Pumping-Like Results for Copyless Cost Register Automata and Polynomially Ambiguous Weighted Automata

Filip Mazowiecki
University of Warsaw, Poland

Antoni Puch
University of Warsaw, Poland

Daniel Smertnig
University of Ljubljana and Institute of Mathematics, Physics, and Mechanics (IMFM), Slovenia

Abstract

In this work we consider two rich subclasses of weighted automata over fields: polynomially ambiguous weighted automata and copyless cost register automata. Primarily we are interested in understanding their expressiveness power. Over the field of rationals and 1-letter alphabets, it is known that the two classes coincide; they are equivalent to linear recurrence sequences (LRS) whose exponential bases are roots of rationals. We develop a tool we call Pumping Sequence Families, which, by exploiting the simple single-letter behaviour of the models, yields two pumping-like results over arbitrary fields with unrestricted alphabets, one for each class. As a corollary of these results, we present examples proving that the two classes become incomparable over the field of rationals with unrestricted alphabets.

We complement the results by analysing the zeroness and equivalence problems. For weighted automata (even unrestricted) these problems are well understood: there are polynomial time, and even NC^2 algorithms. For copyless cost register automata we show that the two problems are PSpace-complete, where the difficulty is to show the lower bound.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Quantitative automata

Keywords and phrases weighted automata, cost register automata, ambiguity, linear recurrence sequences, equivalence problem

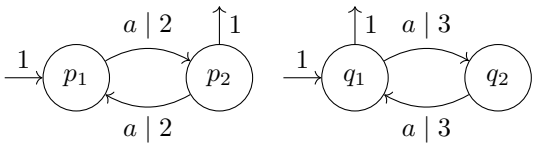
Digital Object Identifier 10.4230/LIPIcs.STACS.2026.67

Related Version *Full Version*: `https://arxiv.org/abs/2502.07356` [19]

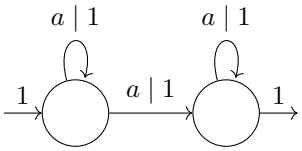
Funding *Filip Mazowiecki*: Supported by Polish National Science Centre SONATA BIS-12 grant number 2022/46/E/ST6/00230.
Antoni Puch: Supported by Polish National Science Centre SONATA BIS-12 grant number 2022/46/E/ST6/00230.
Daniel Smertnig: Supported by the Slovenian Research and Innovation Agency (ARIS) program P1-0288 and grant J1-60025.

1 Introduction

Weighted automata are a computational model assigning values from a fixed domain to words [12]. The domain can be anything with a semiring structure. Typical examples are: fields [28], where in particular probabilistic automata assign to every word the probability of its acceptance [24]; and tropical semirings, popular due to their connection with star height problems [16]. In this paper we focus on weighted automata over fields. These are finite



(a) A four state unambiguous weighted automaton $\mathcal{A}$ over a 1-letter alphabet $\{a\}$. Nonzero initial labels for p_1 and q_1 have value 1. The nonzero final states are p_2 and q_1, both with weight 1. For every word a^n there are two runs: on the left of value $1 \cdot 2^n$; on the right of value $1 \cdot 3^n$. Depending on the parity of n only one of the runs has nonzero output, thus $\mathcal{A}(a^n) = 3^n$ for even n, and $\mathcal{A}(a^n) = 2^n$, otherwise.



(b) A 2-state polynomially ambiguous weighted automaton $\mathcal{B}$ over the alphabet $\{a\}$. Note that every word a^n has n runs with value 1, hence $\mathcal{B}(a^n) = n$.

Figure 1 Weighted automata over the field of rationals $\mathbb{Q}(+, \cdot)$. For clarity, we omit zero labels.

automata with transitions, input and output edges additionally labeled by weights from the field. On an input word the value of a single run is the product of all weights, and the output of the weighted automaton is the sum of values over all runs. See Figure 1 for examples.

Unlike finite automata, nondeterminism makes weighted automata more expressive. This naturally leads to the decision problem of *determinisation*: given a weighted automaton does there exist an equivalent deterministic one? Over fields, it was recently shown that the problem is decidable [4], later improved to a 2-ExpTime upper bound on the running time [5]. Both papers rely on techniques used to obtain Bell and Smertnig's result [3] characterising the intermediate class of *unambiguous weighted automata*: a subclass that allows nondeterminism, but for every word there is at most one run of nonzero value. The authors proved Reutenauer's conjecture [26], which we explain below.

Given a weighted automaton $\mathcal{W}$ over the alphabet Σ consider $\mathcal{W}(\Sigma^*)$, the set of all outputs over all words. For example, in Figure 1 we have: $\mathcal{A}(a^*) = \{\, 2^{2n+1} \mid n \in \mathbb{N} \,\} \cup \{\, 3^{2n} \mid n \in \mathbb{N} \,\}$; and $\mathcal{B}(a^*) = \mathbb{N}$. Reutenauer's conjecture (now Bell and Smertnig's Theorem) states that for every weighted automaton $\mathcal{W}$ over a field K: there exists an equivalent unambiguous weighted automaton if and only if there exists a finitely generated multiplicative subgroup $G \subseteq K$ such that $\mathcal{W}(\Sigma^*) \subseteq G \cup \{0\}$. For example $\mathcal{A}(a^*) \subseteq G_{\mathcal{A}}$, where $G_{\mathcal{A}}$ is generated by the transition weights $\{2, 3\}$. It is not hard to see that this construction generalises to every unambiguous weighted automaton, the crux is to prove the other implication. As an immediate nontrivial application, notice that there is no unambiguous weighted automaton equivalent to $\mathcal{B}$, as the set $\mathbb{N} \setminus \{0\}$ is not contained in any finitely generated subgroup.

The equivalence problem for weighted automata over fields is famously decidable in polynomial time [28]. However, most natural problems are undecidable [24, 14, 11, 10]. This triggered the study of intermediate classes between deterministic and unrestricted weighted automata. One way to define such a class is based on *ambiguity*, generalising unambiguous weighted automata. A much broader class are *polynomially ambiguous weighted automata*, where the number of accepting runs is bounded by a polynomial in the size of the input word (see Figure 1b). Restricting the input automaton to polynomially ambiguous can significantly lower the complexity of a problem, for example, the discussed problem of determinisation is known to be in PSpace over the field of rationals [17]. Another way to define such a class comes from *cost register automata* [1] (CRA), a deterministic model with polynomial register updates. In this context it is natural to consider its copyless restriction (CCRA), because every function recognisable by a CCRA is also recognisable by a weighted automaton [20] (for simplicity the definition of CCRA is postponed to Section 2).

As far as we know, these two classes, polynomially ambiguous weighted automata and copyless CRA, are the richest studied classes that are known to be strictly contained in the class of unrestricted weighted automata. Over the tropical semiring they are known to be incomparable in terms of expressiveness [21, 9], which suggests the same over fields. One attempt to prove this result was in [2], where the authors considered weighted automata over the field of rationals with 1-letter alphabets. By identifying words a^n with their length n, one can view such automata as sequences. In fact weighted automata over 1-letter alphabets are equivalent to the well-known class of *linear recurrence sequences* (LRS) [22]. In [2], the authors prove that polynomially ambiguous weighted automata and copyless CRA coincide, and that they are also equivalent to the class of LRS whose exponential bases are roots of rationals. This means that in the exponential polynomial representation of LRS: $\sum_{i=1}^{n} p_i(x)\lambda_i^x$, for every i there is an n_i such that $\lambda_i^{n_i} \in \mathbb{Q}$. In particular this shows that the Fibonacci sequence does not belong to this class, as the golden ratio φ is not of this form.

Our Contribution

Our work can be seen as a follow-up to [2]. An immediate corollary of our results is that polynomially ambiguous weighted automata and copyless CRA over the field of rationals are incomparable classes in terms of expressiveness. To prove this, we developed a tool we call Pumping Sequence Families (PSF), which allows us to exploit the behaviour of these classes over 1-letter alphabets. In the following we use the standard sequence notation $(a_n)_n = a_0, a_1, a_2, \ldots$.

▶ **Definition 1.** *A Pumping Sequence Family of a function $f : \Sigma^* \mapsto A$, for any set A, is the set of all sequences of the form $\hat{f}(u, w, v) \coloneqq (f(uw^n v))_n$, with u, w, v ranging over all words in Σ^*. We denote it by* $\mathrm{PSF}(f)$.

One should think that f is being projected onto many single-letter-like cases at once, where w plays the role of the single letter in the alphabet, while u and v correspond to slight adjustments of respectively the initial and acceptance conditions. The definition of $\mathrm{PSF}(f)$ exploits that u, v and w range over all words, which captures behaviour beyond the single letter alphabet case. Using fixed words one cannot differentiate polynomially ambiguous weighted automata and copyless CRA due to [2, Theorem 6, Theorem 13]. However, this simple extension of the single letter case analysis will be enough to show those models to be incomparable. To showcase our approach, let us consider a simple application for languages, where $A = \{\top, \bot\}$ (meaning acceptance and rejection of the word).

▶ **Example 2.** Consider $f : \{a\}^* \mapsto \{\top, \bot\}$ which maps all words of even length to $\top$ and all others to $\bot$. Then $\mathrm{PSF}(f)$ consists of four sequences (for simplicity we write an example generator for each sequence):

- $\hat{f}(\varepsilon, \varepsilon, \varepsilon) = \top, \top, \top, \top, \ldots$
- $\hat{f}(a, \varepsilon, \varepsilon) = \bot, \bot, \bot, \bot, \ldots$
- $\hat{f}(\varepsilon, a, \varepsilon) = \top, \bot, \top, \bot, \ldots$
- $\hat{f}(a, a, \varepsilon) = \bot, \top, \bot, \top, \ldots$.

▶ **Example 3.** Consider $g : \{a, b\}^* \mapsto \{\top, \bot\}$ which maps all words of the form $a^n b^n$ to $\top$ and all others to $\bot$. Then $\mathrm{PSF}(g)$ is an infinite set:

- $\hat{g}(ab, \varepsilon, \varepsilon) = \top, \top, \top, \top, \ldots$
- $\hat{g}(a, \varepsilon, \varepsilon) = \bot, \bot, \bot, \bot, \ldots$
- $\hat{g}(\varepsilon, ab, \varepsilon) = \top, \top, \bot, \bot, \ldots$
- $\hat{g}(a^k, b, \varepsilon) = \underbrace{\bot, \ldots \bot}_{k}, \top, \bot, \bot, \ldots$ for every $k \geq 0$.

The above examples already present us with a simple use case for Pumping Sequence Families. By looking at the transition function of the underlying DFA we can see that, generalising Example 2, for a regular language the Pumping Sequence Family of its characteristic function will be finite. However, as witnessed in Example 3, the Pumping Sequence Family of the context-free language $L = \{ a^k b^k \mid k \in \mathbb{N} \}$ is infinite, proving that it is not regular and thus differentiating regular and context-free languages. Note that, due to Parikh's theorem, over 1-letter alphabets the two language classes are equivalent and semi-linear. Meaning if we would fix u, v and w then simply looking at the single letter case behaviour we would not be able to differentiate these classes.

We now present how we use Pumping Sequence Families for weighted functions. There the set A from Definition 1 is simply the underlying field.

For a function h recognised by a Copyless Cost Register Automaton we will restrict elements of its Pumping Sequence Family. Note that, since copyless CRA are a subset of weighted automata, elements of $\mathrm{PSF}(h)$ can be essentially represented as exponential polynomials. Consider such a sequence $a_n = \sum_{i=1}^{d} p_i(n)\lambda_i^n$. Let us write the polynomials p_i explicitly: $p_i(x) = \sum_{j=1}^{m_i} \alpha_{i,j} x^j$. For every degree k we define the sum of k-degree coefficients $S_k((a_n)_n) = \sum_{i=1}^{d} \alpha_{i,k}$. In Theorem 15 we show that, up to minor technical details, for all k the set $\{ S_k((a_n)_n) \mid (a_n)_n \in \mathrm{PSF}(h) \}$ is contained in a finitely generated subsemiring R. For intuition, if we consider the generators $\{\frac{1}{2}, \frac{1}{3}\}$, by adding, subtracting and multiplying, they generate $R = \{ \frac{a}{6^k} \mid a \in \mathbb{Z}, k \in \mathbb{N} \}$. This allows us to give an example of a polynomially ambiguous automaton that is not definable by any copyless CRA (the proof is short but technical, see Example 17).

For polynomially ambiguous weighted automata, our work is inspired by [25], where the authors attempt to characterise polynomially ambiguous weighted automata in a similar manner to Bell and Smertnig's Theorem. For a function recognised by a polynomially ambiguous weighted automaton $\mathcal{W}$ and given a sequence $(c_n)_n \in \mathrm{PSF}(\mathcal{W})$ consider again its exponential polynomial $\sum_{i=1}^{d} p_i(x)\lambda_i^x$ and let $E((c_n)_n) = \{ \lambda_i \mid 1 \leq i \leq d \}$ be the set of exponential bases. In Theorem 28 we show that the set $\bigcup_{(c_n)_n \in \mathrm{PSF}(\mathcal{W})} E((c_n)_n)$ is contained in a finitely generated subgroup G. We provide a self-contained proof, and we show that our property, which is simpler to work with when considering examples, is equivalent to the one in [25] (conjectured to characterise polynomially ambiguous automata). We obtain a corresponding, simple but technical, example of a copyless CRA that is not definable by any polynomially ambiguous weighted automaton (Example 29).

In this context a natural question is whether our property for copyless CRA can be a characterisation. We conjecture that it is not the case and that, in some sense, such a characterisation should not exist. We show examples of functions that satisfy the property we developed for CCRA, but we find it unlikely that there are CCRA that define them. More generally, in [21] the authors prove that the class of CCRA is not closed under reversal for the tropical semiring. More precisely, there is a CCRA $\mathcal{C}$ such that there is no CCRA $\mathcal{C}'(w) = \mathcal{C}(w^r)$, where w^r is w reversed. We conjecture that over fields CCRA are also not closed under reversal, which makes such characterisations unlikely.

Our final contribution is the analysis of the equivalence and zeroness problems for both classes. As already mentioned for weighted automata (even without restrictions) equivalence and zeroness are in polynomial time [28] and even in NC^2 [29]. For copyless CRA the translation to weighted automata [20] yields an exponential blow up in the size of the automaton (we provide a self-contained short translation). Since problems in NC^2 can be solved sequentially in polylogarithmic space [27], this yields a trivial PSpace algorithm. Our contribution is a matching PSpace lower-bound.

▶ **Theorem 4.** *Zeroness and equivalence problems are* PSPACE*-complete for CCRA over* $\mathbb{Q}$.

Organisation

We start with definitions in Section 2. In Section 3 and Section 4 we prove the properties of copyless CRA, and polynomially ambiguous weighted automata, respectively; and we present examples separating the classes. In Section 5 we discuss the decision problems.

2 Preliminaries

Let $\mathbb{N} \coloneqq \{0, 1, 2, \dots\}$. For a field K, let $K^\times \coloneqq K \setminus \{0\}$ denote the multiplicative group of nonzero elements. We sometimes write 1_K and 0_K for the elements 1 and 0 of the field, to emphasize which 1 and 0 we mean.

2.1 Automata and Sequences

A *weighted automaton* over a field K is a tuple $\mathcal{A} = (d, \Sigma, (M(a))_{a\in\Sigma}, I, F)$, where: $d \in \mathbb{N}$ is its dimension; Σ is a finite alphabet; $M(a) \in K^{d\times d}$ are transition matrices; I, $F \in K^d$ are the initial and final vectors, respectively. For simplicity, sometimes we will write $\mathcal{A} = (d, M, I, F)$, that is, we will omit Σ in the tuple.

Weighted automata can be defined more generally over semirings, but in this paper we only consider the case of fields. The field $\mathbb{Q}$ is already rich enough to produce all phenomena of interest to us. Thus, our examples will be for $\mathbb{Q}$ with the usual addition and product, unless stated otherwise.

Given a word $w = w_1 \dots w_n \in \Sigma^*$, we denote $M(w) \coloneqq M(w_1) \cdot \ldots \cdot M(w_n)$. In particular $M(\epsilon)$ is the $d \times d$-identity matrix. A weighted automaton defines a function $\mathcal{A}\colon \Sigma^* \to K$, by $\mathcal{A}(w) \coloneqq I^\intercal \cdot M(w) \cdot F$. We say that a weighted automaton $\mathcal{A}$ is a *linear recurrence sequence* (*LRS*) if $|\Sigma| = 1$. Then, by identifying Σ^* with $\mathbb{N}$, that is, identifying the word a^n with its length n, we write that $\mathcal{A}\colon \mathbb{N} \to K$. We will also denote such sequences $(a_n)_n$ instead of $\mathcal{A}$, where $a_n \coloneqq \mathcal{A}(n)$.

▶ **Example 5.** Consider an LRS $\mathcal{A} = (2, M, I, F)$, where: $M = \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$; $I = \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right)$ and $F = \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right)$. Then $\mathcal{A}(n) = a_n = n$.

For weighted automata we define their underlying automata. A weighted automaton $\mathcal{A} = (d, \Sigma, (M(a))_{a\in\Sigma}, I, F)$ can be interpreted as an automaton with states $\{1, \dots, d\}$ such that for every $a \in \Sigma$ a nonzero entry in $M_a[i, j]$ defines a transition from i to j labeled by a of weight $M_a[i, j]$ (thus we ignore transitions of weight 0). Similarly, initial and final states are i such that $I[i]$ and $F[i]$ are nonzero, respectively. Their weights are $I[i]$ and $F[i]$. By ignoring the weights of transitions, initial and final states, we obtain a finite automaton $\mathcal{B}$, which we call the *underlying automaton* of $\mathcal{A}$.

▶ **Example 6.** The LRS $\mathcal{A} = (2, M, I, F)$ in Example 5 is an equivalent presentation of the weighted automaton $\mathcal{B}$ in Figure 1b.

A weighted automaton $\mathcal{A}$ is *polynomially ambiguous* if there is a polynomial function $p\colon \mathbb{N} \to \mathbb{N}$ such that for every $w \in \Sigma^*$ the number of accepting runs of the underlying automaton on w is bounded by $p(|w|)$. For example, the automaton in Example 6 is polynomially ambiguous as it suffices to take $p(n) = n$. In general this is a strict subclass: there exist weighted automata that are not equivalent to any polynomially ambiguous weighted automata.

LRS can be characterised in another way. An LRS $(a_n)_n$ can be defined by a (homogeneous) recurrence relation of the form $a_{n+k} = \sum_{i=0}^{k-1} c_i \cdot a_{n+i}$ with $c_i \in K$ and k initial values a_0, $\ldots$, a_{k-1}. Here k is the *order* of the recurrence. For instance, the LRS $(a_n)_n$ from Example 5 can be defined by $a_{n+2} = 2a_{n+1} - a_n$ and $a_0 = 0$, $a_1 = 1$. It is well-known that the two definitions coincide [15, Lemma 1.1] [8, Proposition 2.1]. Moreover, the translation is effective in polynomial time, and under this translation, the dimension d of the weighted automaton equals the order k of the recurrence.

Any given LRS $(a_n)_n$ satisfies many different linear recurrences. However, it is well-known that there is a unique (homogeneous) recurrence of minimal order satisfied by $(a_n)_n$ [6, Ch. 6.1]. The corresponding order k is then the *order* of the LRS. This minimal recurrence gives rise to the *characteristic polynomial* $q = x^k - c_{k-1}x^{k-1} - \cdots - c_0$ of $(a_n)_n$ [6, Ch. 6.1] [13]. The roots of the characteristic polynomial (considered in the algebraic closure $\overline{K}$) are the *characteristic roots* of the LRS $(a_n)_n$.

▶ **Example 7.** Continuing from Example 5, the characteristic polynomial is $q = x^2 - 2x + 1 = (x-1)^2$. Hence, the only characteristic root is 1 (with multiplicity 2).

The characteristic roots of LRS definable by polynomially ambiguous weighted automata are always roots of elements of K [2, 18, 25]. So, for example, the Fibonacci sequence $F_0 = 0$, $F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$, is not recognised by a polynomially ambiguous weighted automaton over $\mathbb{Q}$, as its characteristic roots are the golden ratio $\varphi = \frac{1+\sqrt{5}}{2}$ and $\psi = \frac{1-\sqrt{5}}{2}$.

We recall an additional characterisation of LRS, namely as coefficient sequences of rational functions, leading to exponential polynomials. See also [6, Chapter 6][13][15, Proposition 2.11] or [19, Appendix A]. A sequence $(a_n)_n$ is an LRS if and only if the (formal) generating series $F = \sum_{n=0}^{\infty} a_n x^n \in K[\![x]\!]$ is a rational function: the series F is the formal Taylor series expansion of some p/q with p, $q \in K[x]$ coprime polynomials and $q(0) \neq 0$. For nonzero $\lambda \in \overline{K}$, the following are now equivalent:

- λ is a characteristic root of $(a_n)_n$;
- λ appears as an eigenvalue of $M(a)$ in a weighted automaton representation of $(a_n)_n$ of minimal dimension;
- $1/\lambda$ is a pole of F, that is, $q(1/\lambda) = 0$.

Further, the characteristic roots appear as eigenvalues of $M(a)$ in *every* representation of $(a_n)_n$ using a weighted automaton. But in a weighted automaton that is not of minimal dimension, the matrix $M(a)$ may have additional eigenvalues.

In characteristic 0, every LRS $(a_n)_n$, has, for large enough n, a representation as an *exponential polynomial sequence* (*EPS*):

$$a_n = \sum_{i=1}^{k} q_i(n)\lambda_i^n \quad \text{for sufficiently large n,}$$

with q_i polynomials over $\overline{K}$ and $\lambda_i \in \overline{K}$ the nonzero characteristic roots of $(a_n)_n$. Furthermore, the exponential bases λ_i and the polynomials q_i are uniquely determined by $(a_n)_n$.

▶ **Example 8.** The Fibonacci numbers admit the representation $F_n = \frac{1}{\sqrt{5}}\varphi^n - \frac{1}{\sqrt{5}}\psi^n$.

In characteristic $p > 0$, the situation is more complicated (see [19, Appendix A]): an LRS may not have a representation by an exponential polynomial (even for large n). If it does have such a representation, it is however still unique as long as the polynomials q_i are chosen of minimal degree, that is, with $\deg(q_i) < p$. Further, every EPS is an LRS, and the exponential bases of the EPS are precisely the nonzero characteristic roots of the LRS.

2.2 Cost Register Automata

We will introduce one more formalism that generalises weighted automata to polynomial updates [1]. A *cost register automaton* (CRA) over a field K is a tuple $\mathcal{C} = (Q, q_0, d, \Sigma, \delta, \mu, \nu)$, where: Q is a finite set of states; $q_0 \in Q$ is the initial state; $d \in \mathbb{N}$ is its dimension; Σ is a finite alphabet; $\delta \colon Q \times \Sigma \to Q \times \mathrm{Poly}^d$ is a deterministic transition function, where Poly^d is the set of d-dimensional polynomial maps; $\mu \colon K^d$ is the vector of initial register values; and $\nu \colon Q \to \mathrm{Poly}^d$ is the final function. Here, a *polynomial map* $P \in \mathrm{Poly}^d$ is a tuple $P = (p^1, \ldots, p^d)$ with polynomials $p^i \in K[x_1, \ldots, x_d]$. Every polynomial map induces a function $K^d \to K^d$.

Given $q \in Q$ and $a \in \Sigma$ we write $p_{q,a}$ for the polynomial map such that $\delta(q, a) = (q', p_{q,a})$ for some $q' \in Q$. Note that if we ignore the polynomials in δ, then (Q, q_0, Σ, δ) is just a deterministic finite automaton without final states. Thus, given a word w, there is a unique state reachable from q_0 when reading w. We will denote it q_w. For words $w \in \Sigma^+$ we define polynomial maps p_w by induction: if $w = a \in \Sigma$ is a letter then $p_w = p_{q_0,a}$; otherwise if $w = w'a$ for a letter $a \in \Sigma$ then $p_w = p_{q_{w'},a} \circ p_{w'}$.

A CRA defines a function $\mathcal{C} \colon \Sigma^* \to K$, similarly to weighted automata. Formally, given a word $w = w_1 \ldots w_n \in \Sigma^*$ we define $\mathcal{C}(w) = (\nu(q_w) \circ p_w)(\mu)$.

▶ **Example 9.** We can define the automaton recognising the same function as in Example 5. There is only one state, which is also initial, and only one letter. The dimension is 2, we will label the two resulting registers as x and y. There is only one transition defined by the polynomial map (p^x, p^y) with $p^x(x, y) = x + y$ and $p^y(x, y) = y$. The initial vector is defined by $\mu(x) = 0$, $\mu(y) = 1$; and the output is the polynomial x.

One can think of the polynomial maps as generalising linear updates definable by matrices. When restricting the model to linear polynomials, the CRA formalism is equivalent to weighted automata [1], and it is called linear CRA. The resulting weighted automaton is of polynomial size in the size of the linear CRA. Note that CRA use separate notions of states (the set Q) and registers (i.e., the dimension d). In general, states are not needed, as one can easily encode the states by enlarging the dimension to $d \times |Q|$, even for linear CRA. However, such encodings do not preserve the copyless restriction on CRA, which we discuss next.

A *copyless* CRA (CCRA) is a CRA such that all polynomial maps in the transition function and the output function are copyless. A polynomial map $P \in \mathrm{Poly}^d$ is copyless if it can be written using sum, product, variable names and constants using each variable name only once. In particular x^k for $k > 1$ is not copyless.

▶ **Example 10.** For $d = 3$ the map P is defined by three polynomials $p^x(x, y, z)$, $p^y(x, y, z)$ and $p^z(x, y, z)$. If $p^x = (x + 3) \cdot (y + z)$, $p^y = 7$ and $p^z = 1$, then P is copyless; but if $p^x = y + 1$, $p^y = y$ and $p^z = z$, then P is not copyless.

It is easy to see that copyless polynomial maps are preserved under composition. Thus, in a CCRA all polynomial maps p_w are copyless.

Functions definable by Copyless CRA are known to be definable by weighted automata [20, 21].

3 Pumping Sequence Families of CCRA

Throughout the section, fix a field K. In this section we prove a result restricting Pumping Sequence Families of CCRA. This result will be based on the observation that sequences of the form $(\mathcal{A}(uw^{m(n+1)}v))_n$, obtained from a CCRA $\mathcal{A}$, are always representable by very particular exponential polynomials. To this end, we first introduce the following class of functions.

▶ **Definition 11.** *A K-valued sequence $(a_n)_n$ is an* exponential polynomial sequence generable from $A \subseteq K$ (*in short, an* A-generable EPS) *if it can be obtained, using pointwise products and sums, from the following sequence families:*

- *Constant sequences $(\alpha)_n$ for $\alpha \in A$,*
- *The linear sequence $(n \cdot 1_K)_n$,*
- *Exponential sequences (that is, geometric progressions) $(\alpha^n)_n$ for $\alpha \in A$,*
- *Sequences of the form $\left(\frac{1}{\alpha-1}\alpha^n - \frac{1}{\alpha-1}\right)_n$ for $1 \neq \alpha \in A$.*

The last family may be a bit unexpected at first glance. It arises from the geometric sum

$$\frac{1}{\alpha-1}\alpha^n - \frac{1}{\alpha-1} = \frac{\alpha^n - 1}{\alpha - 1} = \sum_{i=0}^{n-1} \alpha^i \qquad (\alpha \neq 1),$$

with the representation in Definition 11 corresponding to the normal form for exponential polynomials (with the two exponential bases α and 1_K). Because possibly $1/(1-\alpha) \notin A$, this last family cannot always be generated from the other three families.

▶ **Example 12.** Since $\sum_{i=0}^{n-1} \alpha^i = \alpha\big(\cdots(\alpha(\alpha+1)+1)\big)+1$, the geometric sum appears when iterating a copyless update rule of the form $x \mapsto \alpha x + 1$ from the starting value 1.

Taking $A = K$, the class of K-generable EPS admits a more familiar description.

▶ **Lemma 13.** *A sequence $(a_n)_n$ is a K-generable EPS if and only if it is an EPS with coefficients and exponential bases in K.*

Proof. We first check that every K-generable EPS is indeed an EPS. Since EPS with coefficients and exponential bases in K are closed under products and sums, it suffices to verify the claimed property for the families in Definition 11. However, each of these families is obviously an EPS and the only exponential bases that appear are 1_K and $\alpha \in K$.

Conversely, suppose that $(a_n)_n$ has a representation $a_n = \sum_{\lambda \in K^\times} \sum_{i \geq 0} \alpha_{\lambda,i} n^i \lambda^n$ with $\alpha_{\lambda,i} \in K$ (only finitely many of which are nonzero). Each of $(\alpha_{\lambda,i})_n$, $(n^i)_n$ and $(\lambda^n)_n$ is clearly a K-generable EPS, and so is therefore $(a_n)_n$. ◀

Recall that the exponential bases being contained in K is a nontrivial restriction on an EPS. In general, these will be contained in the algebraic closure $\overline{K}$. In particular, every A-generable EPS is trivially a K-generable EPS, and hence by Lemma 13 an EPS (in the sense discussed in Section 2), so that our terminology is consistent. Working with, possibly proper, subsets $A \subseteq K$ will be crucial to obtain a pumping-like criterion that is strong enough to differentiate between CCRA and polynomially ambiguous WFA.

We need a final definition before stating our main theorem of the section.

▶ **Definition 14.** *Let $R \subseteq K$ be a subsemiring. An* R-CCRA *is a CCRA with all of its initial register values, output expression and transition coefficients in R.*

We will now exploit Pumping Sequence Families, the main tool introduced in this paper (recall Definition 1).

▶ **Theorem 15.** *If $R \subseteq K$ is a subsemiring and $f\colon \Sigma^* \to K$ is recognised by an R-CCRA, then there exists $m \geq 1$ such that,*

- *for every $g \in \mathrm{PSF}(f)$, the sequence $(h(n))_n = (g(m(n+1)))_n$ is an R-generable EPS, and*
- *if the characteristic of K is 0 and q is the exponential polynomial representing h, then for every $k \in \mathbb{N}$ the sum of k-degree coefficients $S_k(q)$ is in R.*

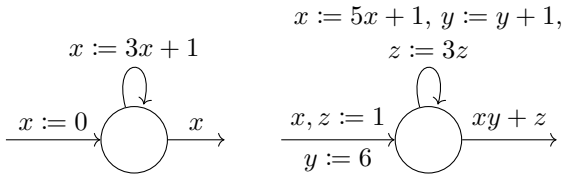


Figure 2 Two simple single-state CCRAs on a single-letter alphabet (Example 16).

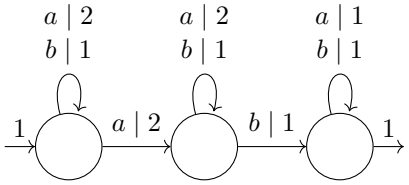


Figure 3 A polynomially ambiguous weighted automaton with no equivalent CCRA (Example 17).

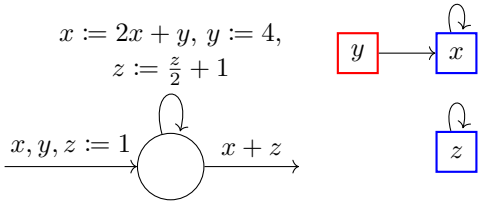


Figure 4 A simple CCRA and its variable flow graph. Red nodes are constant registers; blue nodes are updating ones (Definition 19).

The sum of k-degree coefficients $S_k(q)$ is obtained by summing all the coefficients of x^k in q across all the exponential bases (see [19, Appendix A] for a detailed discussion). The characteristic condition in the second property can be removed, leading to a slightly weaker result which is discussed in [19, Appendix B.7].

It is obvious that, for any input, the output of an R-CCRA is in R. However, this is different from the property in Theorem 15 – we make a claim about the coefficients of the exponential polynomial, not the values that it takes. The individual coefficients do not need to always lie in R, as the following example illustrates.

Observe that we can assume R is finitely generated – by the initial register values, output expression and transition coefficients of the CCRA.

▶ **Example 16.**

- Consider the left $\mathbb{Z}$-CCRA in Figure 2. On words of the form a^{n+1}, this CCRA outputs $q(n) = \mathcal{A}(n+1) = \frac{3^{n+1}-1}{2} = \frac{3}{2} \cdot 3^n - \frac{1}{2} \cdot 1^n$. Even though the automaton itself only uses integer coefficients, a denominator 2 appears in the coefficients of q. However, the sum of the coefficients is $S_0(q) = \frac{3}{2} - \frac{1}{2} = 1$, an integer.
- The second $\mathbb{Z}$-CCRA in Figure 2 outputs

$$\mathcal{B}(a^{n+1}) = \frac{5^{n+2}-1}{4}(n+7) + 3^{n+1} = \left(\tfrac{25}{4}n + \tfrac{175}{4}\right) \cdot 5^n + 3 \cdot 3^n + \left(-\tfrac{1}{4}n - \tfrac{7}{4}\right) \cdot 1^n =: q(n).$$

Here $S_1(q) = \frac{25}{4} - \frac{1}{4} = 6 \in \mathbb{Z}$ and $S_0(q) = \frac{175}{4} + 3 - \frac{7}{4} = 45 \in \mathbb{Z}$.

Before proving Theorem 15, we demonstrate how it can be applied. We use it to show that not every function recognisable by a polynomially ambiguous WFA can be recognised by a CCRA.

▶ **Example 17.** The automaton in Figure 3 is polynomially ambiguous. Let $f\colon \{a,b\}^* \to \mathbb{Q}$ be the function associated with the automaton and, for the sake of contradiction, assume that f can be recognised by a CCRA. This yields a finitely generated subsemiring R and a natural number m such that for all $g \in \mathrm{PSF}(f)$, the sequence $(h(n))_n := (g((n+1)m))_n$

meets the conditions from Theorem 15. Consider, for any k, $g_k \coloneqq \hat{f}(\varepsilon, a^k b, \varepsilon) \in \mathrm{PSF}(f)$. By grouping the paths based on which b is used to transition between the second and third state, we get

$$h_k(n) \coloneqq g_k((n+1)m) = k2^k + 2k2^{2k} + \cdots + m(n+1)k2^{m(n+1)k} = \sum_{j=1}^{m(n+1)} jk2^{jk}.$$

Using the identity $\sum_{j=1}^{l} jx^j = \frac{lx^{l+2}-(l+1)x^{l+1}+x}{(x-1)^2}$, which can be derived from the geometric sum $\sum_{j=1}^{l} x^j = \frac{x^{l+1}-x}{x-1}$ by formal differentiation and some easy manipulations, one finds

$$h_k(n) = q_1(n) \cdot (2^{km})^n + q_2(n) \cdot 1^n,$$

with

$$q_1(n) = \frac{km2^{km+k}}{(2^k-1)}n + \frac{k(m2^k - m - 1)2^{km+k}}{(2^k-1)^2} \quad \text{and} \quad q_2(n) = \frac{k2^k}{(2^k-1)^2}.$$

We have $S_1(h_k) = km2^{km+k}/(2^k-1)$. Only a finite set of prime numbers can appear among denominators of elements of R and only finitely many primes divide $2m$, we can thus take a prime p that fulfills neither of these conditions. We can also now fix $k = p-1$. We get

$$S_1(h_{p-1}) = \frac{(p-1)m2^{(p-1)m+p-1}}{2^{p-1}-1}.$$

Since $p \nmid 2m$, the numerator is not divisible by p. However, by Fermat's Little Theorem, the denominator is. As we have assumed that p does not appear in the denominator of any element of R, this means $S_1(h_{p-1}) \notin R$, contradicting the statement of Theorem 15.

We record the conclusion as a theorem.

▶ **Theorem 18.** *If $|\Sigma| \geq 2$, then there exist functions $f\colon \Sigma^* \to \mathbb{Q}$ that are recognisable by a polynomially ambiguous weighted automaton, but not by a $\mathbb{Q}$-CCRA.*

Proof. By Example 17. ◀

3.1 The Proof of Theorem 15

The proof of Theorem 15 proceeds in several steps. We start with a very simple case and then, in each step, use the previous result to show a slightly more general one.

▶ **Definition 19.** *A single letter, single state CCRA is* simple *if, in the transition, every register value is either set to a constant* (constant registers)*, or depends only on its old value and on the values of constant registers* (updating registers)*.*

Since there is only one state, there is also only one transition, so the definition makes sense. We can visualise constant and updating registers with a graph representing the dependency of register values on each other (Figure 4).

▶ **Lemma 20.** *If $\mathcal{A}$ is a single state, single letter simple R-CCRA, then $(\mathcal{A}(a^{n+1}))_n$ is an R-generable EPS.*

Proof. In a simple CCRA, the register values change in very simple ways. For constant registers, after the first transition, they remain set to the same values. For updating registers, the first update is special. However, after that, the input they get from the constant registers

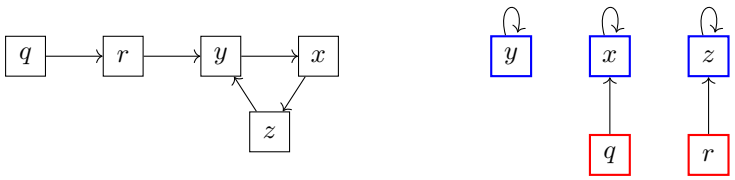


Figure 5 An example for variable flow graphs of $\mathcal{A}$ and $\mathcal{B}$ in the proof of Lemma 21.

stabilizes. Let us consider what happens from that point on. After the first step, the register values are of course still in R. Since the updating registers can only depend on themselves and constants, the update formulas can be reduced to the form $x \coloneqq \alpha x + \beta$ for constants $\alpha, \beta \in R$. This gives us an LRS ($x_{n+1} = \alpha x_n + \beta$). Solving the LRS, we need to distinguish two cases. For $\alpha = 1$, the solution is

$$x_{n+1} = x_1 + \beta n, \qquad \text{and for } \alpha \neq 1 \text{ it is} \qquad x_{n+1} = \alpha^n \frac{\beta}{\alpha - 1} - \frac{\beta}{\alpha - 1} + \alpha^n x_1.$$

In both cases the sequence $(x_{n+1})_n$ is clearly an R-generable EPS. Constant registers, leading to constant sequences, also clearly are R-generable EPS.

The output expression combines these sequences using sums and products of the sequences and additional constants from R. These operations preserve the property of being an R-generable EPS, and so the sequence $(\mathcal{A}(a^{n+1}))_n$ is an R-generable EPS. ◀

In the next two lemmas we will reason about the behaviour of CCRA on cycles. Similar, but different, observations were made in [21, Proposition 1 and Lemma 4].

▶ **Lemma 21.** *If $\mathcal{A}$ is a single state, single letter R-CCRA (not necessarily simple) with r registers, then $(\mathcal{A}(a^{r!(n+1)}))_n$ is an R-generable EPS.*

Proof. Consider the compound effect on registers of the letter a being applied $r!$ times. We can get the corresponding expressions simply by composing the substitution $r!$ times. They will still of course be copyless and polynomial, meaning we can create an auxiliary CCRA $\mathcal{B}$ with a 1-letter alphabet such that $\mathcal{B}(a^n) = \mathcal{A}(a^{nr!})$. It is also easy to see, from how substitutions compose, that $\mathcal{B}$ is still an R-CCRA.

We claim that the new CCRA $\mathcal{B}$ is simple: we will prove this by looking at the variable flow graph of $\mathcal{A}$. Since $\mathcal{A}$ is copyless, there is at most one outgoing edge from any vertex. In $\mathcal{B}$ the expression for register v will use u if and only if in the variable flow graph of $\mathcal{A}$ there is a path of length $r!$ from u to v. (This is visualised in Figure 5.)

Consider an arbitrary register t. It will either be in a cycle on the graph of $\mathcal{A}$ or not. Assume t is not in a cycle and there is a path of length $r!$ from some register u to t. Such a path would have to contain a cycle. However, that is impossible, since each vertex has at most one outgoing edge and t itself is not in a cycle. This means t will be a constant register in the auxiliary automaton.

Now assume t is in a cycle in the variable flow graph of $\mathcal{A}$, and let l be the length of the cycle. We want to prove that t is an updating register in $\mathcal{B}$. Assume there is a path of length $r!$ from some u to t. To show that t is an updating register, we need to show that either $u = t$ or u is a constant register in $\mathcal{B}$. If u is in the same cycle as t, we have $u = t$, since $l \mid r!$. If u is outside the cycle containing t, then u cannot be a part of any cycle, as any vertex can have at most one outgoing edge. This means that u is a constant register in $\mathcal{B}$. Thus, the auxiliary CCRA $\mathcal{B}$ is simple, and we can apply Lemma 20 to it, finishing the proof. ◀

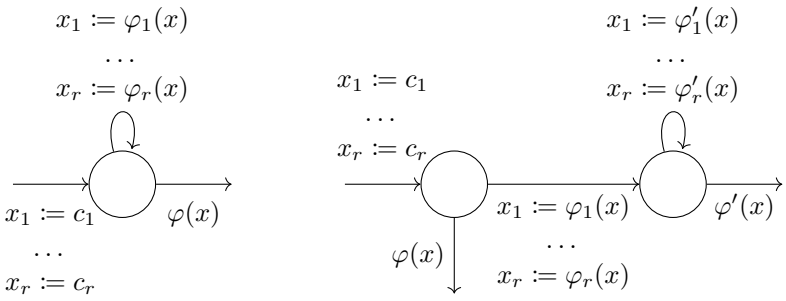


Figure 6 The two possible forms the auxiliary automaton $\mathcal{B}$ can take in the proof of Lemma 22.

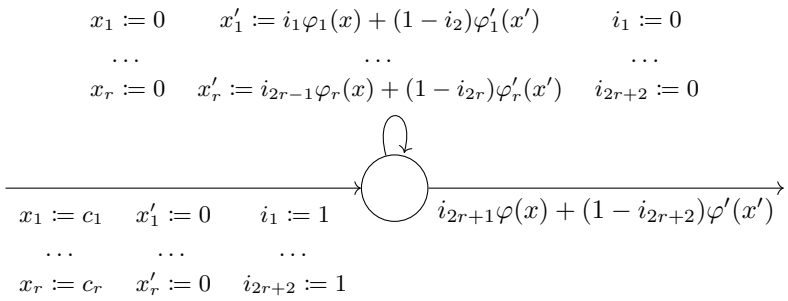


Figure 7 How to transform $\mathcal{B}$ into a single state automaton in the proof of Lemma 22.

▶ **Lemma 22.** *If $\mathcal{A}$ is a single letter R-CCRA (not necessarily single state) with s states and r registers, then $(\mathcal{A}(a^{(4r+2)!s!(n+1)}))_n$ is an R-generable EPS.*

Proof. Consider the compound effect on registers of the letter a being applied $s!$ times. We can get the corresponding transitions between states by looking at paths of length $s!$, and corresponding update expressions by composing appropriate $s!$ substitutions. The updates will of course still be copyless and polynomial, and the transitions deterministic, meaning we can create an auxiliary CCRA $\mathcal{B}$ such that $\mathcal{B}(a^n) = \mathcal{A}(a^{ns!})$. Note that the transition expression coefficients will all still be in R, so $\mathcal{B}$ is still an R-CCRA. Since $\mathcal{A}$ is deterministic, after at most s steps it always reaches a cycle. This cycle has length at most s, and so its length divides $s!$. This means that, after trimming $\mathcal{B}$, we get an automaton of one of the forms in Figure 6.

We want to reduce $\mathcal{B}$ to only one state. The first possible form already has only one state. The second one can easily be simulated with one state, as shown in Figure 7. After this operation, the automaton $\mathcal{B}$ is a single-state R-CCRA with $4r+2$ registers such that $\mathcal{B}(a^n) = \mathcal{A}(a^{ns!})$. This lets us use Lemma 21 and finishes the proof. ◀

▶ **Lemma 23.** *If $\mathcal{A}$ is an R-CCRA (not necessarily single letter) with r registers and s states, then, for all $w \in \Sigma^*$, the sequence $(\mathcal{A}(w^{(4r+2)!s!(n+1)}))_n$ is an R-generable EPS.*

Proof. Consider the composite effect of the word w on registers and state transitions. This effect is still copyless, polynomial, deterministic and all the transition coefficients are still in R. We can thus create an auxiliary R-CCRA $\mathcal{B}$ such that $\mathcal{A}(w^n) = \mathcal{B}(a^n)$. The CCRA $\mathcal{B}$ has a one letter alphabet, letting us use Lemma 22 and finishing the proof. ◀

▶ **Lemma 24.** *If $\mathcal{A}$ is an R-CCRA with r registers and s states, then, for all $u, w, v \in \Sigma^*$, the sequence $(\mathcal{A}(uw^{(4r+2)!s!(n+1)}v))_n$ is an R-generable EPS.*

Proof. Adding some prefix u simply changes the initial register values. The register values are of course still in R. Adding a suffix v simply changes the output expression. Its coefficients are of course still in R. Thus, we obtain an R-CCRA $\mathcal{B}$ with $\mathcal{B}(x) = \mathcal{A}(uxv)$ for all words $x \in \Sigma^*$. By Lemma 23, the sequence $(\mathcal{A}(uw^n v))_n = (\mathcal{B}(w^n))_n$ is an R-generable EPS. ◀

We also need the next lemma which is proven in [19, Appendix B].

▶ **Lemma 25.** *If $R \subseteq K$ is a subsemiring,* $\operatorname{char} K = 0$*, $(a_n)_n$ is an R-generable EPS and q is the exponential polynomial representing $(a_n)_n$, the sum of k-degree coefficients of q is in R.*

We can finally prove the main theorem of Section 3.

Proof of Theorem 15. Let $\mathcal{A}$ be an R-CCRA recognizing f. Let $m = (4r+2)!s!$, where r is the number of registers and s is the number of states of $\mathcal{A}$. By Lemma 24 the sequence $(h(n))_n \coloneqq (g((n+1)m))_n \coloneqq (f(uw^{m(n+1)}v))_n$ is an R-generable EPS. Let q be the exponential polynomial representing h. By Lemma 25, for every k, the sum of k-degree coefficients of this representation is in R. ◀

3.2 CCRA versus R-generable EPS

We have seen that, for functions $f\colon \Sigma^* \to K$ recognisable by a CCRA, there always exists a finitely generated subsemiring R and $m \geq 1$ such that for all $g \in \mathrm{PSF}(f)$ the sequence $(g((n+1)m))_n$ is an R-generable EPS. We conjecture that this is not sufficient to characterise functions recognised by CCRA, even if it is already known that the function is recognised by a weighted automaton.

At present, we do not have a counterexample, but we outline a plausible candidate in this subsection. However, it appears difficult to prove that the given function is not recognised by a CCRA.

▶ **Example 26.** Consider the following function $f\colon \{0,1\}^* \to \mathbb{Q}$. Given $w \in \Sigma^*$, let $0 < k_1 < k_2 < \ldots < k_r$ be the indices of all 1's in w, e.g. for $w = 0110$ we have: $r = 2$, $k_1 = 2$, $k_2 = 3$. Then $f(w) = \sum_{i=1}^{r} k_i$.

Technical computations show that each element of $\mathrm{PSF}(f)$ is a $\frac{1}{2}\mathbb{Z}$-generable EPS ([19, Appendix B]). Nevertheless, it appears unlikely to us that f could be recognised by a CCRA.

The reason why functions can or cannot be recognised by a CCRA can be subtle: while it appears that the function f in Example 26 cannot be recognised by a CCRA, the following example shows a function of similar nature that can be recognised by a CCRA.

▶ **Example 27.** We define $g\colon \{0,1\}^* \to \mathbb{Q}$: as before, given $w \in \Sigma^*$, let $0 < k_1 < k_2 < \ldots < k_r$ be the indices of all 1's. Then $g(w) = \sum_{i=1}^{r} 2^{k_i}$ is recognised by the CCRA in Figure 8.

Another promising example is discussed in [19, Appendix B].

4 Pumping Sequence Families of Polynomially Ambiguous WA

In this section we prove a result restricting Pumping Sequence Families of polynomially ambiguous weighted automata.

▶ **Theorem 28.** *If f is recognised by a polynomially ambiguous weighted automaton over K, then there exist a finitely generated multiplicative semigroup $G \subseteq \overline{K}$ and $N \geq 1$ such that*

- *the characteristic roots of every sequence in* $\mathrm{PSF}(f)$ *are contained in G,*
- *and $\alpha^N \in K$ for all $\alpha \in G$.*

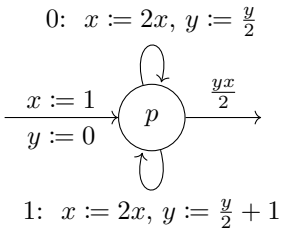


Figure 8 A CCRA recognising a function that, at first glance, may seem unrecognisable by a CCRA (Example 27).

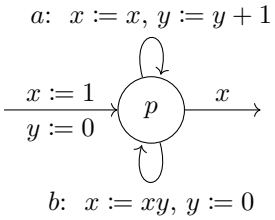


Figure 9 A two-register CRA recognising the function from Example 29.

With this theorem, we can give the following example.

▶ **Example 29.** The function $f\colon \{a,b\}^* \to \mathbb{Q}$ defined by the CCRA in Figure 9 cannot be recognised by a polynomially ambiguous weighted automaton.

Proof. Let us consider inputs of the form $(a^k b)^n$ for $k \geq 1$. We have $\hat{f}(\varepsilon, a^k b, \varepsilon)(n) = f\big((a^k b)^n\big) = k^n$, meaning that every natural number k appears as a characteristic root of an LRS in $\mathrm{PSF}(f)$. However, the monoid $(\mathbb{Z}_{>0}, \cdot)$ generates $(\mathbb{Q}_{>0}, \cdot)$ as a group, and $(\mathbb{Q}_{>0}, \cdot)$ is a countably generated free abelian group (with primes as the generators). Since subgroups of a finitely generated abelian group are finitely generated, but there are infinitely many primes, the natural numbers cannot be a submonoid of a finitely generated abelian group. Theorem 28 shows that f is not recognised by a polynomially ambiguous weighted automaton. ◀

We again record this conclusion as a theorem.

▶ **Theorem 30.** *If $|\Sigma| \geq 2$, then there exist functions $f\colon \Sigma^* \to \mathbb{Q}$ that can be recognised by a $\mathbb{Q}$-CCRA but not by a polynomially ambiguous weighted automaton over $\mathbb{Q}$.*

Proof. By Example 29. ◀

The core of the proof of Theorem 28 will be the following lemma. The argument is similar to an argument in [17] and in [25, Prop. 9.3]. A self-contained proof can be found in [19, Appendix B].

▶ **Lemma 31.** *Let $\mathcal{A} = (d, I, M, F)$ be a trim polynomially ambiguous weighted automaton. Then, for every word $w \in \Sigma^*$, there exists a permutation matrix P such that $P \cdot M(w^{d!}) \cdot P^{-1}$ is upper triangular. Furthermore, all nonzero eigenvalues of $M(w^{d!})$ are products of transition weights (that is, of entries of the matrices $M(a)$ for letters $a \in \Sigma$).*

Before proving Theorem 28, we need a final small observation.

▶ **Lemma 32.** *If $H \subseteq K$ is a finitely generated semigroup and $N \geq 1$, then the semigroup $G = \{\, \alpha \in \overline{K} \mid \alpha^N \in H \cup \{1\} \,\}$ is also finitely generated.*

Proof. Suppose $\beta_1, \ldots, \beta_n$ generate H. For each β_i let $\alpha_i \in \overline{K}$ be a root of $X^N - \beta_i$. Let G' be the subsemigroup of $\overline{K}$ generated by $\alpha_1, \ldots, \alpha_n$ together with the N-th roots of unity in $\overline{K}$ (of which there are at most N, since they are the roots of $X^N - 1$).

We claim $G = G'$. The inclusion $G' \subseteq G$ holds by definition. Suppose $\gamma \in G$. Then $\gamma^N = \beta_1^{k_1} \cdots \beta_n^{k_n}$ for some $k_i \geq 0$. Define $\gamma' := \alpha_1^{k_1} \cdots \alpha_n^{k_n} \in G'$. Then $(\gamma')^N = \gamma^N$. It follows that $\gamma = \gamma'\zeta$ with ζ an N-th root of unity (whether or not $\gamma = 0$). So $\gamma \in G'$. ◀

Proof of Theorem 28. We have to show that there exists a finitely generated multiplicative semigroup $G \subseteq \overline{K}$ and an $N \geq 1$ such that for every u, w, $v \in \Sigma^*$, the characteristic roots of $(f(uw^n v))_n$ are contained in G and $\alpha^N \in K$ for all $\alpha \in G$.

Let $\mathcal{A} = (d, I, M, F)$ be a polynomially ambiguous weighted automaton recognising f. Let $N \coloneqq d!$. Without restriction, we can take $\mathcal{A}$ to be trim. Let $H \subseteq K$ be the subsemigroup of K generated by all the finitely many transition weights of $\mathcal{A}$, and let $G = \{\, \alpha \in \overline{K} \mid \alpha^N \in H \cup \{1\} \,\}$. By Lemma 32, the semigroup G is finitely generated.

The characteristic roots of the LRS $(\mathcal{A}(uw^n v))_n$ are eigenvalues of $M(w)$. By Lemma 31, the eigenvalues of $M(w^N)$ are products of transition weights. We have $M(w^N) = M(w)^N$, and so the eigenvalues of $M(w)$ are roots of degree N of products of transition weights of $\mathcal{A}$. This means they belong to G. ◀

We (ambitiously) conjecture the following converse of Theorem 28.

▶ **Conjecture 33.** *Let $f\colon \Sigma^* \to K$ be recognised by a weighted automaton. If there exists a finitely generated multiplicative subsemigroup $G \subseteq \overline{K}$ and $N \in \mathbb{Z}_{\geq 1}$ such that*

- *the characteristic roots of every sequence in $\mathrm{PSF}(f)$ are contained in G,*
- *and $\alpha^N \in K$ for all $\alpha \in G$,*

then f is recognised by a polynomially ambiguous weighted automaton.

Conjecture 33 postulates a pumping-style characterisation. The following conjecture postulates a "global" characterisation, with a similar restriction as in Conjecture 33 imposed on the eigenvalues of the matrix semigroup. Here it is important that the condition is imposed on *all* matrices, not just on the generators.

▶ **Conjecture 34.** *Let $f\colon \Sigma^* \to K$ be recognised by a weighted automaton. If there exists a finitely generated multiplicative subsemigroup $G \subseteq \overline{K}$ and $N \geq 1$ such that*

- *all eigenvalues of matrices $M(w)$ for $w \in \Sigma^*$ are contained in G,*
- *and $\alpha^N \in K$ for all $\alpha \in G$,*

then f is recognised by a polynomially ambiguous weighted automaton.

A positive resolution of the conjectures would extend a characterisation in similar spirit of functions that can be recognised by unambiguous weighted automata [4]. While the conjectures seem ambitious, in the preprint [25], Conjecture 34 was already proved in the case that all transition matrices are invertible.

The following lemma shows that Conjectures 33 and 34 are in fact equivalent.

▶ **Lemma 35.** *Let $\mathcal{A} = (d, I, M, F)$ be a minimal weighted automaton and let $w \in \Sigma^*$. Then the set of nonzero eigenvalues of $M(w)$ is precisely the set of all nonzero characteristic roots of the LRS $(\mathcal{A}(uw^n v))_n$ as u, $v \in \Sigma^*$ range through all words.*

Proof. One direction is obvious – characteristic roots come from eigenvalues of the matrix $M(w)$. We only have to show that every nonzero eigenvalue $\lambda \in \overline{K}$ of $M(w)$ shows up as characteristic root of some LRS.

Working over $\overline{K}$ we can assume that $K = \overline{K}$ is algebraically closed. This allows us to change to a basis in which $M(w)$ is in the Jordan normal form. In particular, we can assume that $M(w)$ is upper triangular and $M(w)[1,1] = \lambda$. Let $e_1 = (1, 0, \ldots, 0) \in K^{d \times 1}$ and let $e_1^\intercal$ be its transpose. Then $\lambda^n = e_1^\intercal M(w^n) e_1$.

Because $\mathcal{A}$ is minimal, the reachability set $\{\, I^\intercal M(w) \mid w \in \Sigma^* \,\}$ spans $K^{1 \times d}$ as a vector space, and analogously the coreachability set spans $K^{d \times 1}$ – otherwise we could easily decrease the dimension. Therefore, there exist α_i, $\beta_j \in K$ and u_i, $v_j \in \Sigma^*$ such that $e_1^\intercal = \sum_{i=1}^d \alpha_i I^\intercal M(u_i)$ and $e_1 = \sum_{j=1}^d M(v_j) F \beta_j$. Now

$$\lambda^n = \sum_{i=1}^{d} \sum_{j=1}^{d} \alpha_i \beta_j \, I^\intercal M(u_i w^n v_j) F,$$

expresses the LRS $(\lambda^n)_n$ as linear combination of LRS $(I^\intercal M(u_i w^n v_j)F)_n$. Since the former has a characteristic root λ, a summand must have λ as a characteristic root as well: this is easily seen by considering the LRS as rational functions, and recalling that nonzero characteristic roots correspond to reciprocals of poles, or by the uniqueness result in [19, Theorem 39]. ◀

While the main theorem of this section provides a necessary pumping criterion for polynomially ambigualisable automata, that is, those weighted automata that are equivalent to polynomial ambiguous ones, another related open problem is to relate the minimal degree of the polynomial bounding the ambiguity to arithmetic properties of the output (in other words, to characterise linearly ambiguous, quadratically ambiguous, etc.). At least in characteristic zero, a tempting idea is to look at the degrees of polynomials arising in the PSF. Indeed, it is easy to see that if the ambiguity of the automaton is bounded by a polynomial of degree d, then no polynomial of higher degree can appear in the PSF. The converse however does not hold, as the following example shows.

▶ **Example 36.** The function $f\colon \{0,1\}^* \to \mathbb{Q}$, mapping a binary word to the natural number it represents (say, LSB on the left), is easily seen to be recognisable by a weighted automaton. We check that, for any u, w, $v \in \Sigma^*$, the exponential polynomial representation of $(\mathcal{A}(uw^n v))_n$ only contains constant polynomials. Indeed, let $u = u_1 u_2 \dots u_r$, $w = w_1 w_2 \dots w_t$, $v = v_1 v_2 \dots v_l$. We have

$$f(uw^n v) = 2^0 u_1 + 2^1 u_2 + \cdots + 2^{r-1} u_r + (2^r w_1 + 2^{r+1} w_2 + \cdots + 2^{r+t-1} w_t)(1 + 2^t + \cdots + 2^{t(n-1)})$$
$$+ 2^{nt+r} v_1 + \cdots + 2^{nt+r+l-1} v_l = \alpha + \beta \sum_{i=0}^{n-1} 2^{ti} = \alpha + \beta \frac{2^{tn}-1}{2^t - 1} \quad (\alpha, \beta \in \mathbb{Q}).$$

This gives us an exponential polynomial with only constant polynomials.

A set of the form $\{ g_1 + \cdots + g_m \mid m \leq M, g_i \in G \}$ for some $M \geq 0$ and a finitely generated subgroup $G \leq \mathbb{Q}^\times$ is called a Bézivin set [25]. One can show that $\mathbb{N}$ is not a Bézivin set.[1] It is also easy to see that the output set of a finitely ambiguous weighted automaton is a Bézivin set. Since $f(\Sigma^*) = \mathbb{N}$, the function f cannot be recognised by a finitely ambiguous weighted automaton.

5 Equivalence and Zeroness of CCRA

The two problems are defined as follows (for any classes of automata):

- *equivalence*: given two automata $\mathcal{A}$ and $\mathcal{B}$, decide if $\mathcal{A}(w) = \mathcal{B}(w)$ for all $w \in \Sigma^*$.
- *zeroness*: given an automaton $\mathcal{A}$, decide if $\mathcal{A}(w) = 0$ for all w.

It is folklore that for (polynomially) weighted automata and CCRA the two problems are effectively interreducible. Indeed, to decide zeroness of $\mathcal{A}$ it suffices to check equivalence with $\mathcal{B}$ that outputs 0 on all words. Conversely, to check equivalence of $\mathcal{A}$ and $\mathcal{B}$ one can check zeroness of $\mathcal{A} - \mathcal{B}$, which can be efficiently constructed for these models. Therefore we will deal only with zeroness.

For polynomially ambiguous weighted automata, even unrestricted weighted automata, we know that zeroness is in polynomial time [28] and in NC^2 [29]. Thus, we focus on the complexity of zeroness for CCRA. For the problem to make sense we need to introduce the

[1] This is a consequence of a theorem of Bézivin [7, Th. 4]: if $\mathbb{N}$ were Bézivin, its generating series $\sum_{n=0}^{\infty} n x^n = \frac{x}{(1-x)^2}$ would have to have simple poles only, which is not the case.

size of the input CCRA. Given $\mathcal{C} = (Q, q_0, d, \Sigma, \delta, \mu, \nu)$, we say that its size is $|Q| + d + |\Sigma| + \max_p(|p|)$, where p ranges over all polynomials and constants used in δ, μ and ν. We assume that polynomials are represented in the natural succinct form of arithmetic tree circuits, not as a list of all monomials.

Recall that in the update function one cannot use polynomials like x^2 because two copies of x are needed. However, in some sense CCRA can evaluate any polynomial. For example there is a CCRA $\mathcal{C}$ such that $\mathcal{C}(1^n) = n^2$, simply by having two registers storing n and defining the output function as their product. We can say that evaluating x^2 requires two copies of x. We generalise this observation to any polynomial. Given $x_1, \ldots, x_k$, we say that a polynomial $p(x_1, \ldots, x_k)$ is d-copyless if there exists a copyless polynomial $p'(\mathbf{x})$, where $\mathbf{x} = x_{1,1}, \ldots, x_{1,k}, \ldots x_{d,1}, \ldots, x_{d,k}$ (d copies of every x_i) such that $p'(\mathbf{x}) = p(x_1, \ldots, x_k)$, substituting $x_{i,j} = x_j$ for all $1 \leq i \leq d$ and $1 \leq j \leq k$. In particular 1-copyless is copyless.

We will use a standard and convenient lemma that allows us to turn formulas into polynomials. Note that we assume that formulas, like polynomials, are represented as tree circuits. By the size of the formula, denoted $|\varphi|$, we understand the size of the circuit. The proof can be found in [19, Appendix B].

▶ **Lemma 37.** *Let $\mathbf{x} = (x_1, \ldots, x_k)$ and let $\varphi(\mathbf{x})$ be a Boolean quantifier free formula. There exists a polynomial $p(\mathbf{x})$, of size polynomial in $|\varphi|$, such that for every $\mathbf{v} \in \{0,1\}^k$ we have: $p(\mathbf{v}) \in \{0,1\}$; and $p(\mathbf{v}) = 1$ if and only if $\varphi(\mathbf{v})$ evaluates to true. Moreover, the polynomial $p(\mathbf{x})$ is $|\varphi|$-copyless.*

We are ready to prove the main theorem.

▶ **Theorem 4.** *Zeroness and equivalence problems are PSPACE-complete for CCRA over $\mathbb{Q}$.*

Proof. Regarding the upper bound, by [19, Lemma 50], we know that a CCRA can be translated to a weighted automaton of exponential size. It is known that the equivalence problem for weighted automata is in NC^2 [29]. Since problems in NC^2 can be solved sequentially in polylogarithmic space [27], this essentially yields a PSPACE algorithm. One has to take care that the weighted automaton is not fully precomputed (as it would require too much space). A standard approach computing the states and transitions on the fly solves this issue. See e.g. [17, Section 6.1] for a similar construction.

The rest of the proof is devoted to the matching PSPACE lower bound. We reduce from the validity problem for Quantified Boolean Formulas (QBF), which is known to be PSPACE-complete [23, Theorem 19.1]. One can assume the input is a formula of the form

$$\psi = \forall x_1 \exists y_1 \ldots \forall x_k \exists y_k \;\; \varphi(x_1, y_1, \ldots, x_k, y_k), \tag{1}$$

where φ is quantifier-free. The variables x_i and y_i alternate, x_i are quantified universally and y_i are quantified existentially. For simplicity, we write $\mathbf{x} = (x_1, \ldots, x_k)$ and $\mathbf{y} = (y_1, \ldots, y_k)$. We write $\varphi(\mathbf{x}, \mathbf{y})$ instead of $\varphi(x_1, y_1, \ldots, x_k, y_k)$. Given $\mathbf{v} \in \{0,1\}^{2k}$ we denote by $\varphi(\mathbf{v})$ the truth value of the formula φ with all variables evaluated according to $\mathbf{v}$.

For the reduction, we will need to go through many evaluations of x_i and y_i in a way that respects the quantifiers. It will be convenient to define these evaluations using auxiliary formulas. Let $\mathbf{x}'$ and $\mathbf{y}'$ be fresh copies of variables in $\mathbf{x}$ and $\mathbf{y}$ all of dimension k. We define three quantifier-free formulas: $\textsc{start}(\mathbf{x}, \mathbf{y})$, $\textsc{next}(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}')$ and $\textsc{end}(\mathbf{x}, \mathbf{y})$, as follows.

$$\textsc{start}(\mathbf{x}, \mathbf{y}) = \bigwedge_{i=1}^{k} \neg x_i, \qquad \textsc{end}(\mathbf{x}, \mathbf{y}) = \bigwedge_{i=1}^{k} x_i.$$

Note that START and END do not use $\mathbf{y}$, but in this form it will be easier to state the claim later explaining their purpose. We also define

$$\text{NEXT}(\mathbf{x},\mathbf{y},\mathbf{x}',\mathbf{y}') = \bigwedge_{i=1}^{k} \left(\neg x_i \wedge \bigwedge_{j=i+1}^{k} x_j\right) \implies \left(x_i' \wedge \bigwedge_{j=i+1}^{k} \neg x_j' \;\wedge\; \bigwedge_{j=1}^{i-1} (x_j \iff x_j') \wedge (y_j \iff y_j')\right).$$

To understand the formulas, it is easier to ignore the $\mathbf{y}$ and $\mathbf{y}'$ variables at first. Then these formulas essentially encode a binary counter with k bits: START encodes that all x_i are 0; END encodes that all x_i are 1; and NEXT encodes that $\mathbf{x}'$ is $\mathbf{x}$ increased by 1 in binary. The values of $\mathbf{y}$ can be guessed to anything in START and END. In NEXT we keep consistently the guessed existential values for all unchanged universal variables. The following lemma formally states the purpose of the formulas.

▷ Claim 38. The formula ψ in Equation (1) is valid if and only if there exists a sequence $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \{0,1\}^{2k}$ such that:

1. $\varphi(\mathbf{v}_i)$ is true for all $1 \leq i \leq n$;
2. $\text{START}(\mathbf{v}_1)$ is true;
3. $\text{NEXT}(\mathbf{v}_i, \mathbf{v}_{i+1})$ is true for all $1 \leq i \leq n-1$;
4. $\text{END}(\mathbf{v}_n)$ is true.

Proof. The formulas START, NEXT and END are defined in such a way that they go through all possible evaluations of universal variables, guessing consistently the values for existential variables. The first condition guarantees that ψ is valid. ◁

Thanks to Claim 38 we will not need to differentiate between universal and existential variables. In the following we will implicitly use Lemma 37. To avoid additional notation we will write formula names for their corresponding polynomials. Let $\ell = \max\{|\text{START}|, |\text{END}|, |\text{NEXT}|, |\varphi|\}$, then all polynomials corresponding to these formulas are ℓ-copyless. To ease the notation we write

$$\mathbf{z} = (x_1^1, \ldots, x_k^1, y_1^1, \ldots, y_k^1, \ldots, x_1^\ell, \ldots, x_k^\ell, y_1^\ell, \ldots, y_k^\ell)$$

for ℓ identical copies of vectors of variables in $\mathbf{x}$ and $\mathbf{y}$. Note that identical copies occur on indices equal modulo $2k$ (this will be useful when defining the transitions). The number of copies will be sufficient to evaluate all polynomials corresponding to the formulas in a copyless manner. To emphasise this, we will write $\text{START}(\mathbf{z})$, $\text{END}(\mathbf{z})$ and $\text{NEXT}(\mathbf{z})$.

We are ready to define the CCRA $\mathcal{C} = (Q, p_0, d, \Sigma, \delta, \mu, \nu)$, where: $Q = \{\, p_i, q_i \mid 0 \leq i \leq 2k \,\}$; $d = 8\ell k + 1$; $\Sigma = \{0, 1, \#\}$. We denote the $8\ell k + 1$ variables as follows: $\mathbf{z}$, $\mathbf{z}'$, $\mathbf{z}''$, $\mathbf{z}^{\text{old}}$ and s. That is: four disjoint copies corresponding to ℓ copies of $\mathbf{x}$, $\mathbf{y}$ and one extra variable s. We denote the variables in the copies by z_i, z_i', z_i'', z_i^{old} for $1 \leq i \leq 2\ell k$. The initial function assigns the value 1 to all variables. It will be important that $\mu(s) = 1$; for all other variables the initial value could be arbitrary. The final function is defined by $\nu(x) = 0$ for all $x \in Q \setminus \{q_0\}$ and $\nu(q_0) = s \cdot \text{END}(\mathbf{z}^{\text{old}})$.

Before we define the transitions we give an intuition on how the automaton works. We call a subword of length $2k$ a block. The automaton will read a sequence of blocks which correspond to consecutive evaluations $\mathbf{v}_i$ from Claim 38 and store them in multiple copies of $\mathbf{x}$ and $\mathbf{y}$. After reading every block the automaton will check whether: NEXT holds with the previous block; and whether φ holds on the current block. As an invariant, the register s will have value 1 if no error has been detected, and 0 otherwise.

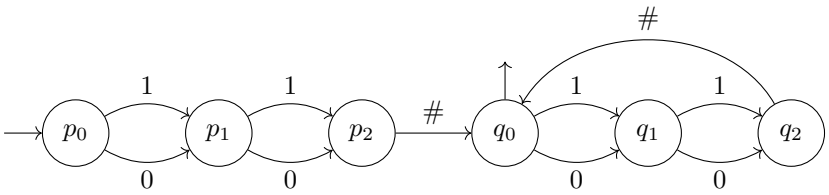


Figure 10 Example for $k = 1$. The state q_0 has an outgoing edge as it is the only one that has a possibly nonzero output.

Most of the transitions will initialise some registers. Given a set of variables Z and $b \in \{0, 1\}$, we define the copyless polynomial map $P_{Z,b}$ as: $P_{Z,b}(z) = b$ for $z \in Z$ and $P_{Z,b}(z) = z$ otherwise. In words, the variables in Z are initialised to b and all others keep their previous value. We will use one type of sets Z, defined as follows: $Z_i = \{ z_j, z'_j, z''_j \mid j \equiv i \mod 2k \}$. This will allow us to remember 3ℓ copies at once.

Formally, we define the transitions as follows (see Figure 10 for the shape of the automaton without the register updates):

1. $\delta(p_{i-1}, b) = (p_i, P_{Z_i,b})$ for all $1 \le i \le 2k$ and $b \in \{0, 1\}$.
2. $\delta(q_{i-1}, b) = (q_i, P_{Z_i,b})$ for all $1 \le i \le 2k$ and $b \in \{0, 1\}$.
3. $\delta(p_{2k}, \#) = (q_0, Q_0)$, where Q_0 resets all variables to 0 except for: $\mathbf{z}^{\text{old}}$ where it puts the content of $\mathbf{z}''$, i.e., $Q_0(z_i^{\text{old}}) = z''_i$ for all $1 \le i \le 2\ell k$; and $Q_0(s) = s \cdot \textsc{start}(\mathbf{z}') \cdot \varphi(\mathbf{z})$.
4. $\delta(q_{2k}, \#) = (q'_0, R_0)$, where R_0 resets all variables to 0 except for: $\mathbf{z}^{\text{old}}$ where it puts the content of $\mathbf{z}''$, i.e., $Q_0(z_i^{\text{old}}) = z''_i$ for all $1 \le i \le 2\ell k$; and $Q_0(s) = s \cdot \textsc{next}(\mathbf{z}^{\text{old}}, \mathbf{z}') \cdot \varphi(\mathbf{z})$.

Note that all polynomials are copyless.

The proof that the reduction works follows essentially from Claim 38. The transitions in Item 1 and Item 2 guess the evaluations $\mathbf{v}_i$. These are stored in three copies: $\mathbf{z}$, $\mathbf{z}'$, $\mathbf{z}''$. The remaining two transitions verify the correctness of these evaluations, i.e., whether they satisfy the conditions in Claim 38. Note that as an invariant these transitions keep in $\mathbf{v}^{\text{old}}$ the previous valuation. In both Item 3, Item 4 we check whether $\varphi(\mathbf{v}_i)$ holds. Additionally, in Item 3 we check whether $\textsc{start}(\mathbf{v}_1)$ is true; and in Item 4 we check whether $\textsc{next}(\mathbf{v}_{i-1}, \mathbf{v}_i)$ is true. All checks are multiplied into the register s, which becomes 0 if any error occurs, and remains 1 otherwise. Finally, the output function guarantees that a nonzero value can be output only if $\textsc{end}(\mathbf{v}_n)$ holds. ◀

References

1 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. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.65.

2 Corentin Barloy, Nathanaël Fijalkow, Nathan Lhote, and Filip Mazowiecki. A robust class of linear recurrence sequences. *Inf. Comput.*, 289(Part):104964, 2022. doi:10.1016/J.IC.2022.104964.

3 Jason Bell and Daniel Smertnig. Noncommutative rational Pólya series. *Selecta Math. (N.S.)*, 27(3):Paper No. 34, 34, 2021. doi:10.1007/s00029-021-00629-2.

4 Jason P. Bell and Daniel Smertnig. Computing the linear hull: Deciding deterministic? and unambiguous? for weighted automata over fields. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023, Boston, MA, USA, June 26-29, 2023*, pages 1–13. IEEE, 2023. doi:10.1109/LICS56636.2023.10175691.

5 Yahia Idriss Benalioua, Nathan Lhote, and Pierre-Alain Reynier. Minimizing Cost Register Automata over a Field. In Rastislav Královic and Antonín Kucera, editors, *49th International Symposium on Mathematical Foundations of Computer Science, MFCS 2024, August 26-30, 2024, Bratislava, Slovakia*, volume 306 of *LIPIcs*, pages 23:1–23:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.MFCS.2024.23.

6 Jean Berstel and Christophe Reutenauer. *Noncommutative Rational Series with Applications*, volume 137 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2011.

7 Jean-Paul Bézivin. Sur un théorème de G. Pólya. *J. Reine Angew. Math.*, 364:60–68, 1986. doi:10.1515/crll.1986.364.60.

8 Michaël Cadilhac, Filip Mazowiecki, Charles Paperman, Michal Pilipczuk, and Géraud Sénizergues. On Polynomial Recursive Sequences. *Theory Comput. Syst.*, 68(4):593–614, 2024. doi:10.1007/S00224-021-10046-9.

9 Agnishom Chattopadhyay, Filip Mazowiecki, Anca Muscholl, and Cristian Riveros. Pumping lemmas for weighted automata. *Log. Methods Comput. Sci.*, 17(3), 2021. doi:10.46298/LMCS-17(3:7)2021.

10 Wojciech Czerwinski, Engel Lefaucheux, Filip Mazowiecki, David Purser, and Markus A. Whiteland. The boundedness and zero isolation problems for weighted automata over nonnegative rationals. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 15:1–15:13. ACM, 2022. doi:10.1145/3531130.3533336.

11 Laure Daviaud, Marcin Jurdzinski, Ranko Lazic, Filip Mazowiecki, Guillermo A. Pérez, and James Worrell. When are emptiness and containment decidable for probabilistic automata? *J. Comput. Syst. Sci.*, 119:78–96, 2021. doi:10.1016/J.JCSS.2021.01.006.

12 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer Science & Business Media, 2009.

13 Graham Everest, Alfred Jacobus Van Der Poorten, Igor Shparlinski, Thomas Ward, et al. *Recurrence Sequences*, volume 104. American Mathematical Society Providence, RI, 2003.

14 Nathanaël Fijalkow, Cristian Riveros, and James Worrell. Probabilistic automata of bounded ambiguity. *Inf. Comput.*, 282:104648, 2022. doi:10.1016/J.IC.2020.104648.

15 Vesa Halava, Tero Harju, Mika Hirvensalo, and Juhani Karhumäki. Skolem's Problem — On the Border between Decidability and Undecidability. *TUCS Technical Reports*, 683, 2005.

16 Kosaburo Hashiguchi. Algorithms for Determining Relative Star Height and Star Height. *Inf. Comput.*, 78(2):124–169, 1988. doi:10.1016/0890-5401(88)90033-8.

17 Ismaël Jecker, Filip Mazowiecki, and David Purser. Determinisation and Unambiguisation of Polynomially-Ambiguous Rational Weighted Automata. In Pawel Sobocinski, Ugo Dal Lago, and Javier Esparza, editors, *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, pages 46:1–46:13. ACM, 2024. doi:10.1145/3661814.3662073.

18 Peter Kostolányi. Polynomially Ambiguous Unary Weighted Automata over Fields. *Theory Comput. Syst.*, 67(2):291–309, 2023. doi:10.1007/S00224-022-10107-7.

19 Filip Mazowiecki, Antoni Puch, and Daniel Smertnig. Pumping-like results for copyless cost register automata and polynomially ambiguous weighted automata. *CoRR*, abs/2502.07356, 2025. doi:10.48550/arXiv.2502.07356.

20 Filip Mazowiecki and Cristian Riveros. Maximal Partition Logic: Towards a Logical Characterization of Copyless Cost Register Automata. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*, pages 144–159. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CSL.2015.144.

21 Filip Mazowiecki and Cristian Riveros. Copyless cost-register automata: Structure, expressiveness, and closure properties. *J. Comput. Syst. Sci.*, 100:1–29, 2019. doi:10.1016/J.JCSS.2018.07.002.

22 Joël Ouaknine and James Worrell. On linear recurrence sequences and loop termination. *ACM SIGLOG News*, 2(2):4–13, 2015. doi:10.1145/2766189.2766191.

23 Christos H. Papadimitriou. Computational complexity. In *Encyclopedia of Computer Science*, pages 260–265. 2003.

24 Azaria Paz. *Introduction to Probabilistic Automata*. Academic Press, 1971.

25 Antoni Puch and Daniel Smertnig. Factoring through monomial representations: Arithmetic characterizations and ambiguity of weighted automata, 2024. Preprint. doi:10.48550/arXiv.2410.03444.

26 Christophe Reutenauer. On Polya series in noncommuting variables. In Lothar Budach, editor, *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arthmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*, pages 391–396. Akademie-Verlag, Berlin, 1979.

27 Walter L. Ruzzo. On Uniform Circuit Complexity. *J. Comput. Syst. Sci.*, 22(3):365–383, 1981. doi:10.1016/0022-0000(81)90038-6.

28 Marcel Paul Schützenberger. On the Definition of a Family of Automata. *Inf. Control.*, 4(2-3):245–270, 1961. doi:10.1016/S0019-9958(61)80020-X.

29 Wen-Guey Tzeng. On Path Equivalence of Nondeterministic Finite Automata. *Inf. Process. Lett.*, 58(1):43–46, 1996. doi:10.1016/0020-0190(96)00039-7.