Polynomial Automata: Zeroness and Applications

Michael Benedikt*, Timothy Duff†, Aditya Sharad*, and James Worrell*
*Department of Computer Science, University of Oxford, UK
†School of Mathematics, Georgia Institute of Technology, USA

***Abstract*—We introduce a generalisation of weighted automata over a field, called polynomial automata, and we analyse the complexity of the Zeroness Problem in this model, that is, whether a given automaton outputs zero on all words. While this problem is non-primitive recursive in general, we highlight a subclass of polynomial automata for which the Zeroness Problem is primitive recursive. Refining further, we identify a subclass of affine VAS for which coverability is in 2EXPSPACE. We also use polynomial automata to obtain new proofs that equivalence of streaming string transducers is decidable, and that equivalence of copyless streaming string transducers is in PSPACE.**

I. Introduction

Systems with polynomial dynamics arise in a number of areas of computer science, including control theory [1], hybrid systems [2], and program analysis [3]. In this paper our focus is on polynomial systems in the context of automata theory. We define a class of *polynomial automata*: finite state machines with input, output, and discrete-time polynomial dynamics. Such automata can be seen both as generalisations of vector addition systems (also known as VAS or Petri Nets) and of weighted automata over a field [4]. We concentrate on two basic decision problems for polynomial automata, Zeroness and Equivalence, and illustrate connections with other decision problems in automata theory, including equivalence of streaming string transducers and coverability for vector addition systems.

We begin by introducing the model of polynomial automata, an *extended* variant capable of generalising vector addition systems, and the decision problems associated with these automata. We demonstrate a reduction of the Coverability Problem for the class of reset VAS to the Non-Zeroness Problem, the complement of the Zeroness Problem, for polynomial automata. Since the Coverability Problem for reset VAS is known to be non-primitive recursive, we immediately obtain the result that the Zeroness Problem for polynomial automata is non-primitive recursive. We then show that the Zeroness Problem is decidable, with Ackermannian complexity, using techniques related to those used to analyse VAS coverability [5]. This motivates the consideration of conditions that lead to primitive recursive decidability of the Zeroness Problem.

Next, we define a class of so-called *invertible polynomial automata*. Roughly speaking, these are polynomial automata in which each transition function has a rational inverse. We show that the Non-Zeroness Problem for invertible polynomial automata is primitive recursive, though non-elementary. Refining a previous construction, we show that the Coverability Problem for affine VAS that have injective transition functions reduces to the Non-Zeroness Problem for invertible polynomial automata, and in this case we can moreover obtain elementary bounds. This last result suggests a new perspective on "whole-place" operations in VAS: specifically that the Coverability Problem remains primitive recursive provided that such operations transform configurations injectively. Notice that such a restriction excludes reset VAS, consistent with the fact that the Coverability Problem for reset VAS is non-primitive recursive.

We then turn to classes of polynomial automata that arise from word transducers. We show that polynomial automata can be used to simulate *streaming string transducers*, which were formulated in [6] for the verification of list-processing programs, and shown in [7] to encompass the set of MSO-definable transductions. Using this, we show that the equivalence problem for general streaming string transducers (which was first proved decidable in [8]) is Ackermannian. We then define a class of *copyless polynomial automata*, and show that the Zeroness Problem for this class is in PSPACE. The definition of copyless polynomial automata is inspired by the notion of copyless streaming string transducers [6], and we give a reduction of the equivalence problem for copyless streaming string transducers to the Zeroness Problem for copyless polynomial automata, thereby giving a new proof that the former problem is decidable in polynomial space.

Thus polynomial automata give a common generalisation of vector addition systems and weighted automata, and also provide another method for bounding the complexity of analysis problems on transducers.

Organisation: We start by giving some background that we need from algebraic geometry, complexity theory, and automata theory in Section III. We then define our new automaton model in Section IV along with the basic problem that we study, Non-Zeroness. Section V gives a decidability result for the Non-Zeroness Problem for general polynomial automata, along with a coarse bound on the computational complexity of the problem. Section VI introduces invertible polynomial automata, and provides a primitive recursive bound for the Non-Zeroness Problem for these. It also shows that the invertible polynomial automata subsumes a rich class of vector addition systems, and that for this subclass we can get an elementary upper bound. Sections VII and VIII apply polynomial automata to the analysis of streaming string transducers. The paper closes with conclusions and open issues in Section IX.

II. Acknowledgments

We thank the referees for many corrections and suggestions. The work of Benedikt was funded by EPSRC grants EP/M005852/1, EP/N014359/1, and EP/L012138/1. The work of Worrell was funded by EPSRC grant EP/M012298/1.

A. Related Work

To the best of our knowledge, the use of machines with polynomial register transformations as weighted language acceptors has not been explicitly explored in the literature. However, there are a number of related formalisms that extend VAS or Petri Nets. For example, the reachability problem for one-dimensional polynomial automata was shown to be PSPACE-complete in [9]. Polynomial automata are similar to the extensions of Petri Nets given in [10], but the Zeroness Problem that we focus on here is not studied in [10]. Another related work on VAS is [11], which defines the class of *strongly increasing* affine vector addition systems and shows that coverability is EXPSPACE-complete for this class.

Our upper bounds on the complexity of the Zeroness Problem are based on an analysis of the length of increasing chains of polynomial ideals and the corresponding decreasing chains of varieties. Ackermannian upper and lower bounds on chains of ideals have been studied in a number of settings; see in particular [12]. However lower bounds on the lengths of chains do not imply corresponding lower bounds on computational problems. Novikov and Yakovenko [13], [14] introduce conditions that lead to primitive recursive upper bounds on decreasing chains of varieties. The results of [13], [14] underly our complexity bounds on invertible polynomial automata in Section VI.

Many of the results in this paper are taken from the unpublished MSc theses [15] and [16].

III. Preliminaries

In this section we start by summarising some basic notions from algebraic geometry that will be used in this paper. These can be found in any standard text, such as [17]. We also recall some relevant notions from order theory that underlie both Hilbert's Basis Theorem and the notion of the dimension of an algebraic variety.

1) Polynomials: Let $\mathbb{F}$ be a field and $\mathbb{F}[x_1,\dots,x_n]$ the corresponding ring of polynomials. A *monomial ordering* is a well-founded total ordering on monomials in $\mathbb{F}[x_1,\dots,x_n]$ that respects multiplication: if $\boldsymbol{x}^{\boldsymbol{\alpha}} \leq \boldsymbol{x}^{\boldsymbol{\beta}}$ for some $\boldsymbol{\alpha},\boldsymbol{\beta} \in \mathbb{N}^n$ then $\boldsymbol{x}^{\boldsymbol{\alpha}+\boldsymbol{\gamma}} \leq \boldsymbol{x}^{\boldsymbol{\beta}+\boldsymbol{\gamma}}$ for all $\boldsymbol{\gamma} \in \mathbb{N}^n$. Fix a monomial ordering on $\mathbb{F}[x_1,\dots,x_n]$, and write leading Term $\mathrm{LT}(f)$ for the largest monomial of a polynomial f with respect to this ordering. By extension, write $\mathrm{LT}(I) := \{\mathrm{LT}(f) : f \in I\}$ for a set of polynomials $I \subseteq \mathbb{F}[x_1,\dots,x_n]$.

2) Order Theory: Given $n \in \mathbb{N}$, we consider $\mathbb{N}^n$ under the pointwise order, defined by $\boldsymbol{u} \leq \boldsymbol{v}$ if $u_1 \leq v_1,\dots,u_n \leq v_n$. Dickson's Lemma states that the pointwise order is a well-quasi-order: any decreasing sequence of downward-closed sets must terminate. An *order ideal* in $\mathbb{N}^n$ is a directed downward-closed set $C \subseteq \mathbb{N}^n$. Such a set has the form

$$C = \{\boldsymbol{u} \in \mathbb{N}^n \mid u_{i_1} \leq a_1,\dots,u_{i_k} \leq a_k\},$$

for given fixed indices $1 \leq i_1 < \ldots < i_k \leq n$ and natural numbers $a_1,\dots,a_k$. We define the norm of C to be $\|C\| := \max_{1\leq i\leq k} a_k$ and we define the *dimension* of C to be $n-k$. Any downward-closed set $S \subseteq \mathbb{N}^n$ can be decomposed into a finite union of order ideals. If only maximal ideals are used, this gives a unique, canonical ideal decomposition of S. This allows us to extends the above norm from order ideals to all downwards-closed sets in $\mathbb{N}^n$: if S has canonical ideal decomposition $\bigcup_{1\leq j\leq l} C_j$, then we define its norm to be $\|S\| := \max_{1\leq j\leq l} \|C_j\|$.

3) Ideals and Varieties: Recall that an *(affine) variety* $V \subseteq \mathbb{F}^n$ is the set of common roots of a finite set of polynomials in $\mathbb{F}[x_1,\dots,x_n]$. A *polynomial ideal* is an ideal I in the ring $\mathbb{F}[x_1,\dots,x_n]$. Given a set of polynomials $S \subseteq \mathbb{F}[x_1,\dots,x_n]$, possibly infinite! we denote by $\langle S\rangle$ the ideal generated by S. Hilbert's Basis Theorem states that each such ideal I is finitely generated. Hilbert's Basis Theorem entails the *ascending chain condition* for polynomial ideals (an increasing sequence of ideals must terminate) and the *descending chain condition* for varieties (a decreasing sequence of varieties must terminate).

Another consequence of Hilbert's Basis Theorem is that for a polynomial ideal I, $\mathbf{V}(I) := \{\boldsymbol{v} \in \mathbb{F}^n \mid p(\boldsymbol{v}) = 0 \text{ for all } p \in I\}$ is a variety, called the *variety of the ideal* I. If $\mathbb{F}$ is algebraically closed and $V = \mathbf{V}(I)$ for some ideal I, then the dimension of V can be characterised as the maximum dimension of any order ideal $C \subseteq \mathbb{N}^n$ such that $\{\boldsymbol{x}^{\boldsymbol{\alpha}} : \boldsymbol{\alpha} \in C\} \cap \mathrm{LT}(I) = \varnothing$ (see [17, Chapter 9, Section 3, Theorem 8]).

Given an ideal $I \subseteq \mathbb{F}[x_1,\dots,x_n]$, we say a finite subset $\{g_1,\dots,g_m\}$ is a *Gröbner basis* of I if $\langle \mathrm{LT}(I)\rangle = \langle \mathrm{LT}(g_1),\dots,\mathrm{LT}(g_m)\rangle$. We say a Gröbner basis $\{g_1,\dots,g_n\}$ for I is *reduced* if the g_i are monic and no proper subset is a basis for I. A Gröbner basis for I is necessarily a generating set for I. For the field of rationals, a basis may be computed using exponential space via Buchberger's algorithm [17].

A variety V is *irreducible* if whenever $V \subseteq V_1 \cup V_2$ for varieties V_1, V_2 then $V \subseteq V_1$ or $V \subseteq V_2$. If a variety V_1 is contained in another variety V_2, then we say V_1 is a *subvariety* of V_2. Each variety $V \subseteq \mathbb{F}^n$ can be uniquely written as an irredundant union of irreducible subvarieties: $V = V_1 \cup \ldots \cup V_k$, where $V_i \nsubseteq V_j$ for $i \neq j$. Assuming that $\mathbb{F}$ is algebraically closed, if a variety $V \subseteq \mathbb{F}^n$ is the set of zeros of a collection of polynomials of total degree at most d, then the number of irreducible components V of dimension at least s is at most d^{n-s} (cf. [13]).

4) Rational Maps: Recall that a polynomial $f \in \mathbb{F}[x_1,\dots,x_n]$ can naturally be viewed as a function $f : \mathbb{F}^n \to \mathbb{F}$, and that a function $f : \mathbb{F}^m \to \mathbb{F}^n$ is called a *polynomial map* if there are polynomials $f_1,\dots,f_n \in \mathbb{F}[x_1,\dots,x_m]$ such that

$$f(a_1,\dots,a_m) = (f_1(a,\dots,a_m),\dots,f_n(a_1,\dots,a_m))$$

for all $a_1,\dots,a_m \in \mathbb{F}$.

A *rational map* between varieties $V \subseteq \mathbb{F}^m$ and $W \subseteq \mathbb{F}^n$ is a partial function of the form $\phi(\boldsymbol{x}) = (\frac{f_1(\boldsymbol{x})}{g_1(\boldsymbol{x})},\dots,\frac{f_n(\boldsymbol{x})}{g_n(\boldsymbol{x})})$, where each $f_i, g_i \in \mathbb{F}[x_1,\dots,x_m]$, there is some $\boldsymbol{x} \in V$ such that $g_1(\boldsymbol{x}),\dots,g_n(\boldsymbol{x}) \neq 0$ (so ϕ is defined at some point in V), and $\phi(V) \subseteq W$, i.e. $\phi(\boldsymbol{x}) \in W$ for all points $\boldsymbol{x} \in V$ where ϕ is defined. We write $\phi : V \dashrightarrow W$. Observe that polynomial maps are also rational maps that are defined throughout their domain.

Composition of rational maps $\phi : U \dashrightarrow V$ and $\psi : V \dashrightarrow W$ is defined as long as ϕ is defined at some point of U and ψ

poly-rec $\overset{?}{\geq}$ unary polynomial automata?
(without output)

is defined at some point of $\phi(U)$. Equality of rational maps is defined by cross-multiplying the component functions: if $\phi(\boldsymbol{x}) = (\frac{f_1(\boldsymbol{x})}{g_1(\boldsymbol{x})}, \dots, \frac{f_n(\boldsymbol{x})}{g_n(\boldsymbol{x})})$ and $\psi(\boldsymbol{x}) = (\frac{h_1(\boldsymbol{x})}{k_1(\boldsymbol{x})}, \dots, \frac{h_n(\boldsymbol{x})}{k_n(\boldsymbol{x})})$ are rational maps between $V \subseteq \mathbb{F}^m$ and $W \subseteq \mathbb{F}^n$, then we say $\phi = \psi$ if for every $i \in \{1, \dots, n\}$, $f_i k_i - h_i g_i \in I(V)$.

Two varieties $V \subseteq \mathbb{F}^m$ and $W \subseteq \mathbb{F}^n$ are *birationally equivalent* if there is a pair of rational maps $\phi : V \dashrightarrow W$ and $\psi : W \dashrightarrow V$ such that $\psi \circ \phi$ is defined and equal to the identity on V, and $\phi \circ \psi$ is defined and equal to the identity on W, in the manner given above. It is shown by Cox et al. [17] that birationally equivalent varieties have the same dimension.

5) Complexity: We assume familiarity with the standard time and space classes, including the class ELEMENTARY and its complement NONELEMENTARY. We also consider a complexity class and strong notion of hardness given by the *Ackermann function* F_ω, which is formed by diagonalising a sequence of inductively-defined functions F_k with F_0 being linear and $F_{k+1}(n)$ being the n-fold composition of F_k applied to n. It is a standard example of a computable function that is not primitive-recursive. We say a decision problem is in the complexity class ACKERMANN if it is solvable by a Turing Machine running in time at most $F_\omega(p(n))$ on input of size n, where p is a primitive recursive function [18]. We say that a decision problem is ACKERMANN-*hard* if every problem in ACKERMANN can be reduced to the given problem under primitive recursive many-one reductions. This is a strengthening of the notion of hardness considered in [18].

6) Vector Addition Systems: Recall that an *n-dimensional affine vector addition system* (affine VAS) is a finite set $\mathcal{R}$ whose elements are triples $(A, \boldsymbol{a}, \boldsymbol{b})$, where A is an $n \times n$ matrix of non-negative integers and $\boldsymbol{a}, \boldsymbol{b}$ are n-dimensional vectors, with the entries of $\boldsymbol{a}$ lying in $\{0, 1\}$ and the entries of $\boldsymbol{b}$ in $\{-1, 0, 1\}$. Given an affine VAS $\mathcal{R}$, we define a transition relation $\rightarrow \subseteq \mathbb{N}^n \times \mathbb{N}^n$ by $\boldsymbol{u} \rightarrow \boldsymbol{u}'$ if and only if there exists $(A, \boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}$ with $\boldsymbol{u} - \boldsymbol{a} \geq 0$ and $\boldsymbol{u}' = A(\boldsymbol{u} - \boldsymbol{a}) + \boldsymbol{b}$.

Such an affine VAS models a system of n registers, each holding a non-negative integer value, which can be updated by applying transitions in $\mathcal{R}$ to the vector formed by the current register values. Observe that a transition can only be applied to a (non-negative) vector $\boldsymbol{u}$ if the resulting vector $\boldsymbol{u}'$ will remain non-negative.

An *ordinary VAS* is an affine VAS $\mathcal{R}$ where every transition increments, decrements, or leaves unchanged each entry of the vector it is applied to, i.e. for each transition $(A, \boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}$, A is the identity matrix I and $\boldsymbol{a}$ is the zero vector.

A *reset VAS* is an ordinary VAS $\mathcal{R}$ augmented with transitions that reset a given entry of the current vector to zero. Formally, they are affine VAS where for every rule $(A, \boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}$, A is a diagonal matrix with entries in $\{0, 1\}$, and $\boldsymbol{a}$ is the zero vector.

The *Coverability Problem* for any such class of VAS asks, given an n-dimensional VAS $\mathcal{R}$ of the given class and a non-negative *initial vector* $\boldsymbol{u}$, whether there is a vector $\boldsymbol{v}$ with $v_n > 0$ and sequence of transitions from $\boldsymbol{u}$ to $\boldsymbol{v}$.[1]

IV. Polynomial Automata

stateless, deterministic

A. The Basic Model

A *polynomial automaton* is a tuple $\mathcal{A} = (\Sigma, n, \boldsymbol{\alpha}, p, \gamma)$, where Σ is a finite *alphabet*, n is the *dimension*, $\boldsymbol{\alpha} \in \mathbb{Q}^n$ is the *initial configuration*, the *transition function* $p : \Sigma \times \mathbb{Q}^n \to \mathbb{Q}^n$ comprises a family of polynomial maps $p(\sigma, -)$ (which we also write as p_σ) for $\sigma \in \Sigma$, and the *output function* $\gamma : \mathbb{Q}^n \to \mathbb{Q}$ is also a polynomial map.

We extend p to a map $p : \Sigma^* \times \mathbb{Q}^n \to \mathbb{Q}^n$, where p_ε is the identity on $\mathbb{Q}^n$ and $p_{\sigma w} = p_w \circ p_\sigma$ for $\sigma \in \Sigma$ and $w \in \Sigma^*$. We then define the output of $\mathcal{A}$ on a word $w \in \Sigma^*$ to be $[\![\mathcal{A}]\!](w) \overset{\text{def}}{=} \gamma(p_w(\boldsymbol{\alpha}))$.

Note that we operate over the field $\mathbb{Q}$ of rational numbers for computational purposes. When studying the functions computed by these automata in later sections, we may instead work within the algebraic closure $\mathbb{A}$ of $\mathbb{Q}$.

We say that a polynomial automaton $\mathcal{A} = (\Sigma, n, \boldsymbol{\alpha}, p, \gamma)$ is *linear* if the output function γ and each transition function p_σ are linear (i.e., the underlying polynomials are homogeneous and of total degree one). Linear polynomial automata are essentially the same as weighted automata over the field $\mathbb{Q}$ (also called linear representations) as defined in [19].

It is not surprising that polynomial automata can define word functions that are not expressible by linear polynomial automata. The following examples are from [15].

Example 1. *We define a single-state polynomial automaton over the rationals, reading unary words. It has initial configuration 2, transition map $x \mapsto x^2$ and output map the identity. Thus for each input symbol that is read, the automaton squares its current value, and outputs the final value at the end of the input word. Thus on input of length n, this automaton produces 2^{2^n} as output. This dominates any function that solves a linear recurrence, and thus (e.g., by Theorem 5.38 of [20]) cannot be realised by a weighted automaton.*

Example 2. *We now show that polynomial automata can naturally and succinctly model alternation. Specifically, we argue that for any* alternating finite automaton *there is a polynomial automaton of the same size that simulates it. Given a set Q, let $B_+(Q)$ denote the set of Boolean expressions generated from Q using $\wedge$ and $\vee$. Recall that an alternating finite automaton is a tuple $\mathcal{B} = (\Sigma, Q, q_1, \delta, F)$, where Σ is a finite alphabet, $Q = \{q_1, \dots, q_n\}$ is a finite set of states, $q_1 \in Q$ is the initial state, $\delta : Q \times \Sigma \to B_+(Q)$ is the transition function, and $F \subseteq Q$ is the set of final states.*

The acceptance condition of such an alternating automaton can be described as follows. Beginning in the initial Boolean expression q_1, for each input symbol σ read, the automaton updates its current expression by replacing each state q in the expression with $\delta(q, \sigma)$. When the input word has been consumed, the automaton accepts *if and only if the resulting expression evaluates to* true *by replacing all final states in the expression with* true *and non-final states with* false *.*

From such an automaton we derive a (linearly sized) polynomial automaton $\mathcal{A} = (\Sigma, n, \boldsymbol{\alpha}, p, \gamma)$ such that $[\![\mathcal{A}]\!]$ is the characteristic function of the reverse language of $\mathcal{B}$.

[1] Note that we consider the problem of covering a place, which is polynomial-time equivalent to the problem of covering a vector.

First, for each formula $f \in B_+(Q)$ we associate a polynomial $\widetilde{f} \in \mathbb{Q}[x_1,\dots,x_n]$ by induction as follows: $\widetilde{q_i} = x_i$, $\widetilde{f \wedge g} = \widetilde{f} \cdot \widetilde{g}$, and $\widetilde{f \vee g} = \widetilde{f} + \widetilde{g} - \widetilde{f} \cdot \widetilde{g}$. We then define $p(\sigma, \boldsymbol{x})_i = \widetilde{\delta(q_i,\sigma)}$. We let $\gamma(\boldsymbol{x}) = x_1$ and $\boldsymbol{\alpha}$ be the characteristic vector of the set F, i.e., $\alpha_i = 1$ if $q_i \in F$ and $\alpha_i = 0$ if $q_i \notin F$.

For an input word $w = w_1 \dots w_k$, the polynomial $\gamma \circ p_w = \gamma \circ p_{w_k} \circ \dots \circ p_{w_1}$ represents the Boolean expression formed by the alternating automaton $\mathcal{B}$ after reading $rev(w)$, the reverse of the word w, simulating the logical operations $\wedge$ and $\vee$ using polynomial combinations of $\{0,1\}$-valued inputs, as described above. Evaluating such a polynomial on the characteristic vector $\boldsymbol{x} \in \{0,1\}^n$ of a set of states $S \subseteq Q$ will give 1 if S satisfies the Boolean expression (in the manner described) and 0 otherwise. Thus $\mathcal{B}$ accepts the word $rev(w)$ if and only if the set F of final states satisfies the resulting expression, which holds if and only if the polynomial $\gamma \circ p_w$ evaluates to 1 on its characteristic vector $\boldsymbol{\alpha}$, i.e. if and only if $[\![\mathcal{A}]\!](w) = 1$.

The *Equivalence Problem* for polynomial automata asks, given two polynomial automata $\mathcal{A}$ and $\mathcal{B}$ over the same alphabet Σ, whether $[\![\mathcal{A}]\!](w) = [\![\mathcal{B}]\!](w)$ for all $w \in \Sigma^*$. A special case of the equivalence problem is the *Zeroness Problem*, asking whether, given a polynomial automaton $\mathcal{A}$, $[\![\mathcal{A}]\!](w) = 0$ for all $w \in \Sigma^*$. The complement is the *Non-Zeroness Problem*, asking whether a given polynomial automaton $\mathcal{A}$ over an alphabet Σ has $[\![\mathcal{A}]\!](w) \neq 0$ for some word $w \in \Sigma^*$.

The Zeroness Problem for linear polynomial automata is solvable in polylogarithmic space [21]. It is well known that there is a logarithmic-space reduction of the Equivalence Problem for linear polynomial automata to the Zeroness problem (see, e.g., [22]) . This reduction extends to polynomial automata. Specifically, given polynomial automata $\mathcal{A} = (\Sigma, n^{\mathcal{A}}, \boldsymbol{\alpha}^{\mathcal{A}}, p^{\mathcal{A}}, \gamma^{\mathcal{A}})$ and $\mathcal{B} = (\Sigma, n^{\mathcal{B}}, \boldsymbol{\alpha}^{\mathcal{B}}, p^{\mathcal{B}}, \gamma^{\mathcal{B}})$, define another polynomial automaton $\mathcal{C} = (\Sigma, n^{\mathcal{C}}, \boldsymbol{\alpha}^{\mathcal{C}}, p^{\mathcal{C}}, \gamma^{\mathcal{C}})$, where

$$\begin{aligned} n^{\mathcal{C}} &:= n^{\mathcal{A}} + n^{\mathcal{B}} \\ \boldsymbol{\alpha}^{\mathcal{C}} &:= (\boldsymbol{\alpha}^{\mathcal{A}}, \boldsymbol{\alpha}^{\mathcal{B}}) \\ p^{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}) &:= (p^{\mathcal{A}}(\boldsymbol{x}), p^{\mathcal{B}}(\boldsymbol{y})) \\ \gamma^{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}) &:= \gamma^{\mathcal{A}}(\boldsymbol{x}) - \gamma^{\mathcal{B}}(\boldsymbol{y}) . \end{aligned}$$

It is straightforward to show that $[\![\mathcal{C}]\!]$ is identically zero if and only if $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$. As we will see later (Corollary 1), the above reduction gives us a way to decide the Equivalence Problem for polynomial automata.

B. Extended Polynomial Automata

By adding additional constraints on the allowed configurations of an automaton, we can naturally model richer systems.

An *extended polynomial automaton* is a tuple $\mathcal{A} = (\Sigma, n, \xi, \boldsymbol{\alpha}, p, \gamma)$, where the *domain function* $\xi : \mathbb{Q}^n \to \mathbb{Q}$ is a polynomial map, and the other components are as with ordinary polynomial automata. We say that a configuration $\boldsymbol{v} \in \mathbb{Q}^n$ of $\mathcal{A}$ is valid if $\xi(\boldsymbol{v}) \neq 0$,

We extend the notion of the output of $\mathcal{A}$ on a word $w \in \Sigma^*$ to be

$$[\![\mathcal{A}]\!](w) = \begin{cases} \gamma(p_w(\boldsymbol{\alpha})) & \text{if } \xi(p_u(\boldsymbol{\alpha})) \neq 0 \text{ for all } u \leq w \\ 0 & \text{otherwise,} \end{cases}$$

Notice that a run of $\mathcal{A}$ only generates a non-zero output if each configuration along the run is valid.

Observe that a polynomial automaton can be viewed as an extended polynomial automaton where the domain function is a constant non-zero function. We now show the power of these additional constraints.

Example 3. *Recall from Section III-6 that* reset VAS *are ordinary vector addition systems extended with reset transitions. In this example we show that the Coverability Problem for reset VAS reduces to the Non-Zeroness Problem for polynomial automata.*

EXTENDED

Consider the following three conditions on a polynomial automaton $\mathcal{A} = (\Sigma, n, \xi, \boldsymbol{\alpha}, p, \gamma)$

R1 *the initial configuration $\boldsymbol{\alpha} \in \mathbb{Q}^n$ is a vector of non-negative integers,*

R2 *the domain function is*

Uses the fact that decrements are by at most -1

$$\xi(x_1, \dots, x_n) = (x_1 + 1)(x_2 + 1) \dots (x_n + 1) .$$

R3 *for each $\sigma \in \Sigma$ and $i \in \{1, \dots, n\}$ the transition map $p_\sigma(\boldsymbol{x})_i$ is either the zero polynomial or has the form $x_i + c$, where $c \in \{-1, 0, 1\}$,*

If $\mathcal{A}$ satisfies R1—R3 then its transition function behaves like that of a reset VAS. To see this, first notice that a configuration $\boldsymbol{v} \in \mathbb{Z}^n$ is valid if and only if no component equals -1. Thus if $\boldsymbol{v} \in \mathbb{Z}^n$ is a vector of non-negative integers and $\sigma \in \Sigma$, then $p_\sigma(\boldsymbol{v})$ is a valid configuration if it is also non-negative in every component. Since the initial configuration is non-negative, a run of $\mathcal{A}$ consists exclusively of valid configurations if and only if it all configurations in the run are non-negative.

Here the domain function f is used to encode the restriction that a VAS transition can only be applied to a non-negative vector if the resulting vector remains non-negative.

Suppose that the output function is $\gamma(\boldsymbol{v}) = v_n$. Then $\mathcal{A}$ is non-zero just in case there is a configuration $\boldsymbol{v} \in \mathbb{N}^n$ such that $v_n > 0$ and $\boldsymbol{v}$ is reachable from $\boldsymbol{\alpha}$ by a sequence of transitions in which all intermediate configurations consist of non-negative vectors. From this observation it is clear that the Coverability Problem for reset VAS reduces to the Non-Zeroness Problem for extended polynomial automata.

The Zeroness Problem for extended polynomial automata reduces to the Zeroness Problem for ordinary polynomial automata.

Proposition 1. *The Zeroness Problem for extended polynomial automata reduces in polynomial time to the Zeroness Problem for ordinary polynomial automata.*

Proof. Let $\mathcal{A} = (\Sigma, n, \xi, \boldsymbol{\alpha}, p, \gamma)$ be an extended polynomial automaton. We define an ordinary polynomial automaton $\mathcal{B} = (\Sigma, n+1, \boldsymbol{\alpha}', p', \gamma')$ such that $[\![\mathcal{A}]\!]$ is identically zero if and only if $[\![\mathcal{B}]\!]$ is identically zero. The idea behind the definition of $\mathcal{B}$ is to introduce an extra variable y. This variable is set to 0 whenever an invalid configuration is encountered, and this forces the output to be 0.

whenever ξ(x) is 0 at any time, this will propagate

We define:

$$\begin{aligned}\boldsymbol{\alpha}' &:= (\boldsymbol{\alpha}, 1)\\ p'_\sigma(\boldsymbol{x}, y) &:= (p_\sigma(\boldsymbol{x}), y\xi(\boldsymbol{x})) \quad (\sigma \in \Sigma)\\ \gamma'(\boldsymbol{x}, y) &:= y\gamma(\boldsymbol{x})\,.\end{aligned}$$

□

We can use Example 3 to show that the Zeroness Problem for polynomial automata is computationally difficult:

Theorem 1. *The Zeroness Problem for polynomial automata is hard for the complexity class* ACKERMANN.

Proof. The Coverability Problem for reset vector addition systems is well-known to be ACKERMANN-hard [5], [23]. Now Example 3 shows that this problem reduces in polynomial time to the Non-Zeroness Problem for extended polynomial automata. Then Proposition 1 in turn shows that the Coverability Problem for reset VAS reduces in polynomial time to the Non-Zeroness Problem for ordinary polynomial automata. □

However, extended and ordinary polynomial automata differ when we consider the Equivalence Problem. We will see later on that the Equivalence Problem for ordinary polynomial automata is decidable. In contrast, undecidability of the Equivalence Problem for extended polynomial automata follows from undecidability of trace equivalence for Petri Nets, a variant of VAS with a labelled transitions [24].

V. Zeroness is Ackermannian

In this section we show that the Zeroness Problem for polynomial automata lies in the complexity class ACKERMANN, matching the lower bound from Theorem 1.

Let $\mathcal{A} = (\Sigma, n, \boldsymbol{\alpha}, p, \gamma)$ be a polynomial automaton. We define an increasing sequence of ideals

$$I_0 \subseteq I_1 \subseteq I_2 \subseteq \dots \tag{1}$$

in $\mathbb{Q}[x_1, \dots, x_n]$ by

$$I_k = \langle \gamma \circ p_w : w \in \Sigma^{\leq k} \rangle\,.$$

all outputs of reachable configurations

By the Hilbert Basis Theorem, such a sequence eventually stabilises. Moreover we have:

Proposition 2. *Let $k_* \in \mathbb{N}$ be such that $I_{k_*} = I_{k_*+1}$. Then*

(i) *$I_k = I_{k_*}$ for all $k \geq k_*$, and*
(ii) *$[\![\mathcal{A}]\!]$ is identically zero if and only if $\boldsymbol{\alpha} \in \mathbf{V}(I_{k_*})$.*

Proof. Item (i) follows immediately from the observation that the sequence of ideals $\{I_k : k \in \mathbb{N}\}$ satisfies the inductive relation

infinite!

$$I_{k+1} = \langle f \circ p_a : f \in I_k, a \in \Sigma \cup \{\varepsilon\} \rangle$$

generators

for all $k \in \mathbb{N}$.

For (ii), by construction of the I_k we have that $[\![\mathcal{A}]\!]$ is identically zero iff $\boldsymbol{\alpha} \in \mathbf{V}(I_k)$ for all k. Thus $[\![\mathcal{A}]\!]$ is identically zero iff $\boldsymbol{\alpha} \in \mathbf{V}(I_{k_*})$ by (i). □

membership test in the variety

We will analyse the stabilisation of the chain of ideals (1) in terms of the associated sequence of downward-closed sets in $\mathbb{N}^n$

$$D_0 \supseteq D_1 \supseteq D_2 \supseteq \dots\,, \tag{2}$$

defined by

$$D_k := \{\boldsymbol{\alpha} \in \mathbb{N}^n : \boldsymbol{x}^{\boldsymbol{\alpha}} \notin \underset{\text{leading term (monomial)}}{\mathrm{LT}(I_k)}\}$$

for all $k \in \mathbb{N}$. The key fact is:

Proposition 3. *For all $k \in \mathbb{N}$, $D_k = D_{k+1}$ if and only if $I_k = I_{k+1}$.*

Proof. The $\overset{\Leftarrow}{\underset{\Rightarrow}{\text{“if”}}}$ direction is immediate from the definition of D_k. Conversely suppose $D_k = D_{k+1}$. Then $\mathrm{LT}(I_k) = \mathrm{LT}(I_{k+1})$ and hence every Gröbner basis of I_k is also a Gröbner basis of I_{k+1}. It follows that $I_k = I_{k+1}$. □ !

Sequence (2) stabilises in finitely many steps by Dickson's Lemma and hence the sequence of ideals (1) eventually stabilises: this is essentially the argument underlying Hilbert's Basis Theorem (see, e.g., [17, Chapter 2, Section 5]). However Dickson's Lemma does not yield quantitative bounds on the stabilisation of (2). To obtain such bounds, in place of Dickson's Lemma we use results of Lazić and Schmitz [5] on the length of "controlled" descending sequences of downwards-closed sets in $\mathbb{N}^n$.

Recall from Section III-2 that for downward closed $D \subseteq \mathbb{N}^n$, $\|D\|$ denotes the largest natural number present in the representation of D as a finite union of order ideals. For a monotone function $g : \mathbb{N} \to \mathbb{N}$ and $m \in \mathbb{N}$, the sequence $(D_k)_{k \geq 0}$ is said to be *(g, m)-controlled* if for every index $k \in \mathbb{N}$, $\|D_k\| \leq g^k(m)$.

Theorem 2 (Length function theorem [5])**.** *Any (g, m)-controlled descending chain of downwards-closed subsets of $\mathbb{N}^n$ is of length at most $h_{\omega^{n+1}}(m \cdot n!)$, where $h(x) \overset{\text{def}}{=} n \cdot g(x)$ and h_α denotes the αth Cichoń function [25] with base function h and ordinal α.*

For a polynomial automaton $\mathcal{A}$, let us assume that all polynomials involved have total degree at most d. Each ideal I_k is generated by polynomials of the form $\gamma \circ p_w$ where $|w| \leq k$, and these must have degree at most d^{k+1}. The following result of Dubé [26] can be used to give a bound on the degree of the polynomials in a Gröbner basis for I_k.

Theorem 3. *Let $I \subseteq \mathbb{Q}[x_1, \dots, x_n]$ be an ideal generated by polynomials of degree at most m. Then there is a Gröbner basis for I comprising polynomials of degree at most $2(m+1)^{2^n}$.* 2-exp

It follows from Theorem 3 that each ideal I_k has a Gröbner basis consisting of polynomials of degree at most $2(d+1)^{(k+1)2^n}$. We now use this fact to obtain a similar bound on $\|D_k\|$.

Let $\{g_1, \dots, g_\ell\}$ be a reduced Gröbner basis for I_k, with $\mathrm{LT}(g_i) = c_i \boldsymbol{x}^{\boldsymbol{\beta_i}}$. Then we have (applying the distributive law in the third line)

$$\begin{aligned}D_k &= \Big\{\boldsymbol{\alpha} \in \mathbb{N}^n : \bigwedge_{i=1}^{\ell} \boldsymbol{\beta_i} \nleq \boldsymbol{\alpha}\Big\}\\ &= \Big\{\boldsymbol{\alpha} \in \mathbb{N}^n : \bigwedge_{i=1}^{\ell} \bigvee_{j=1}^{n} \beta_{i,j} > \alpha_j\Big\}\\ &= \bigcup_{f:[\ell]\to[n]} C_f\,,\end{aligned}$$

where $[\ell] = \{1,\dots,\ell\}$, $[n] = \{1,\dots,n\}$, and

$$C_f = \left\{ \boldsymbol{\alpha} \in \mathbb{N}^n : \bigwedge_{i=1}^{\ell} \beta_{i,f(i)} > \alpha_{f(i)} \right\}.$$

Thus we can express D_k as a union of order ideals C_f, as above. It is moreover clear that for any f, $\|C_f\|$ is at most the maximum of the entries $\beta_{i,j}$, for $i = 1,\dots,l$ and $j = 1,\dots,n$, and is thus at most $2(d+1)^{(k+1)2^n}$. It follows that $\|D_k\| = \max_f \|C_f\|$ is at most this quantity. With some algebraic manipulation, we obtain the following:

Proposition 4. *The sequence $(D_k)_{k\geq 0}$ of downwards-closed sets induced by an extended polynomial automaton $\mathcal{A}$ of degree bound d and dimension n is (g,m)-controlled, where $m = 2(d+1)^{2^n}$ and $g(x) = mx = 2x(d+1)^{2^n}$.*

Using Theorem 2, the length of the sequence $(D_k)_{k\geq 0}$ is bounded by the Cichoń function $h_{\omega^{n+1}}(mn!)$ where $h(x) = mg(x) = mnx$. Such a function is bounded by $F_\omega(p(n))$, where p is primitive recursive and F_ω is the Ackermann function as defined in Section III-5 (see [27, Fact 2.30] or [18] for more detail). By Proposition 2, this gives us a bound on the length of a potential witness word that must be considered to demonstrate Non-Zeroness. By testing all words up to this length bound we have:

Theorem 4. *The Zeroness Problem for polynomial automata (and hence also extended polynomial automata) is in the complexity class* ACKERMANN.

In Section IV we presented a polynomial-time reduction from Equivalence to Zeroness for polynomial automata. Combining this with Theorem 4 we get:

Corollary 1. *The Equivalence Problem for polynomial automata is in the complexity class* ACKERMANN.

VI. Invertible Polynomial Automata

A. Definition and Examples

In this section we present a subclass of polynomial automata for which the Non-Zeroness Problem is primitive recursive. The condition that enables this improved complexity bound for the Zeroness Problem is invertibility of the transition function. More formally, we consider the following condition on an extended polynomial automaton $\mathcal{A} = (\Sigma, n, \xi, \boldsymbol{\alpha}, p, \gamma)$:

(I) Each transition function p_σ has a rational inverse that is defined on the set of legitimate states, that is, there exists a rational map $q_\sigma : \mathbb{Q}^n \to \mathbb{Q}^n$, defined on all points at which the domain map ξ is non-zero, such that $q_\sigma(p_\sigma(\boldsymbol{v})) = \boldsymbol{v}$ for all $\boldsymbol{v} \in \mathbb{Q}^n$ for which $q_\sigma(p_\sigma(\boldsymbol{v}))$ is defined and $p_\sigma(q_\sigma(\boldsymbol{v})) = \boldsymbol{v}$ for all $\boldsymbol{v} \in \mathbb{Q}^n$ for which $p_\sigma(q_\sigma(\boldsymbol{v}))$ is defined.

We say that an extended polynomial automaton satisfying Condition (I) is *invertible*.

Example 4. *The class of polynomial automata $\mathcal{A} = (\Sigma, n, \xi, \boldsymbol{\alpha}, p, \gamma)$ used in Example 3 to simulate reset VAS is not invertible in general. Specifically, such an automaton fails to be invertible if for some $\sigma \in \Sigma$ the transition map p_σ has the zero polynomial as a component. In this case σ represents a "reset transition" (A, a, b), where the matrix A has a zero entry on its diagonal. If such reset transitions are omitted, then we obtain a class of invertible polynomial automata corresponding to the case of ordinary VAS. More generally, in Subsection VI-D we describe a subclass of affine VAS whose coverability problem reduces in polynomial time to the Non-Zeroness Problem for invertible polynomial automata.*

Notice that in Condition (I) the inverse map q_σ need only be defined when the domain map ξ is non-zero. We point the reader forward to Example 6 to illustrate the utility of this condition.

B. Strong Monotonicity

In this section we associate a decreasing sequence of varieties $\{V_k : k \in \mathbb{N}\}$ with a polynomial automaton. Such a sequence corresponds to the increasing chain of ideals studied in Section V (modulo the fact that we deal directly with extended polynomial automata in this section). We use the condition of invertibility of polynomial automata to prove a strengthened version of the monotonicity condition on the varieties V_k. From this we obtain a primitive recursive stabilisation bound on the sequence of varieties. In this section, *all varieties are defined over the field $\mathbb{A}$ of algebraic numbers*. We will still be computing only over the rationals, but we will need algebraic closedness of the ambient field to compute the bounds in Proposition 7.

Define a decreasing sequence of varieties

$$V_0 \supseteq V_1 \supseteq V_2 \supseteq \dots \tag{3}$$

in $\mathbb{A}^n$ by

$$V_k = \bigcap_{w \in \Sigma^{\leq k}} \left[\mathbf{V}(\gamma \circ p_w) \cup \bigcup_{u \leq w} \mathbf{V}(\xi \circ p_u) \right], \tag{4}$$

where $u \leq w$ denotes that u is a prefix of w.

The descending chain of varieties (3) is eventually constant by the Hilbert Basis Theorem. Moreover we have:

Proposition 5. *Let $k_* \in \mathbb{N}$ be such that $V_{k_*} = V_{k_*+1}$. Then*

(i) *$V_k = V_{k_*}$ for all $k \geq k_*$, and*

(ii) *$[[\mathcal{A}]]$ is identically zero if and only if $\boldsymbol{\alpha} \in V_{k_*}$.*

Proof. Item (i) follows from the observation that the sequence of varieties $\{V_k : k \in \mathbb{N}\}$ satisfies the inductive relation

$$V_{k+1} = V_k \cap \bigcap_{\sigma \in \Sigma} \left[p_\sigma^{-1}(V_k) \cup \mathbf{V}(\xi) \right] \tag{5}$$

for all $k \in \mathbb{N}$.

For (ii), by construction of the V_k we have that $[[\mathcal{A}]]$ is identically zero iff $\boldsymbol{\alpha} \in V_k$ for all k. Thus $[[\mathcal{A}]]$ is identically zero iff $\boldsymbol{\alpha} \in V_{k_*}$ by (i). □

Say that an irreducible subvariety W of V_k is *proper* in V_k if $W \nsubseteq V_{k+1}$. We say that the sequence $\{V_k : k \in \mathbb{N}\}$ is *strongly monotone* if for all $k \in \mathbb{N}$, whenever some subvariety W of V_{k+1} is proper in V_{k+1} then there exists a subvariety W' of V_k that is proper in V_k and is such that $\dim(W') \geq \dim(W)$.

The notion of strongly monotone chains of varieties was introduced by Novikov and Yakovenko [13]. Lazić and

Schmitz [5] define the closely related notion of *ω-monotone* chains of lower sets of $\mathbb{N}^n$ (with order ideals playing the role of irreducible varieties). Lazić and Schmitz apply their notion to obtain a bound on termination of the backward algorithm for solving the Coverability Problem for VAS. We will apply the notion of strong monotonicity to the more general setting of invertible affine VAS.

First we show that for an invertible polynomial automaton the sequence of varieties (3) is strongly monotone.

Proposition 6. *Suppose that $\mathcal{A} = (\Sigma, n, \xi, \boldsymbol{\alpha}, p, \gamma)$ is invertible. Then the sequence of varieties (3) is strongly monotone.*

Proof. Suppose that W is an irreducible variety such that $W \subseteq V_k$ and $W \not\subseteq V_{k+1}$ for some $k \in \mathbb{N}$. Considering the expression (5), since $W \subseteq V_k$ and $W \not\subseteq V_{k+1}$ it must be true that $W \not\subseteq \mathbf{V}(\xi)$ and there exists $\sigma \in \Sigma$ such that $p_\sigma(W) \not\subseteq V_k$. Moreover, since $W \subseteq V_k$ and $W \not\subseteq \mathbf{V}(\xi)$, then we must have $p_\sigma(W) \subseteq V_{k-1}$.

Define W' to be the Zariski closure of $p_\sigma(W)$, that is, W' is the smallest variety containing $p_\sigma(W)$. Then W' is a subvariety of V_{k-1}, but $W' \not\subseteq V_k$. We can now show that W' remains irreducible, referring to the definition in Section III-3. Suppose that $W' \subseteq W_1' \cup W_2'$ for varieties W_1', W_2'. Then, $W \subseteq p_\sigma^{-1}(W_1') \cup p_\sigma^{-1}(W_2')$. Hence $W \subseteq p_\sigma^{-1}(W_i')$ for some $i \in \{1,2\}$, as W is irreducible and the pre-image of a variety under a polynomial remains a variety. Taking images in p_σ, we see that $p_\sigma(W) \subseteq W_i'$; hence by minimality of the Zariski closure we have $W' \subseteq W_i'$. Thus W' is irreducible.

To establish strong monotonicity it remains to show that $\dim(W') \geq \dim(W)$. Now since $W' \subseteq V_{k-1}$, and $W' \not\subseteq V_k$, it must be true that $W' \not\subseteq \mathbf{V}(\xi)$. Thus the pair $p_\sigma : W \dashrightarrow W'$ and $q_\sigma : W' \dashrightarrow W$ is a birational equivalence between W and W'. Thus $\dim(W') = \dim(W)$. □

C. Primitive Recursive Bound

In this section we recall the argument of [13] bounding the stabilisation of strongly monotone chains of varieties under certain degree bounds on the defining polynomials. From this we derive a primitive recursive bound on the complexity of the Zeroness Problem for invertible polynomial automata.

Given $b \in \mathbb{N}$, define $b \Uparrow 0 = 1$ and $b \Uparrow (k+1) = b^{b \Uparrow k}$ for all $k \geq 0$. Then $b \Uparrow k$ is a tower of b's of height k.

Proposition 7 ([13])**.** *Let $\{V_k : k \in \mathbb{N}\}$ be a strongly monotone sequence of varieties in $\mathbb{A}^n$. Suppose moreover that there is an integer constant c such that for all $k \geq 1$ the variety V_k is defined by a finite collection of polynomials of degree at most c^k. Then the sequence $\{V_k\}$ stabilises after at most $(1+c^n) \Uparrow (n+1)$ steps.*

Proof. For $j \in \{0,\dots,n+1\}$, let k_j be the least index $k \geq 1$ such that V_k has no proper subvariety of dimension $\geq n-j+1$. Then $k_0 = 1$ and the sequence $\{V_k : k \in \mathbb{N}\}$ is stable at index k_{n+1}.

Fix $j \in \{0,\dots,n\}$. For all $k \in \{k_j,\dots,k_{j+1}-1\}$, V_k contains a proper subvariety W of dimension exactly $n-j$. We claim that W is a maximal irreducible subvariety of V_{k_j}. Indeed if W were strictly included in some irreducible subvariety W' of V_{k_j} then we would have $\dim(W') > n-j$ by [17, Prop. 10, Chapter 9.4]; but then neither W' nor W could be proper.

As noted in Section III, the total number of irreducible components of V_{k_j} that have dimension at least $n-j$ is at most $(c^{k_j})^j$. Thus we have

$$\begin{aligned} k_{j+1} &\leq k_j + (c^{k_j})^j \\ &= k_j + (c^j)^{k_j} \\ &\leq (1+c^n)^{k_j} . \end{aligned}$$

Iterating this inequality, we have $k_{n+1} \leq (1+c^n) \Uparrow (n+1)$. □

Theorem 5. *The Non-Zeroness Problem for the class of invertible polynomial automata is primitive recursive.*

Proof. Let $\mathcal{A}$ be an invertible polynomial automaton. Suppose that all polynomials involved in the definition of $\mathcal{A}$ have total degree at most d. For the associated sequence of varieties $\{V_k : k \in \mathbb{N}\}$, defined in (4), for all $k \in \mathbb{N}$ we have

$$V_k = \mathbf{V}\left(\left\{(\gamma \circ p_w) \cdot \prod_{u \leq w} \xi \circ p_u : w \in \Sigma^{\leq k}\right\}\right).$$

Thus V_k is the set of common zeros of a collection of polynomials of degree at most $(k+2)d^{k+1}$.

Applying Proposition 7 with $c = 3(d+1)^2$, we deduce that if $\mathcal{A}$ is non-zero then there exists a word w of length at most $(1+3^n(d+1)^{2n}) \Uparrow (n+1)$ such that $[\![\mathcal{A}]\!](w) \neq 0$. Thus a primitive recursive algorithm for checking non-zeroness is to evaluate $\mathcal{A}$ on all words up to this length bound. □

D. Affine VAS and Invertibility

We now look at the case of affine VAS in more detail.

Let us say that an affine VAS $\mathcal{R}$ is *invertible* if for each transition rule $(A, \boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}$ the matrix A has an inverse (possibly with rational entries). For example, such a VAS could permute the values of registers, could add the contents of one register to another, or could multiply the contents of a register by a non-zero constant. We show that the Zeroness Problem for polynomial automata corresponding to invertible VAS is well-behaved, and hence the Coverability Problem for invertible VAS is well-behaved as well.

Theorem 6. *The Coverability Problem for invertible affine VAS is in* 2EXPSPACE.

Proof. We begin by giving a class of polynomial automata capturing general affine VAS.

Consider the following three conditions on a polynomial automaton $\mathcal{A} = (\Sigma, n, \xi, \boldsymbol{\alpha}, p, \gamma)$:

A1 the initial configuration $\boldsymbol{\alpha} \in \mathbb{Q}^n$ is a vector of nonnegative integers,

A2 the domain function is

$$\xi(x_1,\dots,x_n) = (x_1+1)(x_2+1)\dots(x_n+1).$$

A3 for each $\sigma \in \Sigma$ there is a matrix A of nonnegative integers and vectors $\boldsymbol{a}, \boldsymbol{b}$ with entries in $\{-1,0,1\}$ such that $p_\sigma(\boldsymbol{x}) = A(\boldsymbol{x}-\boldsymbol{a}) + \boldsymbol{b}$.

Following the same reasoning as Example 3, the Coverability Problem for affine VAS reduces to the Non-Zeroness Problem for polynomial automata satisfying A1, A2, and A3: given an affine VAS $\mathcal{R}$, the associated polynomial automaton has a transition function $p(\boldsymbol{x}) = A(\boldsymbol{x}-\boldsymbol{a})+\boldsymbol{b}$ for each rule $(A, \boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}$.

Let $\mathcal{R}$ be an invertible affine VAS and let $\mathcal{A} = (\Sigma, n, \xi, \boldsymbol{\alpha}, p, \gamma)$ be the associated polynomial automaton satisfying A1, A2, and A3. Then $\mathcal{A}$ is an invertible polynomial automaton: the inverse of the transition function $p_\sigma(\boldsymbol{x}) = A(\boldsymbol{x}-\boldsymbol{a})+\boldsymbol{b}$ is $q_\sigma(\boldsymbol{x}) = A^{-1}(\boldsymbol{x}-\boldsymbol{b})+\boldsymbol{a}$.

Now let $\{V_k : k \in \mathbb{N}\}$ be the decreasing sequence of ideals associated with $\mathcal{A}$. Recall that $V_k = \mathbf{V}(I_k)$ where I_k is generated by the polynomials

$$(\gamma \circ p_w) \cdot \prod_{u \le w} \xi \circ p_u$$

for $w \in \Sigma^{\le k}$. But the maps p_u are polynomials of degree 1 and ξ is a polynomial of degree n. It follows that for all $k \ge 1$, V_k is generated by polynomials of degree at most $(n+1)(k+1)$.

Now, following the proof of Proposition 7, we can define a sequence of indices $k_0 \le k_1 \le \ldots k_n$, where k_j is the least index $k \ge 1$ such that V_k has no proper subvariety of dimension $\ge n-j+1$. Then the sequence $\{V_k : k \in \mathbb{N}\}$ is stable at index k_{n+1} and the indices k_j satisfy the recurrence:

$$\begin{aligned} k_{j+1} &\le k_j + ((n+1)(k_j+1))^j \\ &\le (1+(n+1)(k_j+1))^j . \end{aligned}$$

From this we get a bound on the stabilisation point k_{n+1} that is doubly exponential in n and thus we obtain a double-exponential bound on a witness word for Non-Zeroness. Now guessing such a word a single letter at a time gives a non-deterministic 2EXPSPACE algorithm (each application of a transition function increases the bit-size of the configuration vector by a constant), which can be converted to a deterministic 2EXPSPACE algorithm by Savitch's theorem. □

We conclude by noting that already for ordinary VAS, the shortest seqeuence of transitions leading from an initial configuration to some configuration covering a given place may have length doubly exponential in the dimension of the VAS [28]. The extra exponential in Theorem 6, compared to the well-known EXPSPACE-completeness of Coverability for ordinary VAS, simply comes from the extra space required to represent configurations.

VII. Streaming String Transducers

A streaming string transducer is a finite-state one-way string-to-string transducer that is equipped with registers storing string values. Registers can be combined and updated along a computation, and the output of the transducer over a particular run is a function of the final register values. The subclass of so-called *copyless* streaming string transducers expresses precisely the class of MSO-definable string-to-string transductions [7], and is thus equivalent in expressiveness to the class of deterministic two-way string transducers [29].

A. Basic Definitions

We follow the notation used by Filiot and Reynier in [8].

Definition 1. *A streaming string transducer (SST) T consists of the following components:*

- *finite input alphabet Σ and output alphabet Γ,*
- *finite set of states Q, with distinguished start state $q_0 \in Q$ and set of final (or accepting) states $Q_f \subseteq Q$,*
- *transition function $\delta : Q \times \Sigma \to Q$,*
- *finite set of register variables $\mathcal{X} = \{X_1, \ldots, X_m\}$,*
- *register update function $\rho : Q \times \Sigma \to S_{\mathcal{X},\Gamma}$, where $S_{\mathcal{X},\Gamma} := \{\sigma : \mathcal{X} \to (\Gamma \cup \mathcal{X})^*\}$ is a set of substitutions,*
- *initial register valuation a ground substitution $\sigma_0 \in S_{\mathcal{X},\Gamma}$ (i.e., such that $\sigma_0(X) \in \Gamma^*$ for all $X \in \mathcal{X}$), and*
- *output function $\theta : Q_f \to (\Gamma \cup \mathcal{X})^*$, defined on the set of accepting states.*

A substitution $\sigma : \mathcal{X} \to (\Gamma \cup \mathcal{X})^*$ can be extended freely to a homomorphism $\widehat{\sigma} : (\Gamma \cup \mathcal{X})^* \to (\Gamma \cup \mathcal{X})^*$, which acts on a string by replacing each variable in $\mathcal{X}$ with its image under σ and leaving symbols in Γ unchanged. We then define composition of substitutions σ_1 and σ_2 by writing $(\sigma_2 \circ \sigma_1)(X) := \widehat{\sigma}_2(\sigma_1(X))$ for all $X \in \mathcal{X}$.

Intuitively when an SST T is in state $q \in Q$ and reads input $a \in \Sigma$ then it makes a transition to state $\delta(q,a)$ and updates its registers by making a simultaneous assignment $X := \rho(q,a)(X)$ for all $X \in \mathcal{X}$. Formally, the (uniquely defined) run of T on a word $w = w_1 \ldots w_n \in \Sigma^*$ is the sequence

$$(q_0, \sigma_0) \xrightarrow{w_1} (q_1, \sigma_1) \xrightarrow{w_2} \ldots \xrightarrow{w_n} (q_n, \sigma_n), \qquad (6)$$

where $(q_i, \sigma_i) \in Q \times S_{\mathcal{X},\Gamma}$ for $i = 0, \ldots, n$, q_0 is the initial state, σ_0 is the initial register valuation, and $q_i = \delta(q_{i-1}, w_i)$ and $\sigma_i = \sigma_{i-1} \circ \rho(q_{i-1}, w_i)$ for $i = 1, \ldots, n$. Note that each substitution σ_i in (6) is a ground substitution.

An SST T induces a partial function $[\![T]\!]$ from Σ^* to Γ^* whose domain $\mathrm{dom}([\![T]\!])$ is the language of the finite automaton underlying T. Given a word $w \in \Sigma^*$ such that the unique run of T on w ends in the configuration (q, σ), we define

$$[\![T]\!](w) = \begin{cases} \sigma(\theta(q)) & \text{if } q \text{ is accepting} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We say that two SSTs T and T' (over common input and output alphabets) are equivalent if the partial functions $[\![T]\!]$ and $[\![T']\!]$ are identical. The *Equivalence Problem* for SSTs is to determine whether two given SSTs T and T' are equivalent.

Example 5. *Let Σ be a finite alphabet and $\$ \notin \Sigma$ a special symbol. Denote by* rev *the string reversal function on Σ^*. We define a single-state SST, with input and output alphabets both equal to $\Sigma \cup \{\$\}$, that computes a transduction mapping each string $u_1\$u_2\$\ldots\$u_n$, where $u_1, \ldots, u_n \in \Sigma^*$, to* $\mathrm{rev}(u_1)\$\mathrm{rev}(u_1u_2)\$\ldots\$\mathrm{rev}(u_1u_2\ldots u_n)$.

There are two register variables X and Y, with initial register valuation $\sigma_0(X) = \sigma_0(Y) = \varepsilon$. Being in the single-

state case, we elide the state arguments of the register-update map ρ and output map θ. The register update map is given by

$$\begin{array}{rcl} \rho(a)(X) & = & aX \\ \rho(a)(Y) & = & Y \end{array} \quad (a \in \Sigma) \qquad \begin{array}{rcl} \rho(\$)(X) & = & X \\ \rho(\$)(Y) & = & YX\$ \end{array}$$

and output map by $\theta = YX$.

B. Simulating Single-state SSTs with Polynomial Automata

The goal of this section is to reduce the Equivalence Problem for single-state SSTs to the Equivalence Problem for polynomial automata. After that, we will show that the Equivalence Problem for general SSTs can be reduced to that of single-state SSTs.

We give an equivalence-preserving translation of single-state SSTs over some fixed output alphabet Γ to polynomial automata. The key to this translation is the existence of an injective monoid homomorphism $\Phi : \Gamma^* \to M_2(\mathbb{Q})$, where $M_2(\mathbb{Q})$ is the monoid of 2×2 rational matrices. Indeed, we obtain an injective homomorphism $\Psi : \{0,1\}^* \to M_2(\mathbb{Q})$ by writing

$$\Psi(0) := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \Psi(1) := \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

and extending freely to $\{0,1\}^*$ [30]. Now for a finite alphabet $\Gamma = \{a_1, \ldots, a_k\}$ we obtain an injective homomorphism $\Phi : \Gamma^* \to M_2(\mathbb{Q})$ by writing

$$\Phi(a_i) = \Psi(\underbrace{0 \ldots 0}_{i}\,1) \quad \text{for } i = 1, \ldots, k$$

The map Φ gives a numerical encoding of strings in Γ^* under which string concatenation corresponds to matrix multiplication. Now matrix multiplication is a polynomial map $M_2(\mathbb{Q}) \times M_2(\mathbb{Q}) \to M_2(\mathbb{Q})$, given by

$$\left(\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}, \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix} \right) \mapsto \begin{pmatrix} x_{11}y_{11} + x_{12}y_{21} & x_{11}y_{12} + x_{12}y_{22} \\ x_{21}y_{11} + x_{22}y_{21} & x_{21}y_{12} + x_{22}y_{22} \end{pmatrix}$$

Given a single-state SST T with input alphabet Σ and output alphabet Γ we will define a polynomial automaton $\mathcal{A}$ over alphabet Σ with output in $\mathbb{Q}^4$ such that $[\![\mathcal{A}]\!](w) = \Phi([\![T]\!](w))$ for all $w \in \Sigma^*$. Before giving a formal definition of the automaton $\mathcal{A}$ we give an example of a polynomial automaton corresponding to the SST T in Example 5.

Example 6. *We define a polynomial automaton $\mathcal{A} = (\Sigma, n, \boldsymbol{\alpha}, p, \gamma)$ over alphabet $\Sigma = \{0, 1, \$\}$ with output in $\mathbb{Q}^4$. The dimension is $n = 8$; we represent configurations of $\mathcal{A}$ as pairs of 2×2 matrices and we represent the output of $\mathcal{A}$ as a single 2×2 matrix. Thus we define*

$$\begin{array}{rcll} \boldsymbol{\alpha} & = & (I, I) & \\ p_a(X,Y) & = & (\Phi(a)X, Y) & a \in \{0,1\} \\ p_\$(X,Y) & = & (X, YX\Phi(\$)) & \\ \gamma(X,Y) & = & YX\,, & \end{array}$$

where I denotes the 2×2 identity matrix and

$$X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \quad \textit{and} \quad Y = \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \end{pmatrix} \tag{7}$$

are matrices of indeterminates. From the fact that Φ is a homomorphism one easily checks that $[\![\mathcal{A}]\!](w) = \Phi([\![T]\!](w))$ for any word $w \in \Sigma^*$.

We remark in passing that if we augment automaton $\mathcal{A}$ with a domain map $\xi(X,Y) = \det(X)$, making it an extended polynomial automaton, then each transition map p_σ has a rational inverse that is is defined whenever $\xi(X,Y) \neq 0$. Thus $\mathcal{A}$ becomes an invertible polynomial automaton.

We now give the formal definition of the polynomial automaton $\mathcal{A}$ corresponding to a single-state SST T. Assume that T has set of register variables $\mathcal{X} = \{X_1, \ldots, X_m\}$, initial register valuation $\sigma_0 : \mathcal{X} \to \Gamma^*$, register update map $\rho : \Sigma \to S_{\mathcal{X},\Gamma}$, and output string $\theta \in (\Gamma \cup \mathcal{X})^*$. We identify the set of ground substitutions $\sigma : \mathcal{X} \to \Gamma^*$ with $(\Gamma^*)^m$ by representing σ as the tuple $(\sigma(X_1), \ldots, \sigma(X_m)) \in (\Gamma^*)^m$. Under this identification we define maps,

$$\begin{array}{l} \mathrm{eval}_\theta : (\Gamma^*)^m \to \Gamma \\ \mathrm{comp}_{\rho_a} : (\Gamma^*)^m \to (\Gamma^*)^m \quad (a \in \Sigma) \end{array}$$

by $\mathrm{eval}_\theta(\sigma) = \sigma(\theta)$ and $\mathrm{comp}_{\rho_a}(\sigma) = \sigma \circ \rho_a$.

Define $\mathcal{A} = (\Sigma, n, \boldsymbol{\alpha}, p, \gamma)$, where the dimension is $n = 4m$. In describing the remaining structure of $\mathcal{A}$ we will identify $\mathbb{Q}^{4m}$ with $M_2(\mathbb{Q})^m$. The initial vector is $\boldsymbol{\alpha} = (\Phi(\sigma_0(X_1)), \ldots, \Phi(\sigma_0(X_m)) \in M_2(\mathbb{Q})^m$. The transition function $p : \Sigma \times M_2(\mathbb{Q})^m \to M_2(\mathbb{Q})$ and output function $\gamma : M_2(\mathbb{Q})^m \to M_2(\mathbb{Q})$ are chosen to be polynomial maps such that the diagrams below commute. Such polynomial maps exist since the monoid multiplication in $M_2(\mathbb{Q})$ is a polynomial map.

$$\begin{tikzcd} (\Gamma^*)^m \arrow[r, "\mathrm{eval}_\theta"] \arrow[d, "\Phi^m"] & (\Gamma^*) \arrow[d, "\Phi"] \\ M_2(\mathbb{Q})^m \arrow[r, "\gamma"] & M_2(\mathbb{Q}) \end{tikzcd} \qquad \begin{tikzcd} (\Gamma^*)^m \arrow[r, "\mathrm{comp}_{\rho_a}"] \arrow[d, "\Phi^m"] & (\Gamma^*)^m \arrow[d, "\Phi^m"] \\ M_2(\mathbb{Q})^m \arrow[r, "p_a"] & M_2(\mathbb{Q})^m \end{tikzcd} \tag{8}$$

This completes the definition of $\mathcal{A}$. It remains to show that $\Phi([\![T]\!](w)) = [\![\mathcal{A}]\!](w)$ for all $w \in \Sigma^*$. To this end, we have:

$$\begin{array}{rcll} \Phi([\![T]\!](w)) & = & \Phi((\sigma_0 \circ \rho_{w_1} \circ \ldots \circ \rho_{w_n})(\theta)) & \\ & = & (\Phi \cdot \mathrm{eval}_\theta)(\sigma_0 \circ \rho_{w_1} \circ \ldots \circ \rho_{w_n}) & \\ & = & (\Phi \cdot \mathrm{eval}_\theta \cdot \mathrm{comp}_{\rho_{w_n}} \cdot \ldots \cdot \mathrm{comp}_{\rho_{w_1}})(\sigma_0) & \\ & = & (\gamma \cdot \Phi^m \cdot \mathrm{comp}_{\rho_{w_n}} \cdot \ldots \cdot \mathrm{comp}_{\rho_{w_1}})(\sigma_0) & (9) \\ & = & (\gamma \cdot p_{w_n} \cdot \ldots \cdot p_{w_1} \cdot \Phi^m)(\sigma_0) & (10) \\ & = & (\gamma \cdot p_{w_n} \cdot \ldots \cdot p_{w_1})(\boldsymbol{\alpha}) & \\ & = & [\![\mathcal{A}]\!](w)\,. & \end{array}$$

Line (9) follows by commutativity of the left-hand diagram in (8), while line (10) follows by (repeated application of) commutativity of the right-hand diagram in (8).

Theorem 7. *The Equivalence Problem for single-state SSTs can be reduced in logarithmic space to the Equivalence Problem for polynomial automata.*

Proof. Let T and T' be single-state SSTs over the same input alphabet Σ and with common output alphabet $\Gamma = \{0,1\}$. Let $\mathcal{A}$ and $\mathcal{A}'$ be polynomial automata, as defined above, such that $\Phi(\llbracket T \rrbracket(w)) = f_{\mathcal{A}}(w)$ and $\Phi(\llbracket T' \rrbracket(w)) = f_{\mathcal{A}'}(w)$ for all $w \in \Sigma^*$. Since Φ is injective we have that T and T' are equivalent if and only if $\mathcal{A}$ and $\mathcal{A}'$ are equivalent. □

C. From General SSTs to Single-State SSTs

We now show how to reduce the equivalence problem for general SSTs to that for single-state SSTs, thereby obtaining a decision procedure for equivalence of general SSTs.

Proposition 8. *The Equivalence Problem for SSTs can be reduced in logarithmic space to the Equivalence Problem for single-state SSTs.*

Proof. We define an equivalence-respecting transformation of an arbitrary SST T to a single-state SST T'. This transformation is such that $\llbracket T' \rrbracket(w) = \varepsilon$ for all $w \notin \mathrm{dom}(\llbracket T \rrbracket)$ and $\llbracket T' \rrbracket(w) = \$ \cdot \llbracket T \rrbracket(w)$ for all $w \in \mathrm{dom}(\llbracket T \rrbracket)$, where $\$ \in \Gamma$ is a distinguished output symbol. We give the transformation in three steps.

Let T be an SST as in Definition 1. As a first step we make T "total" by declaring every state to be accepting, extending the output function to map hitherto non-accepting states to the empty string, and prefixing all existing output strings by \$ to make them non-empty.

Secondly, we obtain an equivalent SST in which the register update and output functions ρ and θ are "constant-free", that is, both functions take values exclusively in $\mathcal{X}^*$. To do this we introduce a new register variable X for every string constant $u \in \Gamma^*$ appearing in the definitions of ρ and θ. We alter ρ and θ by replacing each occurrence of u with X, we stipulate that $\rho(q,a)(X) = X$ for all $q \in Q$ and $a \in \Sigma$, and we extend the initial register valuation by defining $\sigma_0(X) = u$.

Now let T be total, with constant-free update and output functions. The final step is to construct an equivalent single-state SST T'. Suppose that T has set of variables $\mathcal{X} = \{X_1, \ldots, X_m\}$ and set of states Q. The set of variables of T' is $\mathcal{X}' = \{X_{i,q} : q \in Q, 1 \le i \le m\}$. The idea is that that a configuration (p, σ) of T is encoded by a register valuation σ' of T' such that for $i = 1, \ldots, m$ we have

$$\sigma'(X_{i,q}) := \begin{cases} \sigma(X_i) & \text{if } q = p \\ \varepsilon & \text{if } q \neq p. \end{cases} \tag{11}$$

The initial register valuation σ_0' of T' encodes the initial configuration (q_0, σ_0) of T, following (11). Specifically define $\sigma_0'(X_{i,q_0}) := \sigma_0(X_i)$ and $\sigma_0'(X_{i,q}) := \varepsilon$ if $q \neq q_0$.

Let ρ' denote the register-update function of T'. Since T' has a single state, we simply write $\rho'(a)$ for the register update when reading input symbol $a \in \Sigma$. To define $\rho'(a)$ we introduce a renaming map $\iota_q : (\Gamma \cup \mathcal{X})^* \to (\Gamma \cup \mathcal{X}')^*$, for each $q \in Q$, defined by $\iota_q(X_i) = X_{i,q}$. Then we write

$$\rho'(a)(X_{i,p}) = \prod_{q:\delta(q,a)=p} \iota_q(\rho(a,q)(X_i)) .$$

The product here denotes string concatenation. The precise order of this concatenation does not matter since only one of the factors is non-null.

Finally, the output string $\theta' \in (\mathcal{X}')^*$ of T' is obtained as the following concatenation, where, again, the order of the product is immaterial:

$$\theta' := \prod_{q \in Q_f} \iota_q(\theta(q)) .$$

□

Composing these logarithmic space reductions, we see that:

Corollary 2. *The Equivalence Problem for SSTs can be reduced in logarithmic space to the Equivalence Problem for polynomial automata.*

This immediately gives decidability:

Corollary 3. *The Equivalence Problem for SSTs is in the complexity class* ACKERMANN.

Decidability of the equivalence problem was announced in [8], but to date no complexity bounds have appeared.

VIII. Copyless Polynomial Automata and SSTs

In this section we identify a subclass of polynomial automata for which the Zeroness and Equivalence problems are solvable in PSPACE. We call this subclass *copyless* by analogy with the class of copyless SSTs. In fact it will turn out that the translation of SSTs to polynomial automata given in Section VII-B restricts to a translation of copyless SSTs to copyless polynomial automata. We thus obtain a new proof that equivalence of copyless SSTs is decidable in PSPACE—a result that was first shown in [6].

We first recall the definition of copyless SSTs [8]. Let T be an SST, as in Definition 1, with set of variables $\mathcal{X}$ and output alphabet Γ. We say that a substitution $\sigma \in S_{\mathcal{X},\Gamma}$ is *multilinear* if for every $X \in \mathcal{X}$, each variable in $\mathcal{X}$ occurs at most once in $\sigma(X)$. We say that σ is *disjoint* if for every pair of distinct variables $X, Y \in \mathcal{X}$, no variable from $\mathcal{X}$ occurs in both $\sigma(X)$ and $\sigma(Y)$. We say that T is *copyless* if all the register update maps $\rho(q,a)$ are multilinear and disjoint and if each output string $\theta(q)$ mentions each variable in $\mathcal{X}$ at most once.

Example 7. *The SST is Example 5 is not copyless. However if we change the definition of the register update map by stipulating that $\rho(\$)(X) = \varepsilon$ then we do get a copyless automaton. This automaton computes the transduction that maps $u_1\$u_2\$\ldots\$u_n$ to $\mathrm{rev}(u_1)\$\mathrm{rev}(u_2)\$\ldots\$\mathrm{rev}(u_n)$ for $u_1, \ldots, u_n \in \Sigma^*$.*

Next we give the analogous definition of the class of copyless polynomial automata. Let $\mathcal{A} = (\Sigma, n, \boldsymbol{\alpha}, \delta, \gamma)$ be a polynomial automaton whose transition map and output function are defined by polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$. We say that $\mathcal{A}$ is *copyless* if there is a partition of the set of variables $\{x_1, \ldots, x_n\}$ into components $X_1, \ldots, X_k$ such that the following two conditions hold:

- **multilinearity.** The output polynomial γ and update polynomials $\delta(\sigma, -)_i$, where $\sigma \in \Sigma$ and $i \in \{1, \ldots, n\}$, have degree at most one in each of the components

$X_1, \ldots, X_k$.[2] In particular, in all mentioned polynomials each variable has degree at most one.

- **disjointness.** If variables x_i and x_j belong to different components then for each $\sigma \in \Sigma$ and each component X_l, $1 \leq l \leq k$, at least one of $\delta(\sigma,-)_i$ and $\delta(\sigma,-)_j$ has degree zero in X_l.

Observe that linear polynomial automata are copyless with respect to the trivial partition on the set of variables (that is, with a single component). In this case the multilinearity condition just expresses linearity, and disjointness holds vacuously.

Another source of copyless polynomial automata (motivating the terminology) is copyless SSTs:

Example 8. *Following the terminology of Example 6, the polynomial automaton arising from the copyless SST in Example 7 has the form $\mathcal{A} = (\Sigma, n, \boldsymbol{\alpha}, p, \gamma)$, where $n = 8$ and*

$$\begin{array}{rcll} \boldsymbol{\alpha} & = & (I, I) & \\ p_a(X,Y) & = & (\Phi(a)X, Y) & a \in \{0,1\} \\ p_\$(X,Y) & = & (I, YX\Phi(\$)) & \\ \gamma(X,Y) & = & YX\,. & \end{array}$$

This automaton is copyless with respect to the partition of the set of variables into two components $X = \{x_1, \ldots, x_4\}$ and $Y = \{y_1, \ldots, y_4\}$.

More generally, if T is a copyless SST then the corresponding single-state SST T', obtained in Proposition 8, is also copyless. In turn, the polynomial automaton $\mathcal{A}$ that is constructed from T' by the reduction in Theorem 7 is copyless.

Below we give an exponential transformation of copyless polynomial automata into equivalent linear automata. To give some intuition, we look to the polynomial automaton from Example 8. The transition and output functions of this automaton are both bilinear functions in the variables $X = \{x_1, \ldots, x_4\}$ and $Y = \{y_1, \ldots, y_4\}$. To obtain an equivalent linear automaton the idea is to introduce a family of variables $Z = \{z_{i,j} : 1 \leq i, j \leq 4\}$, where $z_{i,j}$ represents the product $x_i y_j$, and to rewrite the transition and output functions as linear functions in the variables Z.

Proposition 9. *Given a copyless polynomial automaton $\mathcal{A}$ there is a linear automaton $\mathcal{B}$ such that $f_{\mathcal{A}} = f_{\mathcal{B}}$.*

Proof. Let $\mathcal{A} = (\Sigma, n, \boldsymbol{\alpha}, \delta, \gamma)$ be a copyless polynomial automaton with respect to a partition of the set of variables $\{x_1, \ldots, x_n\}$ into components $X_1, \ldots, X_k$. We define an equivalent linear automaton $\mathcal{B} = (\Sigma, m, \boldsymbol{\alpha}', \delta', \gamma')$ as follows:

- We introduce a variable $y_{i_1,\ldots,i_s}$ corresponding to each monomial $x_{i_1} x_{i_2} \ldots x_{i_s}$ such that $i_1 < i_2 < \ldots < i_s$ and the x_{i_j} all lie in different components.
- We define the initial value of $y_{i_1,\ldots,i_s}$ to be $\boldsymbol{\alpha}'_{i_1,\ldots,i_s} := \alpha_{i_1} \ldots \alpha_{i_s}$.
- Fixing $\sigma \in \Sigma$ and a variable $y_{i_1,\ldots,i_s}$ of $\mathcal{B}$. In order to define the polynomial $\delta'(\sigma,-)_{i_1,\ldots,i_s}$, notice that by the multilinearity and disjointness conditions for $\mathcal{A}$, the product $\delta(\sigma,-)_{i_1} \ldots \delta(\sigma,-)_{i_s}$ has degree at most one in each of the sets $X_1, \ldots, X_k$. We obtain $\delta'(\sigma,-)_{i_1,\ldots,i_s}$ by replacing every monomial $x_{j_1} \ldots x_{j_t}$ in this product with the single variable $y_{j_1,\ldots,j_t}$.
- We likewise obtain the output polynomial γ' of $\mathcal{B}$ by replacing each monomial $x_{i_1} \ldots x_{i_s}$ in γ with the single variable $y_{i_1,\ldots,i_s}$.

Then automaton $\mathcal{B}$ is linear by construction.

Given a word $w = w_1 \ldots w_m \in \Sigma^*$, suppose that $\mathcal{A}$ has computation $u^{(0)}, \ldots, u^{(m)}$ on w and $\mathcal{B}$ has computation $v^{(0)}, \ldots, v^{(m)}$ on w. We claim that for each tuple $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ and for $0 \leq j \leq m$,

$$v^{(j)}(y_{i_1,\ldots,i_k}) = u^{(j)}(x_{i_1}) \cdots u^{(j)}(x_{i_k})\,.$$

The proof of the claim is by induction on j. □

Theorem 8. *The Zeroness and Equivalence Problems for copyless polynomial automata are solvable in* PSPACE.

Proof. Proposition 9 gives an exponential translation of copyless polynomial automata into equivalent linear automata. This transformation requires polynomial working space. Now the Zeroness and Equivalence problems for linear automata (which are weighted automata over a field) are solvable in polylogarithmic space [21]. Thus in combination we obtain a polynomial-space procedure for deciding Zeroness and Equivalence of copyless polynomial automata. □

Corollary 4. *The Equivalence Problem for copyless streaming string transducers is solvable in* PSPACE.

Proof. The equivalence-respecting transformation of SSTs to single-state SSTs, given in the proof of Proposition 8, specialises to a transformation from copyless SSTs to single-state copyless SSTs. Likewise the equivalence-respecting transformation of single-state SSTs to polynomial automata, given in the proof of Theorem 7, maps copyless SSTs to copyless polynomial automata. Combining the two reductions we obtain a logarithmic space reduction of the Equivalence Problem of copyless SSTs to that of copyless polynomial automata. □

IX. Conclusions

Polynomial automata give a common generalisation of weighted automata over a field and vector addition systems. The Zeroness Problem for polynomial automata generalises the Coverability Problem for VAS, while the Equivalence Problem for polynomial automata generalises the equivalence problem for weighted automata, as studied in [22]. We provide a combination of algebraic and combinatorial conditions that suffice to give elementary bounds on the Zeroness Problem, and show that these include some classes of automata of interest. We also show how analysis of the Equivalence Problem for polynomial automata can yield results for problems that look quite different on the surface, e.g., equivalence of transducers.

We leave open the question of matching upper and lower complexity bounds for both invertible polynomial automata and invertible affine VAS. Similarly we leave open the question of tight bounds for our SST equivalence problems.

[2] Let X be a set of variables and $Y \subseteq X$. Clearly a polynomial $f \in \mathbb{Z}[X]$ can be regarded as a polynomial $\widetilde{f}$ in set of variables Y with coefficients in $\mathbb{Z}[X \setminus Y]$. By the degree of f in the set Y we mean the total degree of $\widetilde{f}$.

Deciding equivalence of automata is a key component of Angluin's active learning framework [31]. Existing work has studied the learnability of deterministic, non-deterministic, and alternating finite automata [32], [33] and weighted automata [34], [35] in this framework, based on variants of the Myhill–Nerode Theorem [36], [37] and extensions to weighted automata [38], [39]. In ongoing work, based on [16], we are exploring the extension of these results on learnability to the setting of polynomial automata. Already, by combining the translation of copyless SSTs to copyless polynomial automata in Section VIII with the learning algorithm for weighted automata over a field in [34], we obtain a learning algorithm for SSTs in the active learning framework.

References

[1] E. Sontag and Y. Rouchaleau, "On discrete-time polynomial systems," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 1, no. 1, pp. 55–64, 1976.

[2] O. Maler, "Algorithmic verification of continuous and hybrid systems," in *INFINITY*, 2013.

[3] M. Müller-Olm and H. Seidl, "Computing polynomial program invariants," *Inf. Process. Lett.*, vol. 91, no. 5, pp. 233–244, 2004.

[4] M.-P. Schützenberger, "On the Definition of a Family of Automata," *Information and Control*, vol. 4, pp. 245–270, 1961.

[5] R. Lazić and S. Schmitz, "The Ideal View on Rackoff's Coverability Technique," in *RP*, 2015.

[6] R. Alur and P. Černý, "Streaming transducers for algorithmic verification of single-pass list-processing programs," in *POPL*, 2011.

[7] ——, "Expressiveness of streaming string transducers," *FSTTCS*, 2010.

[8] E. Filiot and P. Reynier, "On streaming string transducers and HDT0L systems," *CoRR*, vol. abs/1412.0537, 2014. [Online]. Available: http://arxiv.org/abs/1412.0537

[9] A. Finkel, S. Göller, and C. Haase, "Reachability in register machines with polynomial updates," in *Proceedings of Mathematical Foundations of Computer, 38th International Symposium, MFCS*, ser. Lecture Notes in Computer Science, vol. 8087. Springer, 2013, pp. 409–420.

[10] C. Dufourd, A. Finkel, and P. Schnoebelen, *Reset nets between decidability and undecidability*, 1998.

[11] C. Dufourd and A. Finkel, "Polynomial-Time Many-One Reductions for Petri Nets," in *FSTTCS*, 1997.

[12] G. M. Socias, "Length of polynomial ascending chains and primitive recursiveness." *MATHEMATICA SCANDINAVICA*, vol. 71, no. 0, pp. 181–205, 1992.

[13] D. Novikov and S. Yakovenko, "Trajectories of polynomial vector fields and ascending chains of polynomial ideals," *Ann. Inst. Fourier*, vol. 49, no. 2, 1999.

[14] ——, "Meandering of trajectories of polynomial vector fields in the affine n-space," *Publicacions Matemàtiques*, vol. 41, no. 1, pp. 223–242, 1997.

[15] T. Duff, "Beyond Weighted Automata: The Complexity of Equivalence and Minimization," Master's thesis, University of Oxford, 2015.

[16] A. Sharad, "Learning and Consistency for Weighted Automata," Master's thesis, University of Oxford, 2016.

[17] D. A. Cox, J. B. Little, and D. O'Shea, *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*, 2nd ed. Springer-Verlag, 1997.

[18] S. Schmitz, "Complexity hierarchies beyond elementary," *ACM Trans. Comput. Theory*, vol. 8, no. 1, pp. 3:1–3:36, 2016.

[19] J. Berstel and C. Reutenauer, *Rational series and their languages*, ser. EATCS monographs on theoretical computer science. Berlin, New York: Springer-Verlag, 1988.

[20] J. Sakorevitch, "Rational and recognisable power series," in *Handbook of Weighted Automata*, 1st ed., M. Droste, W. Kuich, and H. Vogler, Eds. Springer, 2009.

[21] W.-G. Tzeng, "On path equivalence of nondeterministic finite automata," *Inf. Process. Lett.*, vol. 58, no. 1, pp. 43–46, 1996.

[22] ——, "A polynomial-time algorithm for the equivalence of probabilistic automata," *SIAM Journal on Computing*, vol. 21, no. 2, pp. 216–227, 1992.

[23] P. Schnoebelen, "Revisiting Ackermann-Hardness for Lossy Counter Machines and Reset Petri Nets," in *MFCS*, 2010.

[24] M. Hack, *Decidability Questions for Petri Nets*, ser. Outstanding Dissertations in the Computer Sciences, 1975.

[25] S. Schmitz, "Complexity bounds for ordinal-based termination," in *RP*, 2014.

[26] T. Dubé, "The structure of polynomial ideals and Gröbner bases," *SIAM Journal on Computing*, vol. 19, no. 4, pp. 750–773, 1990.

[27] S. Schmitz and P. Schnoebelen, "Algorithmic aspects of wqo theory. lecture notes," http://cel.archives-ouvertes.fr/cel-00727025.

[28] R. J. Lipton, "The reachability problem requires exponential space," Dep. Comput. Sci., Yale University, Tech. Rep. 62, 1976.

[29] J. Engelfriet and H. J. Hoogeboom, "MSO definable string transductions and two-way finite-state transducers," *TOCL*, vol. 2, no. 2, pp. 216–254, 2001.

[30] V. S. Guba, "Equivalence of infinite systems of equations in free groups and semigroups to finite subsystems," *Mathematical notes of the Academy of Sciences of the USSR*, vol. 40, no. 3, pp. 688–690, 1986.

[31] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and Computation*, vol. 75, no. 2, pp. 87–106, 1987.

[32] B. Bollig, P. Habermehl, C. Kern, and M. Leucker, "Angluin-style Learning of NFA," in *IJCAI*, 2009.

[33] D. Angluin, S. Eisenstat, and D. Fisman, "Learning regular languages via alternating automata," in *IJCAI*, 2015.

[34] A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio, "Learning functions represented as multiplicity automata," *Journal of the ACM*, vol. 47, no. 3, pp. 506–530, 2000.

[35] W.-G. Tzeng, "Learning Probabilistic Automata and Markov Chains via Queries," *Machine Learning*, vol. 8, no. 2, pp. 151–166, 1992.

[36] J. Myhill, "Finite automata and the representation of events," Wright Air Development Division, Tech. Rep. 57-264, 1957.

[37] A. Nerode, "Linear Automaton Transformations," *Proceedings of the American Mathematical Society*, vol. 9, no. 4, pp. 541–544, 1958.

[38] J. W. Carlyle and A. Paz, "Realizations by Stochastic Finite Automata," *Journal of Computer and System Sciences*, vol. 5, pp. 26–40, 1971.

[39] M. Fliess, "Matrices de Hankel," *Journal de Mathématiques Pures et Appliquées*, vol. 53, 1974.