrec system as in Definition 4 of dimension  $k$  and degree  $D$ . We shall use  $\mathbf{x}$  as a short-hand for the vector of variables  $x_1, \dots, x_k$ . The assumed system naturally defines  $k$  sequences of rational functions  $F_n^{(1)}, \dots, F_n^{(k)} \in \mathbb{Q}(\mathbf{x})$  where  $F_0^{(i)} = x_i$  and

$$F_{n+1}^{(i)} = P_i(F_n^{(1)}, \dots, F_n^{(k)}), \quad \text{for all } n \in \mathbb{N}. \quad (3)$$

Thus, each sequence  $F_n^{(i)}$  is a ratrec sequence over the field  $\mathbb{Q}(\mathbf{x})$ , and is mapped to the sequence  $u_n^{(i)}$  by an operation that substitutes variables  $x_1, \dots, x_k$  with the initial condition  $u_0^{(1)}, \dots, u_0^{(k)} \in \mathbb{Q}$ .

Note that in the recursive definition of the rational functions  $F_n^{(i)}$  it never happens that a denominator of a fraction turns out to be a zero polynomial. Indeed, if this was the case, then the same would happen in the recursive definition of the sequences  $u_n^{(i)}$ , contradicting the assumption that the considered system defines the sequence  $\mathbf{u}^{(1)}$ .

Let us make a few observations about the sequences  $F_n^{(1)}, \dots, F_n^{(k)}$ . First, a straightforward estimation shows that the degrees of functions  $F_n^{(1)}, \dots, F_n^{(k)}$  grow at most single-exponentially in  $n$ ; the proof is in the appendix.

**Lemma 7.** For  $n \in \mathbb{N}$ , let  $d_n$  be the maximum degree of  $F_n^{(1)}, \dots, F_n^{(k)}$ . Then  $d_n \leq (k \cdot D)^n$ .

The next lemma is a key property implied by the recurrence: if several consecutive elements of the sequence  $F_n^{(i)}$  satisfy some algebraic constraint, then this constraint is also satisfied at every step later in the sequence.

**Lemma 8** (Substitution lemma). If  $Q(y_0, \dots, y_m) \in \mathbb{Q}(y_0, \dots, y_m)$  is a rational function such that  $Q(F_0^{(i)}(\mathbf{x}), \dots, F_m^{(i)}(\mathbf{x})) = 0$ , then we also have  $Q(F_n^{(i)}(\mathbf{x}), \dots, F_{n+m}^{(i)}(\mathbf{x})) = 0$  for all  $n \in \mathbb{N}$ .

Polynomial?   
 If  $Q(y_0, \dots, y_m) \in \mathbb{Q}(y_0, \dots, y_m)$  is a rational function such that  $Q(F_0^{(i)}(\mathbf{x}), \dots, F_m^{(i)}(\mathbf{x})) = 0$ , then we also have  $Q(F_n^{(i)}(\mathbf{x}), \dots, F_{n+m}^{(i)}(\mathbf{x})) = 0$  for all  $n \in \mathbb{N}$ .   
  $\rightarrow$  whenever the latter is well-def.!!   
  $\rightarrow$  one is well defined ??



$$F_1^{(i)} = P_i(x_1, \dots, x_k)$$

*Proof.* By assumption we have

$$Q(F_0^{(i)}(\mathbf{x}), \dots, F_m^{(i)}(\mathbf{x})) = 0. \quad (4)$$

Consider the ring homomorphism  $h: \mathbb{Q}[\mathbf{x}] \rightarrow \mathbb{Q}(\mathbf{x})$  that maps the variables  $x_1, \dots, x_k$  to rational functions  $F_1^{(1)}, \dots, F_1^{(k)}$ , respectively. For a rational function  $P/Q$  such that  $h(Q) \neq 0$ , by  $h(P/Q)$  we understand the rational function  $h(P)/h(Q)$  (note that such an extension of  $h$  to  $\mathbb{Q}(\mathbf{x})$  does not have to be a field homomorphism). From the definition of the sequence  $F_n^{(i)}$  it readily follows that

$$h(F_{n+1}^{(i)}) = F_{n+1}^{(i)}, \quad \text{for all } n \in \mathbb{N}.$$

Thus, by applying  $h$  to both sides of (4) we infer that

$$Q(F_1^{(i)}(\mathbf{x}), \dots, F_{m+1}^{(i)}(\mathbf{x})) = 0. \quad \text{whenever defined}$$

We conclude by repeating this reasoning  $n$  times.  $\square$

In the following we introduce some basic terminology about (commutative) fields (c.f. [25, Sec. II.1], [12, Sec. V.3], or [19, Sec. 13.1 and 13.2] for more details). Let  $\mathbb{E}, \mathbb{F}$  be two fields. When  $\mathbb{E} \subseteq \mathbb{F}$  we say that  $\mathbb{F}$  is a *field extension* of  $\mathbb{E}$ , which is called the *base field*. The *degree* of  $\mathbb{F}$  over  $\mathbb{E}$ , written  $\deg_{\mathbb{E}} \mathbb{F}$  is the dimension of  $\mathbb{F}$  as a vector space over the base field  $\mathbb{E}$ . For instance,  $\mathbb{Q}(\sqrt{2})$  has degree 2 over  $\mathbb{Q}$  (its elements can be put in the form  $a + b \cdot \sqrt{2}$ ) and  $\mathbb{Q}(\sqrt[3]{2})$  has degree 3 (its elements can be put in the form  $a + b \cdot \sqrt[3]{2} + c \cdot (\sqrt[3]{2})^2$ ). Field extensions need not have finite degree. For instance,  $\mathbb{Q}(\pi)$  and  $\mathbb{Q}(x)$  are two field extensions of  $\mathbb{Q}$  of infinite degree. The degree is multiplicative:

**Lemma 9** (c.f. [19, Theorem 14]). *Consider field extensions  $\mathbb{E} \subseteq \mathbb{F} \subseteq \mathbb{G}$ . Then,  $\deg_{\mathbb{E}} \mathbb{G} = \deg_{\mathbb{E}} \mathbb{F} \cdot \deg_{\mathbb{F}} \mathbb{G}$ .*

An element  $f \in \mathbb{F}$  is *algebraic* over the base field  $\mathbb{E}$  if there is a nonzero polynomial  $P(x) \in \mathbb{E}[x]$  s.t.  $P(f) = 0$ . The field extension  $\mathbb{F}$  is *algebraic* over the base field  $\mathbb{E}$  if every element in  $\mathbb{F}$  is algebraic over  $\mathbb{E}$ .

Let  $\mathbb{F}$  be a field extension of  $\mathbb{E}$ . A subset  $\{f_1, \dots, f_n\} \subseteq \mathbb{F}$  of elements of  $\mathbb{F}$  is *algebraically independent* over  $\mathbb{E}$  if there is no nonzero polynomial  $P(x_1, \dots, x_n) \in \mathbb{E}[x_1, \dots, x_n]$  such that  $P(f_1, \dots, f_n) = 0$ . The *transcendence degree* of  $\mathbb{F}$  over  $\mathbb{E}$ , denoted  $\text{tr deg}_{\mathbb{E}} \mathbb{F}$ , is the largest number of elements of  $\mathbb{F}$  which are algebraically independent over  $\mathbb{E}$ . Note that  $\mathbb{F}$  is algebraic over  $\mathbb{E}$  if and only if  $\text{tr deg}_{\mathbb{E}} \mathbb{F} = 0$ . Like the algebraic degree is additive, the transcendence degree is additive:

**Lemma 10** (c.f. [12, Corollary to Theorem 4, A.5.111]). *Consider field extensions  $\mathbb{E} \subseteq \mathbb{F} \subseteq \mathbb{G}$ . Then,  $\text{tr deg}_{\mathbb{E}} \mathbb{G} = \text{tr deg}_{\mathbb{E}} \mathbb{F} + \text{tr deg}_{\mathbb{F}} \mathbb{G}$ .*

In the following we will always take as the base field  $\mathbb{E} = \mathbb{Q}$ , in which case we will write just  $\text{tr deg } \mathbb{F}$  instead of  $\text{tr deg}_{\mathbb{Q}} \mathbb{F}$ . For example,  $\text{tr deg } \mathbb{Q}(\sqrt{2}) = 0$  because  $\sqrt{2}$  is an algebraic number over  $\mathbb{Q}$ ,  $\text{tr deg } \mathbb{Q}(\sqrt{2}, \pi) = 1$  because  $\pi$  is a transcendental number, and  $\text{tr deg } \mathbb{Q}(x_1, \dots, x_n) = n$ .

Given a field extension  $\mathbb{F}$  over  $\mathbb{E}$  and elements  $f_1, \dots, f_n \in \mathbb{F}$ , let  $\mathbb{E}(f_1, \dots, f_n)$  be the smallest field extension over  $\mathbb{E}$

containing  $f_1, \dots, f_n$ . If  $\mathbb{F} = \mathbb{E}(f_1, \dots, f_n)$ , then we say that  $\mathbb{F}$  is *finitely generated over  $\mathbb{E}$*  (with generators  $f_1, \dots, f_n$ ).

The motivation to look at field extensions is that a ratrec system naturally defines the following sequence of field extensions

$$\mathbb{Q} \subseteq \mathbb{F}_0 \subseteq \mathbb{F}_1 \subseteq \dots \subseteq \mathbb{Q}(\mathbf{x}), \quad (5)$$

where  $\mathbb{F}_0 = \mathbb{Q}(x_1)$  and  $\mathbb{F}_{n+1} = \mathbb{F}_n(F_{n+1}^{(1)}(\mathbf{x}))$  for  $n \in \mathbb{N}$ .

#### A. Ascending sequences of field extensions

In this section we prove the following Noether-like result.

**Theorem 11.** *Consider any ascending sequence of field extensions of the form*

$$\mathbb{Q} \subseteq \mathbb{F}_0 \subseteq \mathbb{F}_1 \subseteq \dots \subseteq \mathbb{Q}(x_1, \dots, x_k).$$

*The sequence eventually stabilises, that is, there exists  $n_0$  such that  $\mathbb{F}_{n_0} = \mathbb{F}_{n_0+1} = \mathbb{F}_{n_0+2} = \dots$ .*

The crucial reason for the result above is that the number  $k$  of variables is fixed. In the proof of Theorem 11 we use the following result on finitely generated extensions.

**Lemma 12** (c.f. [12, A.5.118, Cor. 3]). *If  $\mathbb{G}$  is a finitely generated extension over  $\mathbb{E}$ , then every subextension  $\mathbb{E} \subseteq \mathbb{F} \subseteq \mathbb{G}$  of  $\mathbb{G}$  over  $\mathbb{E}$  is also finitely generated.*

*Proof of Theorem 11.* First of all, observe that

$$\text{tr deg } \mathbb{F}_n \leq \text{tr deg } \mathbb{Q}(x_1, \dots, x_k) = k, \quad \text{for all } n.$$

Hence, there is  $n_1$  such that  $\text{tr deg } \mathbb{F}_{n_1} = \text{tr deg } \mathbb{F}_{n_1+1} = \dots$ . Let  $\mathbb{F}_{\infty} := \bigcup_{n=0}^{\infty} \mathbb{F}_n$  and consider the ascending sequence

$$\mathbb{F}_{n_1} \subseteq \mathbb{F}_{n_1+1} \subseteq \dots \subseteq \mathbb{F}_{\infty}. \quad (6)$$

We have  $\text{tr deg } \mathbb{F}_{n_1} = \text{tr deg } \mathbb{F}_{n_1+i}$  for all  $i > 0$ , and, by Lemma 10,  $\text{tr deg}_{\mathbb{F}_{n_1}} \mathbb{F}_{n_1+i} = 0$ , i.e.,  $\mathbb{F}_{n_1+i}$  is algebraic over  $\mathbb{F}_{n_1}$ . Moreover,  $\mathbb{F}_{\infty}$  is also algebraic over  $\mathbb{F}_{n_1}$  because any element of  $\mathbb{F}_{\infty}$  belongs to some  $\mathbb{F}_{n_1+i}$ . Since  $\mathbb{F}_{n_1} \subseteq \mathbb{Q}(x_1, \dots, x_k)$  is a finitely generated extension of  $\mathbb{F}_{n_1}$  and  $\mathbb{F}_{n_1} \subseteq \mathbb{F}_{\infty} \subseteq \mathbb{Q}(x_1, \dots, x_k)$  is a subextension of  $\mathbb{F}_{n_1}$ , by Lemma 12 we have that  $\mathbb{F}_{\infty}$  is also a finitely generated extension of  $\mathbb{F}_{n_1}$ . In other words, there are generators  $f_1, \dots, f_m \in \mathbb{F}_{\infty}$  such that

$$\mathbb{F}_{\infty} = \mathbb{F}_{n_1}(f_1, \dots, f_m).$$

Since the generators  $f_1, \dots, f_m$  are algebraic over  $\mathbb{F}_{n_1}$ ,  $\mathbb{F}_{\infty}$  is an algebraic extension of finite degree over  $\mathbb{F}_{n_1}$  by Lemma 9. (Concretely, an upper bound for the degree is the product of the degrees of minimal polynomials of the generators  $f_1, \dots, f_m$ .) It follows that the sequence in (6) is an ascending sequence of vector subspaces of  $\mathbb{F}_{\infty}$ , where we treat  $\mathbb{F}_{\infty}$  as a vector space over  $\mathbb{F}_{n_1}$ . Since the dimension of  $\mathbb{F}_{\infty}$  as a vector space over  $\mathbb{F}_{n_1}$  is finite, this sequence must eventually stabilize at  $\mathbb{F}_{n_0}$  for some  $n_0 \geq n_1$ .  $\square$

We now prove the existence part of Theorem 6 using Theorem 11.

$$\begin{aligned} \text{num } F_{m+1}^{(1)}(x) &= \text{Poly}(\text{num } F_0^{(1)}(x), \text{den } F_0^{(1)}(x), \dots) \\ \text{den } F_{m+1}^{(1)}(x) &= \text{Poly}(\quad\quad\quad " \quad\quad\quad ) \end{aligned} \left| \begin{array}{l} \text{If polyrec, then} \\ R \text{ always evaluates} \\ \text{to a polynomial} \end{array} \right.$$

By Lemma 7 and since  $j_1 \leq k$  and  $r \leq k$ , we have, as required,

$$\begin{aligned} j_0 &\leq j_1 + (\deg F_0 \cdots \deg F_{j_1})^r \\ &\leq k + ((kD)^0 \cdots (kD)^k)^k \\ &\leq k + (kD)^{k^3}. \end{aligned} \quad \square$$

We are ready to provide the proof of the quantitative bound promised in Theorem 6.

*Proof (of the second part of Theorem 6).* It suffices to observe that  $m$  in the proof the first part of Theorem 6 can be bounded by  $k + (kD)^{k^3}$  thanks to Lemma 16.  $\square$

We finish this section by giving an example that shows that in the proof of Lemma 16, it may happen that  $j_1 < j_0$ , that is, after the stabilisation of the transcendence degree, there can be several non-trivial algebraic extensions until the fields themselves stabilise. Consider the poly-rec system

$$\begin{cases} u_{n+1}^{(1)} &= (u_n^{(1)})^2 + (u_n^{(2)})^2, \\ u_{n+1}^{(2)} &= u_n^{(1)} + u_n^{(2)}. \end{cases}$$

We have  $F_0^{(1)} = x_1, F_0^{(2)} = x_2$ , then  $F_1^{(1)} = x_1^2 + x_2^2, F_1^{(2)} = x_1 + x_2$  and  $F_2^{(1)} = (x_1^2 + x_2^2)^2 + (x_1 + x_2)^2$ . The chain (5) starts with

$$\mathbb{Q} \subseteq \mathbb{F}_0 = \mathbb{Q}(x_1) \subseteq \mathbb{F}_1 = \mathbb{F}_0(x_1^2 + x_2^2) = \mathbb{Q}(x_1, x_2^2) \subseteq \mathbb{F}_2.$$

Note that  $\text{tr deg } \mathbb{F}_0 = 1$  and  $\text{tr deg } \mathbb{F}_1 = 2$ , which is the maximum value. However, the next extension  $\mathbb{F}_1 \subseteq \mathbb{F}_2 = \mathbb{F}_1(F_2^{(1)}) = \mathbb{F}_1(x_1 x_2)$  is non-trivial, because  $x_1 x_2$  does not belong to  $\mathbb{Q}(x_1, x_2^2)$ . In fact, it is algebraic of degree 2.

## V. ZERONESS FOR POLYNOMIAL AUTOMATA OVER UNARY ALPHABETS: UPPER BOUND

**Mi:** The following lemma should be moved to Section IV and synchronized with the final choice of definitions for rat-rec.

**Lemma 17.** Let  $\mathbf{u}$  be a sequence over  $\mathbb{Q}$  that is strongly rat-rec of dimension  $k$  and degree  $D$ . Suppose  $u_n = 0$  for all  $n \in \{0, 1, \dots, k + (kD)^{k^3}\}$ . Then  $u_n = 0$  for all  $n \in \mathbb{N}$ .

In this section we utilize the understanding gathered so far, in particular that provided by Lemma 17, to design algorithms for the zeroness problem. We restrict attention to poly-rec sequences in order to avoid the discussion of appearance of accidental zeros in denominators during the computation of a rat-rec sequence. However, as we only rely on Lemma 17, all the results can be naturally generalized to the rat-rec setting assuming that no such accidental zeros occur at all. Note that this is a stronger assumption than that used in the definition of a strongly rat-rec sequence, where we allow accidental zeros in the denominators as long as such zeros can be eliminated by taking a limit in a neighborhood of the initial condition.

### A. Encoding poly-rec systems

In order to speak about computational aspects of poly-rec sequences, we need to fix how they are encoded on input. For robustness, we choose to use arithmetic circuits. Formally, for a fixed field  $\mathbb{F}$ , a polynomial  $P \in \mathbb{F}[x_1, \dots, x_k]$  is encoded by a circuit  $C$  that may use the following gates:

- binary addition and multiplication gates;
- nullary input gates, bijectively labelled with variables  $x_1, \dots, x_k$ ; and
- nullary constant gates, each labelled with an element of  $\mathbb{F}$ .

Note that subtraction can be emulated using addition and multiplication by the constant  $-1$ . One of the gates is designated as the output gate. Given a valuation of variables with elements of  $\mathbb{F}$ , the values of the gates can be computed as expected, and the value yielded by the circuit  $C$  is that computed for the output gate.

**Mi:** In the following, I use the notions of the dimension and the degree of a poly-rec system, rather than of a sequence. Do we like it and adjust the preliminaries accordingly?

**L:** OK for me.

A poly-rec system  $S$ , say the one given by (1), is also encoded using one circuit, however the circuit has now  $k$  output gates, bijectively labelled with  $x'_1, \dots, x'_k$ . We require that the gate  $x'_i$  computes the value  $P^{(i)}(x_1, \dots, x_k)$ . In all considered circuits we assume that output gates have fan-out 0, and no input gate is an output gate at the same time.

Note that in the definitions above, we did not specify the encoding scheme for the elements of the field  $\mathbb{F}$ . When working with an abstract field  $\mathbb{F}$ , we simply measure the running time of algorithms in terms of arithmetic operations on elements of  $\mathbb{F}$ . An example of this is the following lemma, which follows by evaluating the gates in a bottom-up manner.

**Lemma 18.** Given an arithmetic circuit  $C$  over a field  $\mathbb{F}$  and a valuation of the input gates of  $C$  with elements of  $\mathbb{F}$ , the values of all the gates of  $C$  can be computed in polynomial time and using a polynomial number of operations in  $\mathbb{F}$ .

We will also work with specific fields:  $\mathbb{F} = \mathbb{F}_p$  for a prime  $p$  (the field of residues modulo  $p$ ) and  $\mathbb{F} = \mathbb{Q}$ . In the first case, an element of  $\mathbb{F}_p$  is represented by an integer with at most  $\log p$  bits. In the second case, a rational number is represented using an irreducible fraction  $\frac{a}{b}$ , where  $a$  is an integer and  $b$  is a positive integer, which takes  $\mathcal{O}(\log |a| + \log |b|)$  bits. In both cases, any arithmetic operation on two field elements can be done in time polynomial in the sizes of their encodings.

Obviously, arithmetic circuits is but one possible choice of a reasonable encoding scheme for poly-rec systems. Arguably, it is a rather concise scheme: note that a circuit with  $s$  gates can encode polynomials of degree as high as  $2^{s-1}$ . However, as the next lemma shows, one can transform in polynomial time a poly-rec system encoded through a circuit into another poly-rec system that defines essentially the same sequence—just with each entry repeated a number of times—but the new

poly-rec system uses only very simple equations. Therefore, the issue of encoding those equations becomes immaterial.

**Lemma 19.** *Suppose that  $\mathbf{u}$  is a poly-rec sequence over a field  $\mathbb{F}$ , defined by a poly-rec system  $S$  of dimension  $k$  that is encoded by a circuit  $C$  with  $s$  gates. Then given  $C$  and the initial condition  $u_0^{(1)}, \dots, u_0^{(k)}$  for  $S$ , one can compute a poly-rec system  $S'$  of dimension  $k' = \mathcal{O}(s^2)$  and its initial condition  $v_0^{(1)}, \dots, v_0^{(k')}$  so that the following conditions hold:*

- $S'$  uses only equations of the form:
  - $v_{n+1}^{(a)} = v_n^{(b)}$ ;
  - $v_{n+1}^{(a)} = v_n^{(b)} + v_n^{(c)}$ ;
  - $v_{n+1}^{(a)} = v_n^{(b)} \cdot v_n^{(c)}$ ; and
  - $v_{n+1}^{(a)} = q$ , where  $q$  is a constant that appears in  $C$ .
- For all  $n \in \mathbb{N}$ , we have  $v_n^{(1)} = u_{\lfloor n/\ell \rfloor}^{(1)}$ , where  $\ell = s - 2k + 1$ .

The computation takes polynomial time and uses a polynomial number of operations in  $\mathbb{F}$ .

*Proof.* Let  $\ell = s - 2k + 1$ . We first transform  $C$  into another circuit  $C'$  with  $\mathcal{O}(s^2)$  gates that computes the same polynomials (and thus also encodes the system  $S$ ), but has the following property: The gates of  $C$  can be partitioned into  $\ell + 1$  layers  $L_0, L_1, \dots, L_\ell$  so that:

- $L_0$  consists of the  $k$  input gates;
- $L_\ell$  consists of the  $k$  output gates; and
- for each  $i \in \{1, \dots, \ell\}$ , the gates in  $L_i$  depend only on the gates in  $L_{i-1}$ .

We allow  $C'$  to also use unary *copy gates*, which just copy the value of another gate.

The construction of  $C'$  proceeds as follows. Let  $g_1, \dots, g_s$  be any topological ordering of the gates of  $C$ , that is, an ordering such that for each gate  $g_i$ , the gates on which  $g_i$  depends appear before  $g_i$  in the ordering. We may assume that  $g_1, g_2, \dots, g_k$  are the input gates labelled with  $x_1, \dots, x_k$ , respectively, and  $g_{s-k+1}, \dots, g_s$  are the output gates labelled with  $x'_1, \dots, x'_k$ , respectively. Gates that are neither input nor output shall be called *internal*; these are gates with indices between  $k + 1$  and  $s - k$ .

We now construct the gates of  $C'$ . Each gate of  $C'$  will be denoted as  $h_{i,j}$ , where  $i \in \{1, \dots, s\}$  is the index of the corresponding gate of  $C$  and  $j \in \{0, \dots, \ell\}$  is the index of the layer to which  $h_{i,j}$  belongs. The gates are constructed as follows:

- For each input gate  $g_i$  ( $i \in \{1, \dots, k\}$ ), construct an input gate  $h_{i,0}$  (also labelled with  $x_i$ ) and, for each  $j \in \{1, \dots, \ell - 1\}$ , a copy gate  $h_{i,j}$  that copies the value of  $h_{i,j-1}$ .
- For each internal gate  $g_i$  ( $i \in \{k+1, \dots, s-k\}$ ), construct a gate  $h_{i,i-k}$  of the same type as  $g_i$ , and, for each  $j \in \{i-k+1, \dots, \ell - 1\}$ , a copy gate  $h_{i,j}$  that copies the value of  $h_{i,j-1}$ . If  $g_i$  is an addition gate that computes the sum of values of  $g_a$  and  $g_b$ , then  $h_{i,i-k}$  is also an addition gate that computes the sum of values of  $h_{a,i-k-1}$  and  $h_{b,i-k-1}$ . The cases when  $g_i$  is a multiplication gate or a constant gate are handled analogously.

- For each output gate  $g_i$  ( $i \in \{s-k+1, \dots, s\}$ ), construct an output gate  $h_{i,\ell}$  of the same type as  $g_i$ . The wiring of  $h_{i,\ell}$  is defined in the same way as for internal gates.

It is clear by construction that for each  $i \in \{1, \dots, s\}$ , all the gates  $h_{i,\cdot}$ 's constructed in  $C'$  for  $g_i$  compute the same polynomial in the variables  $x_1, \dots, x_k$  as those computed by the gates  $g_i$ 's in  $C$ . In particular,  $C'$  computes the same  $k$ -tuple of polynomials and thus defines the same poly-rec system  $S'$ . Also, note that  $C'$  has  $\mathcal{O}(s^2)$  gates and the gates in layer  $L_j$  depend only on gates from  $L_{j-1}$ , for  $j \in \{1, \dots, \ell\}$ . Hence,  $C'$  has all the requested properties.

We now proceed to the construction of the system  $S'$  based on the circuit  $C'$ . Let  $s'$  be the number of gates of  $S'$  and let us reindex those gates as  $h_1, \dots, h_{s'}$ , where  $h_1, \dots, h_k$  are the input gates (labelled with  $x_1, \dots, x_k$ , respectively),  $h_{s'-k+1}, \dots, h_{s'}$  are the output gates (labelled with  $x'_1, \dots, x'_k$ , respectively), and layers  $L_1, \dots, L_{\ell-1}$  are placed in the ordering contiguously, one after the other in the natural order. The system  $S'$  will have dimension  $s' - k$ , hence it will define sequences  $\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(s'-k)}$ ; these sequences will respectively correspond to the non-output gates  $h_1, \dots, h_{s'-k}$ .

Let us start with constructing the initial condition for  $S'$ . For each  $i \in \{1, \dots, s' - k\}$ , let  $v_0^{(i)} \in \mathbb{F}$  be the value of the gate  $g_i$  when  $C'$  is applied on the (given) initial condition  $u_0^{(1)}, \dots, u_0^{(k)}$  for  $S$ . Note that values  $v_0^{(1)}, \dots, v_0^{(s'-k)}$  can be computed within the requested complexity using Lemma 18.

It remains to give recursive equations for sequences  $\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(s'-k)}$ . Consider  $a \in \{1, \dots, s' - k\}$  and let  $a' = a$  if  $a > k$ , and  $a' = s' - k + a$  if  $a \leq k$ . Note that  $h_{a'}$  is not an input gate, hence it is a copy gate, an addition gate, a multiplication gate, or a constant gate. If  $h_{a'}$  is a copy gate, say copying the value of  $h_b$  for  $b < a'$ , then we write equation

$$v_{n+1}^{(a)} = v_n^{(b)}.$$

Note here that  $b \leq s' - k$ , because  $h_b$  does not belong to the last layer  $L_\ell$ . Equations for addition, multiplication, and constant gates are produced similarly as follows:

$$\begin{aligned} v_{n+1}^{(a)} &= v_n^{(b)} + v_n^{(c)}; \\ v_{n+1}^{(a)} &= v_n^{(b)} \cdot v_n^{(c)}; \\ v_{n+1}^{(a)} &= q. \end{aligned}$$

The following claim follows from the construction by a straightforward induction on  $n$ ; we leave the easy verification to the reader.

**Claim 1.** Let  $i \in \{1, \dots, s' - k\}$  and suppose  $h_i \in L_j$ , where  $j \in \{0, 1, \dots, \ell - 1\}$ . Further, let  $n = m\ell + r$ , where  $n, m \in \mathbb{N}$  and  $r \in \{0, 1, \dots, \ell - 1\}$ . Then the value of  $v_n^{(i)}$  is equal to:

- the value of the gate  $h_i$  when  $C'$  is applied on  $u_{n-1}^{(1)}, \dots, u_{n-1}^{(k)}$ , provided  $r \leq j$ ; and
- the value of the gate  $h_i$  when  $C'$  is applied on  $u_n^{(1)}, \dots, u_n^{(k)}$ , provided  $r > j$ .

In particular, from Claim 1 for  $i = 1$  it follows that  $v_n^{(1)} = u_{\lfloor n/\ell \rfloor}^{(1)}$  for all  $n \in \mathbb{N}$ . This concludes the proof.  $\square$



## B. Algorithms for zeroness

We now consider the *zeroness problem* for arithmetic circuits over  $\mathbb{Q}$ : given an arithmetic circuit  $C$  over  $\mathbb{Q}$  with one output gate, and a valuation of the input gates of  $C$  with rationals, decide whether the output gate of  $C$  evaluates to 0. Note that we assume that the input rationals, as well as constants used in the circuit, are encoded in binary. Obviously, when evaluating the consecutive gates of the circuit, the binary encodings of the obtained values may become longer and longer. Hence, if one wished to directly apply Lemma 18 to solve the zeroness problem, one would need to estimate how long these binary encodings may get in order to estimate the cost of the arithmetic operations in  $\mathbb{Q}$ . A straightforward estimation gives an upper bound that is single-exponential in the bitlength of the input, hence this naïve approach leads to an EXPTIME algorithm for the zeroness problem for arithmetic circuits over  $\mathbb{Q}$ . However, there is a standard way of improving this complexity using randomization. Namely, if the value of the circuit is non-zero, then with high probability it is also non-zero when computed modulo a random prime  $p$  of magnitude polynomial in the size of the circuit (where a fraction  $\frac{a}{b}$  is replaced with  $ab^{-1}$  in  $\mathbb{F}_p$ ). The advantage is that the value modulo  $p$  can be computed in polynomial time, instead of exponential. See e.g. [31, Theorem 5] for an application of this idea to straight line programs over integers. This yields the following lemma; the proof is provided in the appendix for completeness.

**Lemma 20.** *The zeroness problem for arithmetic circuits over  $\mathbb{Q}$  is in coRP.*

We can now use Lemma 17 in combination with Lemma 20 to tackle the zeroness problem for poly-rec sequences over  $\mathbb{Q}$ : for a given poly-rec sequence  $\mathbf{u}$  over  $\mathbb{Q}$ , defined using a poly-rec system (encoded by an arithmetic circuit over  $\mathbb{Q}$ ) and an initial condition, decide whether  $u_n = 0$  for all  $n \in \mathbb{N}$ .

**Theorem 2.** *The zeroness problem for polynomial recursive sequences over  $\mathbb{Q}$  is in coREXPTIME.  $\subseteq \text{coNEXP}$*

*Proof.* Let  $\mathbf{u}$  be the sequence in question, and let  $C$  be the circuit that encodes a poly-rec system  $S$  defining  $\mathbf{u}$ . Recall the the input consists of  $C$  and the initial condition for the system  $S$ . Let  $k$  and  $D$  be the dimension and the degree of  $S$ , respectively. Observe that if  $N$  is the total bitlength of the input, then  $k \leq N$  and  $D \leq 2^{N-1}$ . The second bound follows by noticing that a straightforward induction proves the following claim: if  $g_1, \dots, g_s$  is any topological ordering of the gates of  $C$ , then  $g_i$  computes a polynomial of degree at most  $2^i$ .

Let  $M = k + (kD)^{k^3}$  and note that  $M \in 2^{\mathcal{O}(N^4)}$ . By Lemma 17, it suffices to verify whether  $u_t = 0$  for each  $t \in \{1, \dots, M\}$  (whether  $u_0 = 0$  can be checked directly from the input). Such a verification, say for index  $t$ , can be done as follows. Construct a circuit  $C_t$  by taking  $t$  disjoint copies of  $C$  and identifying the output gates of the  $i$ th copy with the input gates of the  $(i+1)$ st copy, for each  $i \in \{1, \dots, t-1\}$ .

The input gates of  $C_t$  are the input gates of the first copy of  $C$ , and the only output gate of  $C_t$  is the output gate labelled  $x'_1$  of the last copy of  $C$ . It is clear from the construction that when  $C_t$  is applied to the given initial condition, the output value is equal to  $u_t$ . Therefore, it suffices to solve the zeroness problem for  $C_t$ , which can be done in time  $t^{\mathcal{O}(1)} \leq M^{\mathcal{O}(1)} \leq 2^{\mathcal{O}(N^4)}$  using Lemma 20. Note that by repeating the algorithm  $\log M + 1 = N^{\mathcal{O}(1)}$  times, we can decrease the probability of a false positive to be below  $\frac{1}{2M}$ .

All in all, we apply the algorithm of Lemma 20 for each value of  $t \in \{1, \dots, M\}$ , which results in a time complexity bound of  $M \cdot 2^{\mathcal{O}(N^4)} = 2^{\mathcal{O}(N^4)}$ . By the union bound, the probability of a false positive is smaller than  $M \cdot \frac{1}{2M} = \frac{1}{2}$ .  $\square$

As discussed in Sec. I, an immediate consequence of Theorem 2 is the following.

**Corollary 21.** *The zeroness problem for polynomial automata over unary alphabets is in coREXPTIME.*

We remark that an inspection of the proof of Theorem 2 shows that a better complexity of coRP can be claimed when the degree  $D$  is bounded polynomially in the bitlength of the input (which may be the case for encodings less concise than by arithmetic circuits) and the dimension  $k$  is a fixed constant. Indeed, then  $M = k + (kD)^{k^3}$  is bounded polynomially in  $N$ . The same observation carries over to Corollary 21; here,  $k$  corresponds to the number of control states.

Finally, we remark that for finite fields we can claim an improved PSPACE upper bound. Note here that once a field is fixed, we assume that its elements can be encoded in constant space and arithmetic operations on them take constant time, as the addition and multiplication tables can be hard-coded in the algorithm.

**Theorem 22.** *For every fixed finite field  $\mathbb{F}$ , the zeroness problem for polyrec sequences over  $\mathbb{F}$  is in PSPACE.*

*Proof.* Let  $m$  be the cardinality of  $\mathbb{F}$ ; note that  $m$  is a fixed constant. A standard periodicity argument shows that if  $\mathbf{u}$  is a rat-rec sequence of dimension  $k$ , then it is zero if, and only if, it is zero for the first  $m^k$  steps. We can check the latter condition by storing in memory a  $k$ -tuple of values and computing the first  $m^k$  values of the sequence, which takes an amount of space which is polynomial in  $k$ .  $\square$

## VI. ZERONESS FOR POLYNOMIAL AUTOMATA OVER UNARY ALPHABETS: LOWER BOUND

In this section we prove the following lower bound that complements the upper bounds presented in Sec. V.

**Theorem 23.** *For every fixed field  $\mathbb{F}$ , the zeroness problem for polyrec sequences over  $\mathbb{F}$  is PSPACE-hard.*

The lower bound claimed in the introduction follows from the theorem above by taking  $\mathbb{F} = \mathbb{Q}$ . Note also that together with Theorem 22, we can conclude that the problem is actually PSPACE-complete for every fixed finite field  $\mathbb{F}$ .

### A. Extended polyrec sequences

In the reductions leading to the lower bound of Theorem 23 it is convenient to construct polyrec sequences according to a definition slightly more general than what we allowed in Definition 4. Namely, the definition of an *extended polyrec system* is the same as before, except that we generalize the format of the  $i$ th equation  $u_{n+1}^{(i)} = P_i(\dots)$  by allowing  $u_{n+1}^{(i)}$  to additionally depend on  $u_{n+1}^{(1)}, \dots, u_{n+1}^{(i-1)}$ . Thus, the  $i$ th equation takes the form:

$$u_{n+1}^{(i)} = P_i(u_n^{(1)}, \dots, u_n^{(k)}, u_{n+1}^{(1)}, \dots, u_{n+1}^{(i-1)}), \quad (8)$$

where now  $P_i$  is a polynomial in  $k+i-1$  variables. This more relaxed definition will help focus on the important aspects of the reduction presented in the rest of this section. The following lemma shows that the modification does not affect the complexity of the zeroness problem.

**Lemma 24.** *Suppose  $\mathbf{u}$  is a sequence defined by an extended polyrec system  $S$  of dimension  $k$ , where each polynomial  $P_i$  is represented by circuit  $C_i$ . Then given the circuits  $C_i$ , one can in polynomial time construct a circuit  $C$  that represents a polyrec system  $S'$  of dimension  $k$  that also defines  $\mathbf{u}$  (with the same initial condition as  $S$ ).*

*Proof.* Let the input gates of circuit  $C_i$  be labelled with  $x_1, \dots, x_k, z_1, \dots, z_{i-1}$ , where variables  $z_1, \dots, z_{i-1}$  respectively correspond to the values  $u_{n+1}^{(1)}, \dots, u_{n+1}^{(i-1)}$  in (8). Construct the circuit  $C$  from the union of circuits  $C_1, \dots, C_k$  by performing the following operations for each  $i \in \{1, \dots, k\}$ :

- Fuse all input gates labelled  $x_i$  in circuits  $C_1, \dots, C_k$  into a single input gate labelled  $x_i$ .
- Fuse the output gate of  $C_i$  with all input gates labelled  $z_i$  in circuits  $C_{i+1}, \dots, C_k$ .

The output gates of  $C$  are the output gates of  $C_1, \dots, C_k$ . (Formally, we assumed that output gates must have fan-out 0, but this can be easily obtained by making a copy of each output gate.) It is straightforward to verify that the polyrec system  $S'$  that  $C$  represents defines the same  $k$ -tuple of sequences as  $S$  under the same initial condition.  $\square$

### B. Reduction

We now proceed to the proof of Theorem 23. Let us fix the field  $\mathbb{F}$ ; in the reduction we will use only two constants from  $\mathbb{F}$ , namely 0 and 1. We reduce from the validity problem for Quantified Boolean Formulas (QBF), which is known to be PSPACE-complete (see, e.g., [27, Theorem 19.1]). Recall that the QBF validity problem amounts to determine whether a given QBF of the form

$$\psi = \exists x_1 \forall x_2 \dots Q_k x_k \varphi(x_1, \dots, x_k) \quad (9)$$

is true, where  $\varphi(x_1, \dots, x_k)$  is quantifier-free, the variables with odd indices are quantified existentially, the remaining variables are quantified universally, and  $Q_k$  is either  $\exists$  or  $\forall$  depending on the parity of  $k$ . Hence, we are given a QBF  $\psi$  and we wish to construct, in polynomial time, a polyrec system  $S$  and its initial condition that define a sequence  $\mathbf{u}$  over

$\mathbb{F}$  such that the zeroness of  $\mathbf{u}$  is equivalent to the invalidity of  $\psi$ . By Lemma 24, it suffices to construct an extended polyrec system  $S$  with this property, where each polynomial  $P_i$  involved is represented by a separate circuit. In the following, the size of an extended polyrec system is the total size of its representation through circuits, which is constructed implicitly.

In the reduction it will be convenient to consider formulas obtained by fixing the truth values of a subset of the bound variables of  $\psi$ . For every  $i \in \{0, \dots, k\}$  and  $c_{i+1}, \dots, c_k \in \{0, 1\}$  we define the formula

$$\psi|_{c_{i+1}, \dots, c_k} = \exists x_1 \forall x_2 \dots Q_i x_i \varphi(x_1, \dots, x_i, c_{i+1}, \dots, c_k),$$

where  $Q_i$  is either  $\exists$  or  $\forall$  depending on the parity of  $i$ . In particular, for  $i = k$  we get back  $\psi$ , and for  $i = 0$  the formula  $\psi|_{c_1, \dots, c_k}$  reduces to the truth value of  $\varphi(c_1, \dots, c_k)$ . We encode a quantified Boolean formula  $\varphi$  into a polynomial  $P_\varphi$  using the following simulation of Boolean operators  $\neg$ ,  $\wedge$  and  $\vee$  by arithmetic operations:

$$\begin{aligned} P_x &= x, \\ P_{\neg \varphi} &= 1 - P_\varphi, \\ P_{\varphi \wedge \psi} &= P_\varphi \cdot P_\psi, \\ P_{\varphi \vee \psi} &= P_{\neg(\neg \varphi \wedge \neg \psi)} = 1 - (1 - P_\varphi)(1 - P_\psi) \end{aligned} \quad (10)$$

For example,  $\tau(x, y) = (x \wedge y) \vee \neg y$  is encoded as  $P_\tau(x, y) = 1 - (1 - xy)y$ . The following straightforward claim shows that with the standard interpretation of 1 and 0 representing true, resp., false, such polynomials evaluate as expected. Note that this claim holds in any fixed field.

**Claim 2.** Let  $\tau(x_1, \dots, x_k)$  be a Boolean formula and  $P_\tau(x_1, \dots, x_k)$  its corresponding polynomial. For every  $c_1, \dots, c_k \in \{0, 1\}$  we have  $P_\tau(c_1, \dots, c_k) \in \{0, 1\}$  and

$$(c_1, \dots, c_k) \models \tau \iff P_\tau(c_1, \dots, c_k) = 1.$$

To ease the notation we will directly write formulas as polynomials; for instance,  $P_\tau(x, y) = (x \wedge y) \vee \neg y$ . All sequences in this section will be over  $\{0, 1\}$  and the involved polynomials will be of the form  $P_\tau$ .

a) *Sequences  $\mathbf{c}^1, \dots, \mathbf{c}^k$ :* The truth valuations of variables  $x_1, \dots, x_k$  will be encoded by sequences  $\mathbf{c}^1, \dots, \mathbf{c}^k$ , where for every  $i$  and  $n$  we have

$$c_n^i = \begin{cases} 0 & \text{if } n \bmod 2^i \text{ is less than } 2^{i-1}, \\ 1 & \text{otherwise.} \end{cases} \quad (11)$$

For example, the first eight values of  $\mathbf{c}^1, \mathbf{c}^2, \mathbf{c}^3$  are

$$\begin{array}{cccccccc} \mathbf{c}^1 & = & 0 & 1 & 0 & 1 & 0 & 1 \\ \mathbf{c}^2 & = & 0 & 0 & 1 & 1 & 0 & 0 \\ \mathbf{c}^3 & = & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

**Claim 3.** For every  $i \geq 1$ , the sequence  $\mathbf{c}^i$  is definable by an extended polyrec system over  $\mathbb{F}$  of size polynomial in  $i$ .

*Proof of the claim.* We proceed by induction on  $i$ . For  $i = 1$ , by definition we have  $c_n^1 = 1 - c_{n-1}^1$ , and thus we let

$$c_n^1 = P(c_{n-1}^1), \text{ with } P(x) = \neg x. \quad (12)$$

Now, suppose  $i > 1$  and we have defined  $\mathbf{c}^{i-1}$ . We start by proving the following equality for every  $n > 1$

$$c_n^i = \begin{cases} 1 - c_{n-1}^{i-1} & \text{if } c_{n-1}^{i-1} = 1 \text{ and } c_n^{i-1} = 0; \\ c_{n-1}^{i-1} & \text{otherwise.} \end{cases} \quad (13)$$

Notice that  $\mathbf{c}^i$  is periodic with period  $2^i$ , i.e.,  $c_n^i = c_{n+2^i}^i$  for all  $n$ . Thus it suffices to prove (13) for  $n \in \{1, \dots, 2^i\}$ . By definition,  $c_{n-1}^{i-1} = 1$  and  $c_n^{i-1} = 0$  hold precisely for two values of  $n \in \{1, \dots, 2^i\}$ , namely for  $n = 2^{i-1}$  and  $n = 2^i$ . Thus (13) is proved since  $c_n^i = 0$  for  $0 \leq n < 2^{i-1}$ ;  $c_n^i = 1$  for  $2^{i-1} \leq n < 2^i$ ; and  $c_{2^i}^i = 0$ .

Using (13) one can determine  $c_n^i$  given  $c_{n-1}^{i-1}$ ,  $c_{n-1}^{i-1}$ , and  $c_n^{i-1}$ :

$$c_n^i = Q(c_{n-1}^{i-1}, c_{n-1}^{i-1}, c_n^{i-1}), \quad (14)$$

where  $Q(x, y, z) = (\neg x \wedge (y \wedge \neg z)) \vee (x \wedge (\neg y \vee z))$ . This follows from (13) and from the fact that  $y \wedge \neg z$  and  $\neg y \vee z$  are mutually exclusive formulas encoding the “if” condition in (13). It is clear that the constructed extended polyrec system is of size polynomial in  $i$ . ■

*b) Sequences  $\mathbf{d}^0, \dots, \mathbf{d}^k$ :* We define sequences  $\mathbf{d}^0, \dots, \mathbf{d}^k$ , where for any  $i \geq 0$  we have:

$$d_0^i = 0, \quad d_n^i = \begin{cases} 0 & \text{if } 2^i \nmid n \\ \llbracket \psi|_{c_{n-1}^{i+1}, \dots, c_{n-1}^k} \rrbracket & \text{otherwise,} \end{cases} \quad (15)$$

where for a closed formula  $\xi$  (i.e., with no free variables)  $\llbracket \xi \rrbracket$  is 1 if  $\xi$  is true and 0 otherwise. Notice that the formula depends on  $\mathbf{c}^1, \dots, \mathbf{c}^k$ . Since  $\mathbf{d}^k$  is the zero sequence if, and only if,  $\psi$  is false, it suffices to show that each  $\mathbf{d}^i$  can be defined by an extended polyrec system of polynomial size.

We proceed by induction on  $i$ . In the base case  $i = 0$ ,

$$d_n^0 = P_\varphi(c_{n-1}^1, \dots, c_{n-1}^k), \quad (16)$$

where  $P_\varphi$  is the polynomial obtained from the quantifier-free formula  $\varphi$  according to the rules in (10). (Notice that  $P_\varphi$  can be represented by an arithmetic circuit of size polynomial in the size of  $\varphi$ —this is where we use the conciseness of representation using circuits.) This fulfills the conditions in (15) since, for  $n > 0$ ,  $d_n^0 = 1$  if  $(c_{n-1}^1, \dots, c_{n-1}^k) \models \varphi$  and  $d_n^0 = 0$  otherwise.

Now, fix  $i \geq 1$  and suppose that  $\mathbf{d}^{i-1}$  is defined. The goal is to define  $\mathbf{d}^i$ . Recall that if  $i$  is odd then  $x_i$  is quantified existentially, and otherwise  $x_i$  is quantified universally.

*Claim 4.* Let  $\otimes_i = \vee$  if  $i$  is odd and  $\otimes_i = \wedge$  if  $i$  is even. For every  $n > 0$  and  $0 < i \leq k$ , we have

$$d_n^i = \begin{cases} d_n^{i-1} \otimes_i d_{n-2^{i-1}}^{i-1} & \text{if } 2^i \mid n \\ 0 & \text{otherwise.} \end{cases} \quad (17)$$

*Proof of the claim.* We may focus only on the case  $2^i \mid n$ . Since  $x_i$  is quantified according to the parity of  $i$ , we have

$$\psi|_{c_{n-1}^{i+1}, \dots, c_{n-1}^k} = \psi|_{0, c_{n-1}^{i+1}, \dots, c_{n-1}^k} \otimes_i \psi|_{1, c_{n-1}^{i+1}, \dots, c_{n-1}^k}.$$

We claim that

$$d_n^{i-1} = \psi|_{1, c_{n-1}^{i+1}, \dots, c_{n-1}^k} \quad \text{and} \quad d_{n-2^{i-1}}^{i-1} = \psi|_{0, c_{n-1}^{i+1}, \dots, c_{n-1}^k}.$$

By (11) and the fact that  $2^i \mid n$ , we get  $c_{n-1}^i = c_{2^{i-1}-1}^i = 1$ , which proves the first equation. For the second equation, we observe that  $c_{(n-1)-2^{i-1}}^i = c_{2^{i-1}-1}^i = 0$  and that  $c_{(n-1)-2^{i-1}}^j = c_{n-1}^j$  for all  $j > i$ . The latter assertion readily follows from  $2^i \mid n$  and (11). ■

As an immediate consequence of Claim 4, we can write

$$d_n^i = S(d_n^{i-1}, d_{n-2^{i-1}}^{i-1}, c_{n-1}^{i-1}, c_n^{i-1}), \quad (18)$$

where  $S(x, y, z, t) = (x \otimes_i y) \wedge (z \wedge \neg t)$  (by recalling that  $c_{n-1}^{i-1} = 1, c_n^{i-1} = 0$  holds if, and only if,  $2^i \mid n$ , where  $n > 0$ ). The issue with this recursive definition is that it requires access to the value  $d_{n-2^{i-1}}^{i-1}$ , which in general is not allowed in a polyrec system for  $i \geq 2$  (not even in the extended variant). This will be addressed in the next section by introducing the last family of recursive sequences.

*c) Sequences  $\mathbf{f}^0, \dots, \mathbf{f}^{k-1}$ :* For every  $1 \leq i \leq k$ , the sequence  $\mathbf{f}^{i-1}$  is defined as

$$f_n^{i-1} = \begin{cases} 0 & \text{if } n \bmod 2^i \text{ is less than } 2^{i-1} \\ d_m^{i-1} & \text{otherwise,} \end{cases}$$

where  $m \leq n$  is the unique number such that  $n - m < 2^{i-1}$  and  $2^{i-1} \mid m$ . Thus,  $\mathbf{f}$  is divided into blocks of length  $2^{i-1}$  of equal elements, where every other block is either filled with zeros, or its value is determined by the value of an appropriate entry  $d_m^{i-1}$ . Observe that in particular, if  $2^i \mid n$  then  $f_{n-1}^{i-1} = d_{n-2^{i-1}}^{i-1}$ . Thus, intuitively, the sequence  $\mathbf{f}^{i-1}$  is a “memory” that allows us to store the relevant value of  $\mathbf{d}^{i-1}$  from  $2^{i-1} - 1$  steps back.

We now proceed to defining sequences  $\mathbf{f}^0, \dots, \mathbf{f}^{k-1}$  using polyrec systems. Observe that  $f_0^{i-1} = 0$  and for  $n > 0$ , we can write

$$f_n^{i-1} = \begin{cases} d_n^{i-1} & \text{if } c_{n-1}^{i-1} = 0 \text{ and } c_n^{i-1} = 1; \\ 0 & \text{if } c_{n-1}^{i-1} = 1 \text{ and } c_n^{i-1} = 0; \\ f_{n-1}^{i-1} & \text{otherwise.} \end{cases} \quad (19)$$

Notice that the value of  $f_n^{i-1}$  is copied from  $f_{n-1}^{i-1}$  unless  $c_{n-1}^{i-1}, c_n^{i-1}$  differ. To conclude, recall from (11) that this happens if, and only if,  $2^{i-1} \mid n$ .

*Claim 5.* For every  $i \geq 1$ , the sequences  $\mathbf{d}^i$  and  $\mathbf{f}^{i-1}$  are definable by extended polyrec systems over  $\mathbb{F}$  of size polynomial in  $i$  and the size of  $\varphi$ .

*Proof of the claim.* Using (19), we may write  $\mathbf{f}^{i-1}$  as an extended polyrec sequence  $f_0^{i-1} = 0$  and, for  $n > 0$ ,

$$f_n^{i-1} = R(c_{n-1}^{i-1}, c_n^{i-1}, d_n^{i-1}, f_{n-1}^{i-1}), \quad (20)$$

where

$$R(x, y, z, t) = (z \wedge (\neg x \wedge y)) \vee (t \wedge ((x \wedge y) \vee (\neg x \wedge \neg y))).$$

In turn, this allows us to rewrite (18) as

$$d_n^i = S(d_{n-1}^{i-1}, f_{n-1}^{i-1}, c_{n-1}^{i-1}, c_n^{i-1}), \quad (21)$$

where  $S$  was defined in (18). Note that (20) and (21) are in the extended polyrec format provided that we write the equations for the  $f^i$ 's after the equations for the  $d^i$ 's, and the latter after the equations for  $c^i$ 's (in order to avoid creating a cyclic dependency). In other words, the final extended polyrec system consists of equations (14), followed by (21), and followed by (20), where each set of equations is numbered naturally according to the indices of sequences.

The involved polynomials  $P_\varphi$ ,  $R$  and  $S$  are all of size polynomial in the input size when represented as arithmetic circuits ( $R$  and  $S$  are even of constant size), and we have a polynomial number of equations. Thus, the definition above is an extended polyrec system of polynomial size. ■

As discussed, Claim 5 finishes the proof of Theorem 23.

In the end, we discuss the Skolem problem: given a sequence  $\mathbf{u}$  to determine whether there is  $n$  such that  $u_n = 0$ . This problem was extensively studied for the class of linear recursive sequences (see e.g. [26]). For linear recursive sequences it is open whether the Skolem problem is decidable, but only NP-hardness is known [6, Corollary 2.1]. For polyrec sequences, decidability of the Skolem problem is also open, but we can improve the lower bound.

**Corollary 25.** *The Skolem problem is PSPACE-hard for polyrec sequences.*

*Proof.* Notice that in the proof of Theorem 23 we define a system of sequences over  $\{0, 1\}$ . It remains to observe that for such sequences the zeroness problem and the Skolem problem reduce to each other. Indeed, the zeroness problem of a sequence  $\mathbf{u}$  over  $\{0, 1\}$  is equivalent to the Skolem problem of  $\mathbf{v}$  defined as  $v_n = 1 - u_n$ . *negation of* □

## VII. CONCLUSION

We believe that ratrec is a natural class of sequences with various promising questions deserving further investigation. Questions about decision problems are more natural for polyrec sequences due to their connection to polynomial automata and the issues with division by 0 in ratrec discussed in the introduction. Nevertheless, as this paper showcases, understanding the properties of ratrec can lead to concrete complexity results for polyrec.

An imminent open problem regarding zeroness concerns closing the complexity gap between our PSPACE lower bound and coREXPTIME upper bound.

**L:** I made a bit more explicit the reference to the complexity gap. It also helps to reiterate two results of the paper.

Another interesting problem is the following: Given a polyrec sequence  $\mathbf{u}$ , say defined by a system of width  $k$ , decide whether  $\mathbf{u}$  can be defined by a system of width smaller than  $k$ . This is a variant of the *register complexity problem* (see

e.g. the recent survey [16]), in the special case of polynomial automata.

**L:** I rephrased this slightly since before it gave the impression that this problem was already studied for polynomial automata.

**F:** It was. But over the max-plus semiring and using the name cost-register automata (which = polynomial automata). I've never seen any interesting result though, one of the reasons is that you can encode the determinisation problem of weighted automata into this problem. I'm not sure we want to explain all of this though ;)

**Mi:** I rephrased the problem again, as it took me some effort to understand what this problem is about. I hope the current formulation is more direct.

Regarding mathematical problems recall that ratrec generalises both polyrec and P-recursive sequences. It is natural to ask what natural sequences, such as  $n^n$ , are not in ratrec?

**L:** This is a bit vague. What about studying the *characterisation problem*: Given a ratrec sequence, decide whether it is polyrec / P-recursive / linrec.

**F:** Do it, it's better

Finally, from the point of view of algebra and algebraic geometry, one would like to ask what are the properties (e.g. dimension) of the set of good initial values for a given ratrec system, i.e., values for which the whole sequence is well-defined. We expect that the answer depends on the field over which the system is defined.

**L:** As future directions if space allows I would like to add zeroness for multidimensional polyrec sequences  
 $u_{n+1,k+1} = P(u_{n,k+1}, u_{n+1,k}, \dots)$

**F:** Ok if you define multidimensional polyrec in at most 3 lines.

## REFERENCES

- [1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.
- [2] Rajeev Alur, Loris D'Antoni, Jyotirmoy V. Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 13–22, 2013.
- [3] Corentin Barloy and Lorenzo Clemente. Bidimensional linear recursive sequences and universality of unambiguous register automata. In *To appear in Proc. of STACS'21*, January 2021.
- [4] Corentin Barloy, Nathanaël Fijalkow, Nathan Lhote, and Filip Mazowiecki. A robust class of linear recurrence sequences. In *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain*, pages 9:1–9:16, 2020.
- [5] M. Benedikt, T. Duff, A. Sharad, and J. Worrell. Polynomial automata: Zeroness and applications. In *Proc. of LICS'17*, pages 1–12, June 2017.
- [6] Vincent D. Blondel and Natacha Portier. The presence of a zero in an integer linear recurrent sequence is np-hard to decide. *Linear Algebra and its Applications*, 351–352:91–98, 2002. Fourth Special Issue on Linear Systems and Control.
- [7] Adrien Boiret, Radosław Piórkowski, and Janusz Schmude. Reducing Transducer Equivalence to Register Automata Problems Solved by “Hilbert Method”. In Sumit Ganguly and Paritosh Pandya, editors, *Proc. of FSTTCS'18*, volume 122 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 48:1–48:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.



- [8] Mikołaj Bojańczyk. The hilbert method for transducer equivalence. *ACM SIGLOG News*, 6(1):5–17, February 2019.
- [9] Mikołaj Bojańczyk and Wojciech Czerwiński. An automata toolbox, Feb 2018.
- [10] Mikołaj Bojańczyk and Janusz Schmude. Some Remarks on Deciding Equivalence for Graph-To-Graph Transducers. In Javier Esparza and Daniel Král', editors, *Proc. of MFCS'20*, volume 170 of *LIPIcs*, pages 19:1–19:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
- [11] Alin Bostan, Arnaud Carayol, Florent Koechlin, and Cyril Nicaud. Weakly-Unambiguous Parikh Automata and Their Link to Holonomic Series. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *Proc. of ICALP'20*, volume 168 of *LIPIcs*, pages 114:1–114:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
- [12] N. Bourbaki. *Algebra II. Elements of Mathematics*. Springer Verlag Berlin Heidelberg, 2003.
- [13] Michaël Cadilhac, Filip Mazowiecki, Charles Paperman, Michał Pilipczuk, and Géraud Sénizergues. On polynomial recursive sequences. In *Proc. of ICALP'20*, volume 168 of *LIPIcs*, pages 117:1–117:17. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2020.
- [14] N. Chomsky and M. P. Schützenberger. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, volume 35 of *Studies in Logic and the Foundations of Mathematics*, pages 118–161. Elsevier, 1963.
- [15] Lorenzo Clemente. On the complexity of the universality and inclusion problems for unambiguous context-free grammars. In Laurent Fribourg and Matthias Heizmann, editors, *Proceedings 8th International Workshop on Verification and Program Transformation and 7th Workshop on Horn Clauses for Verification and Synthesis*, Dublin, Ireland, 25–26th April 2020, volume 320 of *EPTCS*, pages 29–43. Open Publishing Association, 2020.
- [16] Laure Daviaud. Register complexity and determinisation of max-plus automata. *ACM SIGLOG News*, 7(2):4–14, 2020.
- [17] J. Denef and L. Lipshitz. Decision problems for differential equations. *Journal of Symbolic Logic*, 54(3):941–950, 1989.
- [18] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer, 1st edition, 2009.
- [19] David S. Dummit and Richard M. Foote. *Abstract Algebra*. Wiley, 3rd edition, 2003.
- [20] Vesa Halava, Tero Harju, Mika Hirvensalo, and Juhani Karhumäki. Skolem's problem - on the border between decidability and undecidability, 2005.
- [21] T. Harju and J. Karhumäki. The equivalence problem of multitape finite automata. *Theoretical Computer Science*, 78(2):347–355, 1991.
- [22] Manuel Kauers and Peter Paule. *The Concrete Tetrahedron - Symbolic Sums, Recurrence Equations, Generating Functions, Asymptotic Estimates*. Texts & Monographs in Symbolic Computation. Springer, 2011.
- [23] Leonard Lipshitz. D-finite power series. *Journal of Algebra*, 122(2):353–373, 1989.
- [24] J. Müller-Quade and R. Steinwandt. Basic algorithms for rational function fields. *Journal of Symbolic Computation*, 27(2):143–170, 1999.
- [25] Masayoshi Nagata. *Theory of Commutative Fields*. Translations of Mathematical Monographs, Vol. 125. American Mathematical Society, 1993.
- [26] Joël Ouaknine and James Worrell. On linear recurrence sequences and loop termination. *ACM SIGLOG News*, 2(2):4–13, April 2015.
- [27] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [28] Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 1971.
- [29] Arto Salomaa and Marti Soittola. *Automata-theoretic aspects of formal power series*. Texts and Monographs in Computer Science. Springer, 1978.
- [30] Bruno Salvy. D-finiteness: Algorithms and applications. In *Proc. of ISAAC'05*, pages 2–3, New York, NY, USA, 2005. ACM.
- [31] Arnold Schönhage. On the power of random access machines. In *6th International Colloquium on Automata, Languages and Programming, ICALP 1979*, volume 71 of *Lecture Notes in Computer Science*, pages 520–529. Springer, 1979.
- [32] Marcel Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2–3):245–270, 1961.
- [33] Helmut Seidl, Sebastian Maneth, and Gregor Kemper. Equivalence of deterministic top-down tree-to-string transducers is decidable. *J. ACM*, 65(4):21:1–21:30, April 2018.
- [34] Richard P. Stanley and Sergey Fomin. *Enumerative combinatorics*, volume 2 of *Cambridge studies in advanced mathematics*. Cambridge University Press, 1 edition, 2001.
- [35] Joris van der Hoeven. Computing with d-algebraic power series. *Applicable Algebra in Engineering, Communication and Computing*, 30(1):17–49, 2019.
- [36] Tzeng Wen-Guey. On path equivalence of nondeterministic finite automata. *Information Processing Letters*, 58(1):43–46, 1996.
- [37] James Worrell. Revisiting the equivalence problem for finite multitape automata. In Fedor V. Fomin, Rūsiņš Freivalds, Marta Kwiatkowska, and David Peleg, editors, *Proc. of ICALP'13*, pages 422–433, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [38] Doron Zeilberger. A holonomic systems approach to special functions identities. *Journal of Computational and Applied Mathematics*, 32(3):321–368, 1990.

## APPENDIX

In this technical appendix we provide proofs omitted in the main body of the paper.

**Lemma 7.** *For  $n \in \mathbb{N}$ , let  $d_n$  be the maximum degree of  $F_n^{(1)}, \dots, F_n^{(k)}$ . Then  $d_n \leq (k \cdot D)^n$ .*

*Proof.* We proceed by induction on  $n$ . Initially we have  $d_0 = 1$  by definition. By (3),  $F_{n+1}^{(i)}$  is obtained by substituting rational functions  $F_n^{(1)}, \dots, F_n^{(k)}$  of degree at most  $d_n$  into a rational function  $P_i$  of degree at most  $D$ . Let  $P_i = \frac{A}{B}$  be the ratio of two polynomials  $A, B \in \mathbb{Q}[x]$  of degree at most  $D$ . Let  $C \in \mathbb{Q}[x]$  be the least common multiple of all denominators of  $F_n^{(1)}, \dots, F_n^{(k)}$ , and thus of degree at most  $k \cdot d_n$ . We can then write  $F_n^{(1)} = \frac{G^{(1)}}{C}, \dots, F_n^{(k)} = \frac{G^{(k)}}{C}$ , where the numerators  $G^{(1)}, \dots, G^{(k)} \in \mathbb{Q}[x]$  are polynomials of degree also at most  $k \cdot d_n$ . It follows that both  $A(F_n^{(1)}, \dots, F_n^{(k)})$  and  $B(F_n^{(1)}, \dots, F_n^{(k)})$  can be written as rational functions of the form  $\frac{\hat{A}}{C^D}$ , resp.,  $\frac{\hat{B}}{C^D}$ , where the numerators are polynomials  $\hat{A}, \hat{B} \in \mathbb{Q}[x]$  of degree at most  $D \cdot k \cdot d_n$  and the same holds for the common denominator  $C^D \in \mathbb{Q}[x]$ . It follows that  $F_{n+1}^{(i)}$  is a rational function of degree  $d_{n+1} \leq k \cdot D \cdot d_n$ , as required.  $\square$

**Lemma 20.** *The zeroness problem for arithmetic circuits over  $\mathbb{Q}$  is in coRP.*

*Proof.* Let  $C$  be the given arithmetic circuit,  $s$  be the number of gates of  $C$ , and  $B$  be the maximum of absolute values of numerators and denominators of numbers given on input. Note that  $s, \log B \leq N$ , where  $N$  is the total bitlength of the input.

Let  $g_1, \dots, g_s$  be any topological ordering of the gates of  $C$ . We may assume that  $g_s$  is the output gate. Let  $\frac{a_i}{b_i}$  be the value computed at  $g_i$ , where  $a_i$  is an integer,  $b_i$  is a positive integer, and  $a_i, b_i$  are coprime. Then the following claim follows from a straightforward induction on  $i$ :

$$|a_i|, |b_i| \leq (2B)^{2^i} \quad \text{for all } i \in \{1, \dots, s\}.$$

In particular if we denote

$$M := (2B)^{2^s} + 1,$$

then  $a_s$ —the numerator of the value computed at the output gate—is smaller than  $M$ . Note that  $M$  is bounded doubly-exponentially in  $O(N)$ .

We will use the following claim that is implied by classic results about the density of prime numbers (precisely, the Prime Number Theorem and Bertrand's postulate).

*Claim 6.* There exists  $\alpha > 0$  such that for each integer  $n > 1$ , an integer chosen uniformly at random from the range  $\{n, n+1, \dots, 2n-1\}$  is prime with probability at least  $\frac{\alpha}{\log n}$ .

Let

$$m := \max \left( B + 1, \left\lceil \frac{4}{\alpha} \cdot \log M \right\rceil \right).$$

Note that  $m \in 2^{\mathcal{O}(N)}$ . By Claim 6, a random integer from the range  $\{m, m+1, \dots, 2m-1\}$  is prime with probability at least  $\frac{\alpha}{\log m}$ . Therefore, if we independently sample  $\lceil \frac{2}{\alpha} \cdot \log m \rceil$  integers from this range, then at least one element of the sample is prime with probability at least

$$1 - \left( 1 - \frac{\alpha}{\log m} \right)^{\frac{2}{\alpha} \cdot \log m} \geq 1 - e^{-2} > \frac{3}{4}.$$

We may now apply the AKS primality test [1] on each element of the sample, thus identifying which of them are primes in time polynomial in  $\log m = \mathcal{O}(N)$ . The sample contains no primes with probability at most  $\frac{1}{4}$ , and otherwise the first of them, call it  $p$ , is a prime sampled uniformly at random from primes within the range  $\{m, m+1, \dots, 2m-1\}$ . In case there are no primes in the sample, the algorithm terminates and outputs the positive outcome; this is a false positive with probability at most  $\frac{1}{4}$ .

Hence, from now on we work under the assumption that the prime  $p$  has been successfully sampled. The following observation is the key point of the proof.

*Claim 7.* If  $a_s \neq 0$ , then the probability that  $a_s$  is divisible by  $p$  is smaller than  $\frac{1}{4}$ .

*Proof of the claim.* Let  $A$  be the set of those primes in the range  $\{m, m+1, \dots, 2m-1\}$  that divide  $a_s$ . Since  $a_s \neq 0$ ,

$$|a_s| \geq \prod_{q \in A} q \geq m^{|A|}.$$

Since  $|a_s| < M$ , we have

$$m^{|A|} < M \leq 2^{\frac{\alpha}{2} \cdot m},$$

which implies that

$$|A| < \frac{\alpha}{2} \cdot \frac{m}{\log m}.$$

However, by Claim 6, the total number of primes in the range  $\{m, m+1, \dots, 2m-1\}$  is at least  $\alpha \cdot \frac{m}{\log m}$ . Hence, a prime chosen from this range uniformly at random belongs to  $A$  with probability smaller than  $\frac{1}{4}$ . ■

Note that  $p \geq m > B$ , hence all denominators of rationals given on input, including constants used in  $C$ , are in the range between 1 and  $p-1$ . Therefore, we can replace each such rational  $\frac{a}{b} \in \mathbb{Q}$  with the value  $ab^{-1} \in \mathbb{F}_p$ , where the inverse  $b^{-1}$  is taken in the field  $\mathbb{F}_p$ . Note that  $b^{-1}$  can be computed in time polynomial in  $p = 2^{\mathcal{O}(N)}$  using the extended Euclidean algorithm. We may now reinterpret the circuit  $C$  as

a circuit  $\hat{C}$  over the field  $\mathbb{F}_p$ . Applying Lemma 18 to  $\hat{C}$ , we can compute the values of all the gates of  $\hat{C}$  in time  $2^{\mathcal{O}(N)}$ ; this is because arithmetic operations in  $\mathbb{F}_p$  take time  $p^{\mathcal{O}(1)} = 2^{\mathcal{O}(N)}$ . Recalling that the gate  $g_i$  of  $C$  evaluates to the rational  $\frac{a_i}{b_i}$ , it now follows by a straightforward induction that in  $\hat{C}$ ,  $g_i$  evaluates to  $a_i b_i^{-1} \in \mathbb{F}_p$  (in particular,  $b_i$  is not divisible by  $p$ ). Letting  $r := a_s b_s^{-1} \in \mathbb{F}_p$  be the value computed for the output gate, we immediately see the following.

*Claim 8.* If  $a_s = 0$  then  $r = 0$ . Moreover, if  $a_s$  is not divisible by  $p$ , then  $r \neq 0$ .

Hence, the algorithm can simply report whether  $r = 0$  in  $\mathbb{F}_p$ . By Claims 7 and 8 there are no false negatives and the probability of a false positive is smaller than  $\frac{1}{2}$  (where  $\frac{1}{4}$  is incurred by the construction of  $p$  and Claim 8 gives another  $\frac{1}{4}$ ). □