Journal of Algebra 359 (2012) 138–155

Contents lists available at SciVerse ScienceDirect

Journal of Algebra

www.elsevier.com/locate/jalgebra

Polynomial bounds for invariant functions separating orbits

Harlan Kadish [1]

Department of Mathematics, Mailstop 3368, Texas A&M University, College Station, TX, 77843-3368, USA

ARTICLE INFO

Article history:
Received 12 July 2011
Available online 27 March 2012
Communicated by Jon Carlson

Keywords:
Algebraic group
Representation
Action
Orbit
Straight line program

ABSTRACT

In a representation of a linear algebraic group G, polynomial invariant functions almost always fail to separate orbits. Unless G is reductive, the ring of invariant polynomials may not be finitely generated. Also the number and complexity of the generators may grow rapidly with the size of the representation. We instead consider an extension of the polynomial ring by introducing a "quasi-inverse" that computes the inverse of a function where defined. With the addition of the quasi-inverse, we write straight line programs defining functions that separate the orbits of any linear algebraic group G. The number of these programs and their length have polynomial bounds in the parameters of the representation.

1. Introduction

1.1. Background

When a linear algebraic group G acts on an affine variety V over a field k, the orbit of $x \in V$ is the set

$$G \cdot x = \{g \cdot x \mid g \in G\}.$$

Applications of invariant theory, such as computer vision, dynamical systems, and structural chemistry, demand constructive and efficient techniques to distinguish the orbits of a group action. Much recent work to distinguish orbits has employed functions in the invariant ring,

$$k[V]^G = \left\{ f \in k[V] \;\middle|\; f(g \cdot x) = f(x)\ \forall g \in G \right\}.$$

E-mail address: hmkadish@math.tamu.edu.

[1] Supported by DMS-0502170, Enhancing the Mathematical Sciences Workforce in the 21st Century (EMSW21) Research Training Group (RTG): Enhancing the Research Workforce in Algebraic Geometry and Its Boundaries in the Twenty-First Century.

0021-8693/$ – see front matter
http://dx.doi.org/10.1016/j.jalgebra.2012.03.008

In fact, for any linear algebraic group G, the invariant ring contains a finitely generated subalgebra S with the following property: Let $p, q \in V$ have disjoint orbits, and suppose there exists $f \in k[V]^G$ such that $f(p) \neq f(q)$. Then there exists $h \in S$ such that $h(p) \neq h(q)$ (see [1]). We say that the function h (and the algebra S) *separates* the orbits of p and q. Hence the functions in S, called *separating invariant polynomials*, separate as many orbits as do the polynomials in $k[V]^G$. Note that $\overline{G \cdot p} = \overline{G \cdot q}$ implies $G \cdot p = G \cdot q$, because the orbits of a linear algebraic group are open in their closures.

A subalgebra S of separating invariant polynomials has several weaknesses. For one, existence proofs for S may not be constructive for all groups G. One can classify a linear algebraic group by its *radical* $R(G)$: its largest connected normal solvable subgroup. Kemper's 2003 algorithm [2] computes a separating subalgebra $S \subseteq k[V]^G$ but assumes that G is reductive, that is, that the unipotent part of $R(G)$ is trivial. Now, generating sets for $k[V]^G$ do have polynomial degree bounds for semisimple groups (where $R(G)$ is trivial) [3], tori [4], and linearly reductive groups (whose representations decompose uniquely into irreducibles) [5]. By contrast, for general G, algorithms to compute orbit-separating, invariant polynomials do not have good complexity bounds. Nor do they have good bounds on the size of a separating subset or the degrees of its elements. Kemper's algorithm [2], for example, requires two Gröbner basis calculations, a normalization algorithm, and an inseparable closure algorithm. Some facts are known in special cases. If V is a representation of G, Domokos in 2007 showed there exists a separating set S for the action on V^m such that each $f \in S$ involves variables from no more than $2\dim(V)$ of the copies of V [6]. When G is finite, Kemper in 2009 provided a bound [7], polynomial in $|G|$, on the size of a separating set.

As a more serious limitation, the invariant ring $k[V]^G$ will usually fail to separate orbit closures. So any subalgebra $S \subseteq k[V]^G$ will usually fail as well.

Example 1.1. Let the multiplicative group $G = k^*$ act on $V = k^2$ by scaling: for $g \in k^*$, define $g \cdot (a, b) = (ga, gb)$. On the one hand, there are infinitely many orbits: the origin and, for each $(v_1, v_2) \in k^2 - (0, 0)$, the sets $\{(tv_1, tv_2) \mid t \in k^*\}$ of punctured lines through the origin. Thus every orbit closure includes the origin. The invariant polynomials, which are continuous functions under the Zariski topology, will then have the same value on each orbit as at the origin. Hence the ring of invariant polynomials is $k[x, y]^G = k$, the constants, and the functions here can detect none of the infinitely many orbits. This phenomenon occurs in any representation where two points p, q have $\overline{G \cdot p} \cap \overline{G \cdot q} \neq \emptyset$. Thus separating orbits with invariant functions requires functions sensitive to locally closed sets.

The goal of this paper is to compute invariant, orbit-separating *functions* $V \to k$ without restrictions on the algebraic group. It is worth noting that the orbit-membership *decision problem* can already be solved in polynomial time. The orbit-membership decision problem asks, "Given, $x, y \in V$, does there exist $g \in G$ such that $gx = y$?" One solution employs an "effective nullstellensatz" algorithm that checks if ideals are proper, as follows. For a representation $V = k^n$ of G, choose $k[G] = k[z_1, \ldots, z_\ell]/\mathbb{I}(G)$ for some ideal $\mathbb{I}(G)$, and consider a representation $\rho : G \to GL(V)$. Note that two points $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in V$ lie in the same G-orbit if and only if the ideal J in $k[G]$ generated by the expressions $\sum_j \rho(z_1, \ldots, z_\ell)_{ij} \cdot x_j - y_i, 1 \leqslant i \leqslant n$, is a proper ideal. Giusti, Heintz, and Sabia in 1993 [8] produced a randomized algorithm that can check the properness of J in sequential time polynomial in n but exponential in ℓ.

One can also solve the orbit-membership decision problem via quantifier elimination. Choose generators $h_1, \ldots, h_s$ for the ideal $\mathbb{I}(G)$ defining G in $\mathbb{A}^\ell$. For $1 \leqslant i \leqslant n$, let $F_i(x, y, z) \in k[X \times X \times G]$ be the polynomial $F_i = \sum_j \rho(z_1, \ldots, z_\ell)_{ij} \cdot x_j - y_i$, and let $F_{n+j} = h_j$ among the chosen generators of $\mathbb{I}(G)$. Then x and y lie in the same orbit if and only if they satisfy the formula

$$\mathcal{P}(x, y) = \exists z_1 \cdots \exists z_\ell \big(F_1(x, y, z_1, \ldots, z_\ell) = 0 \wedge \cdots \wedge F_{n+s}(x, y, z_1, \ldots, z_\ell) = 0\big).$$

In 1998, Puddu and Sabia produced a well-parallelizable algorithm [9] without divisions that can eliminate the block of quantifiers in $\mathcal{P}$ with sequential complexity polynomial in the number of equations and their degree, but exponential in ℓ. That is, their algorithm finds a formula equivalent to $\mathcal{P}$ but

only involving the vanishing and nonvanishing of polynomials in the coordinates of x and y, over an algebraically closed field. For an approach over the integers $\mathbb{Z}$ or $\mathbb{Z}/p\mathbb{Z}$, see the 1983 paper [10] of Heintz.

1.2. Separating orbits with the quasi-inverse

Alternatively, the new algorithm below produces invariant functions $V \to k$ that separate orbits in the style of classical invariant theory, without restrictions on the linear algebraic group. To overcome the limitations of the polynomial functions on V, though, we introduce the *quasi-inverse* $f^\star$ of a regular function $f \in k[V]$:

$$f^\star(p) = \begin{cases} 1/f(p) & f(p) \neq 0, \\ 0 & f(p) = 0. \end{cases}$$

Define the a ring $\widehat{k[V]}$ of functions on V formally as follows. Putting $R = k[V]$, let $\widehat{R^1}$ denote the ring generated by the set $R \cup \{f^\star \mid f \in R\}$. Of course, relations arise among the generators of $\widehat{R^1}$: for example, the definition of the quasi-inverse on V implies that for every $f \in R$, one has $f^2 f^\star = f$ and $f(f^\star)^2 = f^\star$. Continuing in this way, define $\widehat{R^i}$ to be the ring generated by the set $\widehat{R^{i-1}} \cup \{f^\star \mid f \in \widehat{R^{i-1}}\}$. Note that for every $i \leqslant j$ there are inclusions $\widehat{R^i} \hookrightarrow \widehat{R^j}$. In conclusion, define the ring $\widehat{R}$ of functions on V as the direct limit $\widehat{R} = \varinjlim \widehat{R^i}$. Since each $f \in \widehat{R}$ lies in some R^i, we can compute f via a straight line program with inputs in R, that is, with a finite list of sums, products, and quasi-inverses of elements of R. We discuss straight line programs in Section 2.1 below. For now, note the following, for elements $f, h \in k[V]$:

- $1 - ff^\star \in \widehat{R^1}$ is a characteristic function for the set $\mathbb{V}(f) \subseteq V$ where f vanishes: i.e., $1 - ff^\star$ has the value 1 on $\mathbb{V}(f)$ and the value zero elsewhere.
- If $D(f)$ denotes the complement of $\mathbb{V}(f)$, then $ff^\star(1 - hh^\star) \in \widehat{R^1}$ is a characteristic function for the locally closed set $D(f) \cap \mathbb{V}(h) \subseteq V$.

So functions in $\widehat{R}$ can identify more complicated sets than can the polynomials. The definition of the quasi-inverse and the ring $\widehat{R}$ appear to be new. If R is an arbitrary commutative ring with 1, the thesis [11] of the author describes an alternative construction of $\widehat{R}$, its commutative algebra, and the structure of $\operatorname{Spec} \widehat{R}$.

For a representation of any fixed linear algebraic group G, we seek to compute a finite set $\mathcal{C} \subset \widehat{k[V]}$ of invariant functions that separate orbits. That is, if p, q lie in different orbits, then some function $f \in \mathcal{C}$ has $f(p) \neq f(q)$. Even better, we would like the evaluation of f at p to be reasonably simple. Now, $k[V]$ has a grading by degree, but $\widehat{k[V]}$ does not. So we measure the complexity, or length, of the f as straight line programs over $\widehat{k[V]}$, granting unit cost to all ring operations and the quasi-inverse. Of course, the evaluation of such f at $p \in V$ requires branching, but counting the operations needed to compute f serves as an analog of classical degree bounds for invariants.

Over an algebraically closed field k, fix an embedding of an m-dimensional linear algebraic group $G \hookrightarrow \mathbb{A}^\ell$, namely by choosing the ideal $\mathbb{I}(G) \subset k[z_1, \ldots, z_\ell]$ of polynomials vanishing on G. Let $k[V] = k[x_1, \ldots, x_n]$, let $\rho : G \hookrightarrow GL_n(k)$ be a representation, let r be the maximal dimension of an orbit, and let $N = \max\{\deg(\rho_{ij})\}$ be the degree of the representation.

Theorem 1.1. *Let G be any linear algebraic group and V a representation. There is an algorithm to produce a finite set $\mathcal{C} \subset \widehat{k[V]}$ of invariant functions with the following properties*:

1. *The set $\mathcal{C}$ separates orbits in V.*
2. *The size of $\mathcal{C}$ grows as $O(n^2 N^{2\ell(r+1)+3r+2})$.*
3. *The $f \in \mathcal{C}$ can be written as straight line programs, such that the sum of their lengths is $O(n^3 N^{3\ell(r+1)+4r+3})$.*

Corollary 1.1. *Let G be any linear algebraic group and V a representation. The ring of functions $\widehat{k[V]}$ is an extension of the ring $k[V]$ sufficient to produce separating, invariant functions with polynomial length as straight line programs.*

Example 1.2. Recall the action of the multiplicative group $G = k^*$ on $V = k^2$ via $g \cdot (a, b) = (ga, gb)$; the only invariant polynomials are the constants. Nevertheless, the algorithm of Theorem 1.1 produces a set $\mathcal{C}$ including, after some simplification, $f_1 = 1 - xx^\star$, $f_2 = -yx^\star$, and $f_3 = 1 - yy^\star(1 - xx^\star)$. Consider the chart below of the values of the f_i at various points, where $a, b \neq 0$:

	f_1	f_2	f_3
$(0,0)$	1	0	1
$(a,0)$	0	0	0
$(0,b)$	1	0	0
(a,b)	0	$-b/a$	1

Recall that the orbits of this action are the origin and, for each $(v_1, v_2) \in k^2 - (0,0)$, the sets $\{(tv_1, tv_2) \mid t \in k^*\}$. Thus f_1, f_2, f_3 are constant on orbits, i.e. invariant, and separate the orbits as well.

To see the main idea of the algorithm of Theorem 1.1, choose $p \in V = k^n$ and consider the orbit map $\sigma_p : G \to V$ defined by $\sigma_p : g \mapsto g \cdot p$. Note that $\overline{G \cdot p}$ is defined by the polynomials in the kernel of $\sigma_p^* : k[x_1, \ldots, x_n] \to k[G]$. In the choice of the ideal $\mathbb{I}(G) = (h_1, \ldots, h_s) \subset k[z_1, \ldots, z_\ell]$ of polynomials vanishing on G, let $M = \max\{\deg(h_i)\}$. Then recalling the parameters N, r, and ℓ above, there is a degree bound $N^r M^{\ell - m} = \delta(N)$, such that $\overline{G \cdot p}$ is defined by the vanishing of polynomials in $k[x_1, \ldots, x_n]$ up to degree $\delta(N)$; see Section 3 below. Choosing a basis for $k[V]$, we may compute $\ker \sigma_p^*$ in degrees up to $\delta(N)$ via Gaussian elimination. Now, the entries of the kernel vectors vary with p, but they cannot in general be written as regular functions of p. We may nevertheless write them as functions in $\widehat{k[V]}$; these functions form the set $\mathcal{C}$. The functions in $\mathcal{C}$ are invariant and separate orbits because the algorithm computes the reduced row echelon form of a matrix for σ_p^* up to degree $\delta(N)$: points $p, q \in V$ lie in the same G-orbit if and only if $\ker \sigma_p^* = \ker \sigma_q^*$ if and only if, upon fixing a basis of monomials for $k[V]$, matrices for σ_p^* and σ_q^* have the same reduced row echelon form.

The polynomial length of the functions in $\mathcal{C}$ arises (1) from the degree bound $\delta(N)$, which is polynomial in N, and (2) because the algorithm looks in $k[G]$ for algebraic relations among products of the $\sigma_p^*(x_i)$: thus the Hilbert function of $k[G]$ controls how many such products the algorithm must consider. As a result, this new algorithm has complexity polynomial in the parameters n and N of the representation and exponential in $\ell \cdot \dim G$. Recall that the effective nullstellensatz and quantifier elimination algorithms for the orbit membership problem had complexity at least exponential in ℓ. So the ability to write separating functions from the ring $\widehat{k[V]}$ comes at the cost of a factor of $\dim G$ in the exponent.

We proceed as follows. In Section 2, we first define straight line programs in k-algebras with a quasi-inverse. Next, given a matrix $A(p)$ whose entries are functions in $k[V]$, we provide straight line programs for the reduced row echelon form and kernel of $A(p)$, such that the entries of the kernel vectors will be functions of p in $\widehat{k[V]}$. (Recall that in practice, $A(p)$ will be a matrix for σ_p^*, and the entries of vectors in $\ker A(p)$ are the invariant, orbit-separating functions in $\widehat{k[V]}$ that we seek.) In Section 3 we prove the degree bound $\delta(N)$ for the defining polynomials of an orbit closure. In Section 4, we provide an algorithm that computes invariant, orbit-separating functions in $\widehat{k[V]}$. We show that these functions have polynomial length as straight line programs in $\widehat{k[V]}$, and we establish the polynomial bound for their number in terms of n and the degree N of the representation.

2. Matrix computations using the quasi-inverse

2.1. Straight line programs

Before describing functions for the kernel of a matrix, we introduce a scheme to measure the complexity of such functions. Indeed, the degree of a polynomial in $k[V]$ provides a measure of the polynomial's complexity, but there is no grading by degree in $\widehat{k[V]}$. Instead, we adapt the framework of straight line programs over a F-algebra, F any field. For a deeper, traditional treatment, see the book [12]. Let V be a vector space, and let $R = F[V]$. First define $A = (a_{-s}, \dots, a_{-1}) \in (\widehat{R})^s$ to be an input of length s. A *straight line program* Γ is a finite, ordered list of instructions $\Gamma = (\Gamma_0, \dots, \Gamma_{t-1})$. Each instruction Γ_i is of the form $(\star; j, k)$ or $(\star; j)$, where $\star$ is an operation (see below) and j, k are positive integers. The program Γ is executable on input A with output $(a_{-s}, \dots, a_t)$ if, for $i \geqslant 0$,

$$a_i = \begin{cases} a_{i-j} + a_{i-k} & \text{if } \Gamma_i = (+; j, k), \\ a_{i-j} - a_{i-k} & \text{if } \Gamma_i = (-; j, k), \\ a_{i-j} \cdot a_{i-k} & \text{if } \Gamma_i = (\times; j, k), \\ a_{i-j}^{\star} & \text{if } \Gamma_i = (\text{qi}; j), \\ c & \text{if } \Gamma_i = (\text{const}; c) \text{ for } c \in F, \\ a_{i-j} & \text{if } \Gamma_i = (\text{recall}; j) \end{cases} \qquad \text{where } j, k \leqslant i + s.$$

Thus a program is executable if and only if each instruction $\Gamma_i = (\star; j, k)$ or $\Gamma_i = (\star; j)$ references entries with $i - j$, $i - k \geqslant -s$. The length t of the list Γ is the *length* of the program, which measures its computational complexity. The "recall" instruction serves to collect relevant computations in the last entries of the output. The definition of a straight line program in [12] over a F-algebra does not include the "recall" instruction (which collects computations at the end of the output list) or the quasi-inverse "qi" operation. After including the "qi" operation, we can write straight line programs for functions in $\widehat{R}$.

Define the *order-d output* of Γ by $\text{Out}_d(\Gamma, A) = (a_{t-d}, \dots, a_{t-1}) \in (\widehat{R})^d$, where $t = |\Gamma|$. We omit the d where convenient. Write $\Gamma^{(2)} \circ \Gamma^{(1)}$ for the composition of two straight line programs, in which the input of $\Gamma^{(2)}$ is $\text{Out}_d(\Gamma^{(1)}, A)$ for some d depending on $\Gamma^{(2)}$. Then $\Gamma^{(2)} \circ \Gamma^{(1)}$ has input A, and we execute $\Gamma^{(2)} \circ \Gamma^{(1)}$ by concatenating the instruction lists.

2.2. The reduced row echelon form program

Let $A(p) = (a_{ij})$ be a matrix over $k[V]$, whose entries are functions of p. Then the reduced row echelon form (RREF) of $A(p)$ also varies with p. This section constructs formulas in $\widehat{k[V]}$ for the entries of a matrix $R_A(p)$ such that for any choice of $p \in V$, $R_A(p)$ is the RREF of $A(p)$. The lengths of these functions as straight line programs are cubic in the dimensions of A. It is then easy to read off the kernel of A from R_A. Ultimately, in Section 4, we will take $A(p)$ to be a matrix over $k[V]$ for the orbit map $\sigma_p^* : k[V] \to k[G]$ up to degree $\delta(N)$, as outlined above. The invariant, orbit-separating functions that we seek will be the entries of the basis vectors for $\ker A(p)$. Henceforth we sometimes write just A for $A(p)$ and just R_A for $R_A(p)$.

Now, Berkowitz in 1984 [13] and Mulmuley [14] in 1986 developed tools for well-parallelizable linear algebra without divisions. We eschew those tools and instead compute the entries of R_A from A by imitating Gaussian elimination. This approach preserves a transparent connection between the ideal in $k[V]$ of an orbit closure (which $\ker A$ will ultimately represent) and the desired orbit-separating functions in $\widehat{k[V]}$. More generally, this approach also demonstrates how functions in the ring $k[V]$ can encode algorithms in linear algebra and algebraic geometry.

Finally, we do not compute the traditional reduced row echelon form of A, but instead a modified version that assists in the identification of pivot entries. It becomes simpler to compute the kernel of A when we know the locations of the pivots of the RREF of A. To this end, define the *triangular reduced row echelon form* (tRREF) of an $m \times n$ matrix A to be the $n \times n$ matrix $R_A = (r_{ij})$ whose jth

row $\mathbf{r}_j$ is nonzero if and only if the RREF of A has a pivot in column j. In that case, $\mathbf{r}_j$ is the row of the RREF of A containing that pivot. For example,

$$\text{RREF(A)} = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{corresponds to} \quad \text{tRREF}(A) = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Note that the tRREF of A is a square matrix. When we ultimately compute the kernel of A, we will exploit the following fact: the (traditional) RREF of A has a pivot in column j if and only if $r_{jj} = 1$ in the tRREF.

Proposition 2.1. *Let V be a vector space over any field k, and let $A(p) = (a_{ij})$ be an $m \times n$ matrix with entries in $\widehat{k[V]}$. Then there exists a straight line program Γ^{tR} over $\widehat{k[V]}$ of length $O(mn^2 + n^3)$ such that $R_A(p) = \text{Out}_{n^2}(\Gamma^{tR}, (a_{ij}))$ is the tRREF of $A(p)$ for every choice of $p \in V$.*

Note that this proposition does not require k to be algebraically closed, but we impose this condition in Theorem 1.1 to perform the geometry in that theorem's proof.

To prove Proposition 2.1, the following algorithm recursively computes the tRREF of a matrix $A(p)$ via Gaussian elimination. Following the algorithm are formulas for functions in $\widehat{k[V]}$ that compute the result of each step of the algorithm. Concatenating the straight line programs for these functions produces the program Γ^{tR}.

Algorithm 2.1. INPUT: An $m \times n$ matrix $A(p) = (a_{ij})$ over $k[V]$.

OUTPUT: An $n \times n$ matrix $R_A(p)$ over $\widehat{k[V]}$ that is the tRREF of $A(p)$.

1. For $i = 2, \ldots, m$, exchange the first row of A with row i if $a_{i1} \neq 0$. Label the resulting matrix $B = (b_{ij})$. After these steps, either $b_{11} \neq 0$, or $b_{i1} = 0$ for all i.
2. Multiply b_{11} by $b_{11}^{\star}$, and multiply the rest of the first row of B by $(1 - b_{11}b_{11}^{\star} + b_{11}^{\star})$. This operation is equivalent to dividing the first row by b_{11} if $b_{11} \neq 0$. Label the resulting matrix $C = (c_{ij})$.
3. If $n = 1$, OUTPUT $R_A = (c_{11})$.
4. While $n \geqslant 2$,
   (a) For $i = 2, \ldots, m$, subtract $c_{i1} \cdot (c_{11}, \ldots, c_{1n})$ from row i. Label the resulting matrix $D = (d_{ij})$. Then $d_{i1} = 0$ for all $i \geqslant 2$.
   (b) Let $D' = (d_{ij})_{j \geqslant 2}$ and $D'' = (d_{ij})_{i,j \geqslant 2}$, illustrated below:

$$D = \begin{pmatrix} * & \\ 0 & D' \\ \vdots & \\ 0 & \end{pmatrix} = \begin{pmatrix} * & \cdots & * \\ 0 & & \\ \vdots & D'' & \\ 0 & & \end{pmatrix}.$$

   Let D_0'' be the $m \times (n-1)$ matrix formed by appending a row of zeros to the bottom of D''; then D' and D_0'' have the same dimensions.
   (c) Compute $E = (1 - d_{11}) \cdot D' + d_{11} \cdot D_0''$.
   (d) Recursively compute the tRREF of E; call it R_E, an $(n-1) \times (n-1)$ matrix.
   (e) Let R_A' be the $n \times n$ matrix below:

$$R_A' = \begin{pmatrix} d_{11} & \cdots & d_{1n} \\ 0 & R_E & \\ 0 & & \end{pmatrix}.$$

   (f) Let $\mathbf{r}_k$ be the kth row of $R_A' = (r_{ij})$. For $k = 2, \ldots, n$, subtract $d_{1k} \cdot \mathbf{r}_k$ from the first row of R_A'. OUTPUT the result of these reductions as R_A, which is the tRREF of A.

Proof of correctness of Algorithm 2.1. If A is an $m \times 1$ matrix, then Steps 1 through 3 of the algorithm produce the tRREF of A. Now let A be an $m \times n$ matrix, and proceed by induction on n. Steps 1 through 4(a) of the algorithm follow Gaussian elimination to clear $a_{21}, \ldots, a_{m1}$ if the first column of A contains an nonzero entry. The result is the matrix D. If the first column of D contains all zeros, then we should recursively compute the tRREF of the remaining columns, that is, of the submatrix D' defined in Step 4(b). If instead $d_{11} = 1$, then we should compute the tRREF only of rows $i \geqslant 2$ and columns $j \geqslant 2$, that is, of the submatrix D''.

Now, our goal is to compute the tRREF of A with straight line programs whose inputs are the entries of A. The programs must continue the elimination process whether D' or D'' is the correct submatrix for recursion. To meet this requirement, we must first append a row of zeros to the bottom of D'' in Step 4(b), so that the result D''_0 has the same dimensions as D'. Then we compute $E = (1 - d_{11}) \cdot D' + d_{11} \cdot D''_0$. Note that $E = D'$ precisely when d_{11} is zero. Otherwise, $E = D''_0$, as required. Thus in Step 4(d) we can recursively compute the tRREF R_E of the $m \times n - 1$ matrix E using straight line programs. By the induction hypothesis, R_E is an $(n-1) \times (n-1)$ matrix.

With R_E in tRREF, it remains to reduce the first row of the matrix R'_A defined in Step 4(e). In Step 4(f), we let $\mathbf{r}_k$ be the kth row of R'_A. Since R_E is in tRREF, we know that for $k \geqslant 2$, $\mathbf{r}_k \neq 0$ if and only if R_E has a pivot in column k, if and only if $r_{kk} = 1$, if and only if the tRREF of A should have a pivot in column k as well.

There are two cases. If $d_{11} \neq 0$, then there is a pivot in the first row of R_A, the tRREF of A. In this case, in Step 4(f) we subtract $d_{1k} \cdot \mathbf{r}_k$ from the first row of R'_A for all $k \geqslant 2$, to clear the entries of $\mathbf{r}_1$ in precisely the columns k where R_A has a pivot.

If $d_{11} = 0$, then the first row of R_A should be zero. Recall that when $d_{11} = 0$, in fact R_E is the tRREF of D', and the first row of D' is $(d_{12}, \ldots, d_{1n})$. Since we compute R_E by Gaussian elimination, the vector $(d_{12}, \ldots, d_{1n})$ is in the span of the rows of R_E. Therefore, subtracting $d_{1k} \cdot \mathbf{r}_k$, $k \geqslant 2$, from $\mathbf{r}_1$ yields that R_A will have all zeros in its first row as required. The induction is complete. □

Below are formulas that give rise to straight line programs for functions in $\widehat{k[V]}$ that compute each step of the above algorithm. That is, if the input to a step is the matrix (x_{ij}) over $\widehat{k[V]}$, then we provide formulas for functions $y_{ij} \in \widehat{k[V]}$ such that the matrix (y_{ij}) is the output of that step. Concatenating these programs provides the straight line program Γ^{tR}, and we verify its length.

Step 1: For $i = 2, \ldots, m$, the first step exchanges the first row of (a_{ij}) with the ith row if $a_{11} = 0$. Hence for an $m \times n$ input matrix X, this step requires $m - 1$ programs E_i such that $Y = \mathrm{Out}_{mn}(E_i, X)$ exchanges the first and ith rows if necessary. The following formulas describe the entries of $Y = (y_{ij})$:

$$
\begin{aligned}
y_{11} &= x_{11} + (1 - x_{11}x^{\star}_{11})x_{i1}, \\
y_{1j} &= x_{1j} + (1 - x_{11}x^{\star}_{11}) \cdot (x_{ij} - x_{1j}) \quad \text{for all } j > 1, \\
y_{i1} &= x_{i1} \cdot x_{11}x^{\star}_{11}, \\
y_{ij} &= x_{1j} + x_{11}x^{\star}_{11} \cdot (x_{ij} - x_{1j}) \quad \text{for all } j > 1, \\
y_{kj} &= x_{kj} \quad \text{for all } k \neq 1, i, \text{ and for all } j.
\end{aligned}
$$

For example, the straight line program for y_{11} in E_i takes inputs x_{11} in position -2 and x_{i1} in position -1, and then performs the following steps:

(0) (qi; 2),
(1) (×; 3, 1),
(2) (const; 1),
(3) (−; 1, 2),
(4) (×; 1, 5),
(5) (+; 7, 1).

The formulas for the other y_{ij} have similarly obvious representations as straight line programs. If we concatenate these programs within E_i, so that all the entries of Y appear in various (known!) positions in the output, then we can save the recall steps for the end, and we need only compute $x_{11}^\star$, $x_{11}x_{11}^\star$, $(1-x_{11}x_{11}^\star)$, and $(x_{ij}-x_{1j})$ once. With these efficiencies, the program E_i introduces one quasi-inverse, one call to k, $3n$ additions, and $2n$ multiplications. Thus the concatenation of $E_2,\dots,E_{m-1}$ requires $2n(m-1)$ multiplications, $3n(m-1)$ additions, $n-1$ constants from k, $n-1$ quasi-inverses, and mn recalls to collect the entries of Y in the last mn cells of the tape. Call this concatenation Γ^E; we will use it later to collect nonzero rows of a matrix.

Step 2: of the algorithm requires one quasi-inverse, one subtraction, one addition, n multiplications, and n recalls.

Step 4(a): is achieved with the following formulas on an $m\times n$ input matrix (x_{ij}):

$$y_{i1}=0 \quad \text{for all } i>1$$
$$y_{ij}=x_{ij}-x_{1j}\cdot x_{i1}\cdot x_{11}x_{11}^\star \quad \text{for all } i,j>1.$$

These programs require $(m-1)(n-1)$ additions, $(n-1)(m-1)$ multiplications, and mn recalls. Step (5) next requires one subtraction, $m(n-1)$ additions, $2m(n-1)$ multiplications, and $m(n-1)$ recalls.

Step 4(c): has formulas that are clear from the algorithm, requiring one constant call, one subtraction, $2mn$ multiplications, and mn additions.

Step 4(f): requires the following formula for r_{1j}, $j\geqslant 2$:

$$r_{1j}:=(1-r_{jj})\cdot\big(r_{ij}+(-r_{22}\cdot r_{12}r_{2,j}-r_{33}\cdot r_{13}r_{i3,j}-\cdots-r_{j-1,j-1}\cdot r_{1,j-1}r_{j-1,j})\big),$$

This formula sets $r_{1j}=0$ if there is a pivot in column j, that is, if $r_{jj}=1$. Otherwise, the formula subtracts from r_{1j} the effects of clearing columns $<j$. The reduction of r_{1j} requires one call to k, j additions/subtractions, $2(j-2)+1$ multiplications (since $j\geqslant 2$), and n^2 recalls, so reducing the first row has total complexity $O(n^2)$.

Given an $m\times n$ matrix, Algorithm 2.1 performs the above computations n times, since each recursion handles a matrix with one fewer column. Reviewing the counts of operations in each step on an $m\times t$ matrix X, we see that the fastest growing quantities are mt and t^2, for $t=1,\dots,n$. So for large t, the number of operations is $O(mt+t^2)$ for each recursion. Recalling the formulas

$$\sum_{t=1}^{n}t=\frac{1}{2}n(n+1) \quad\text{and}\quad \sum_{t=1}^{n}t^2=\frac{1}{6}n(n+1)(2n+2),$$

we estimate that for a large m and n, the total length of Γ^{tR} is $O(mn^2+n^3)$.

Example 2.1. Recall the action of the multiplicative group $G=k^*$ on $V=k^2$ via $g\cdot(a,b)=(ga,gb)$. We can choose $k[G]=k[t,\frac{1}{t}]\cong k[z_1,z_2]/(1-z_1z_2)$ via $t\mapsto z_1$, and hence choose $\ell=2$ to put $G\subset\mathbb{A}^2$. Then

$$\rho(g)=\rho(z_1,z_2)=\begin{pmatrix}z_1 & 0\\ 0 & z_1\end{pmatrix},$$

so writing $p=(a,b)\in V$, we obtain $\sigma_p(z_1,z_2)=(z_1a,z_1b)$. Thus $\sigma_p^*:k[x,y]\to k[G]$ sends $x\mapsto az_1$ and $y\mapsto bz_1$. A matrix for σ_p^* in degree 1 is

$$A(p)=\begin{pmatrix}a & b\\ 0 & 0\end{pmatrix}.$$

Let us follow Algorithm 2.1 and write functions for the tRREF of $A(p)$. To make this example as illuminating as possible, we will skip steps like Step 1 that have a trivial effect on the output. Step 2 yields

$$\begin{pmatrix} aa^\star & b(1-aa^\star+a^\star) \\ 0 & 0 \end{pmatrix}.$$

For simplicity we skip Step 4(a), because the second row is already zero. In Step 4(b), $D' = (b(1-aa^\star+a^\star),0)^T$, where T denotes transpose, $D'' = (0)$ and $D''_0 = (0,0)^T$. Thus $E = (1-aa^\star)\cdot D' = ((1-aa^\star)b(1-aa^\star+a^\star),0)^T$. Now apply the algorithm again to E to obtain a 1×1 matrix

$$R_E = \left((1-aa^\star)b(1-aa^\star+a^\star)\cdot\left[(1-aa^\star)b(1-aa^\star+a^\star)\right]^\star\right) \tag{1}$$

$$= \left((1-aa^\star)b(1-aa^\star+a^\star)\cdot(1-aa^\star)^\star b^\star(1-aa^\star+a^\star)^\star\right) \tag{2}$$

$$= \left((1-aa^\star)bb^\star\right). \tag{3}$$

Of course, the algorithm writes down only line (1) for R_B. To keep this example clear, we obtain line (2) by noting $(fg)^\star = f^\star g^\star$ for $f, g \in \widehat{k[V]}$, noting $(1-aa^\star)(1-aa^\star)^\star = (1-aa^\star)\ \forall a\in k$, and noting $(1-aa^\star+a^\star)(1-aa^\star+a^\star)^\star = 1\ \forall a\in k$. In Step 4(e) we write

$$R'_A = \begin{pmatrix} aa^\star & b(1-aa^\star+a^\star) \\ 0 & (1-aa^\star)bb^\star \end{pmatrix}.$$

Finally, the reduction in Step 4(f) yields

$$R_A = \begin{pmatrix} aa^\star & b(1-aa^\star+a^\star) - b(1-aa^\star+a^\star)\cdot(1-aa^\star)bb^\star \\ 0 & (1-aa^\star)bb^\star \end{pmatrix}$$

$$= \begin{pmatrix} aa^\star & a^\star b \\ 0 & (1-aa^\star)bb^\star \end{pmatrix}$$

after simplifying and recalling $b^2b^\star = b$. It is now straightforward to check that $R_A(p)$ is the tRREF of $A(p)$ for all $p\in k^2$.

2.3. The kernel of a matrix in tRREF

Finding the kernel of a matrix R in RREF is equivalent to solving the system of equations $R\cdot(x_1,\dots,x_n)^T = 0$: for every pivot r_{ij}, write an equation

$$x_j = -r_{i,j+1}x_{j+1} - r_{i,j+2}x_{j+2} - \cdots - r_{i,n}x_n.$$

Set each free variable equal to 1 in turn, set the other free variables to 0, and read off the vector of values in the pivot variables. These vectors give a basis for the kernel of R.

To compute the kernel of an $m\times n$ matrix A, we use the $n\times n$ tRREF matrix R_A containing the rows of the RREF of A: recall there is a pivot in the jth column of the RREF if and only if the row containing that pivot is jth row of $R_A = (r_{ij})$, if and only if $r_{jj} = 1$. Otherwise, $r_{jj} = 0$.

Lemma 2.1. *Let V be a vector space, and let $R_A(p) = (r_{ij})$ be the $n\times n$ tRREF over $\widehat{k[V]}$ of a matrix $A(p)$ over $k[V]$. Then there exists a straight line program Γ^K of length $O(n^2)$ such that $\mathrm{Out}_{n^2}(\Gamma^K, R_A)$ computes a basis for the kernel of $A(p)$, for any $p\in V$.*

Proof. The kernel of A is given by the vectors $\phi_1, \ldots, \phi_n$, in terms of $R_A = (r_{ij})$, as follows:

$$\phi_j := (1 - r_{jj}) \cdot \big(-r_{1j}, -r_{2j}, \ldots, \overbrace{1}^{j\text{th place}}, \ldots, -r_{nj}\big).$$

First recall that the kernel of a RREF matrix has one basis vector for each non-pivot column. So $\phi_j = 0$ if and only if column j of the RREF has a pivot. Otherwise, $\phi_j \neq 0$, as follows: Put the free variable $x_j := 1$. Now, $r_{kj} = 0$ unless there is a pivot in column k of the RREF. Set each pivot variable x_{kk} equal to the negation of the jth entry of the row containing that pivot.

Of course, $r_{ij} = 0$ whenever $i > j$, but such simplifications complicate the formulas without improving the asymptotic length of their straight line programs. As written, each ϕ_j, $j = 1, \ldots, n$, requires two constants from k, one addition, n scalar multiplications, and n other multiplications. Upon adding recall instructions, this straight line program computing the kernel has length $O(n^2)$. □

Example 2.2. In the running example of k^* acting on $V = k^2$, writing $p = (a, b)$ we obtained the tRREF matrix

$$R_A(p) = \begin{pmatrix} aa^\star & a^\star b \\ 0 & (1 - aa^\star)bb^\star \end{pmatrix}.$$

The formulas in the proof of Lemma 2.1 yield vectors

$$\begin{aligned} \phi_1 &= (1 - aa^\star) \cdot (1, 0) \\ &= (1 - aa^\star, 0), \\ \phi_2 &= \big(1 - (1 - aa^\star)bb^\star\big) \cdot (-a^\star b, 1) \\ &= \big(-a^\star b, 1 - (1 - aa^\star)bb^\star\big) \end{aligned}$$

making repeated use of $f^2 f^\star = f$, $f(f^\star)^2 = f^\star$ for all $f \in \widehat{k[V]}$. It is easy to check that ϕ_1, ϕ_2 give generators for the kernel of R_A for all $p = (a, b)$, which is the tRREF of

$$A = \begin{pmatrix} a & b \\ 0 & 0 \end{pmatrix}.$$

2.4. Collecting nonzero rows

Lastly, the main algorithm in Section 4 below that computes orbit-separating functions requires functions to perform the following task: given a matrix A for a linear map, compute a basis for the image of A from among the columns of A. One needs only to choose the columns j of A such that the RREF of A has a pivot in column j, that is, the columns j such that the tRREF $R_A = (r_{ij})$ of A has $r_{jj} = 1$. Hence the diagonal of R_A indicates which columns of A to choose. We now write a straight line program Γ^B over $\widehat{k[V]}$ that collects among the columns of A a basis for the column space.

The input to Γ^B is an $m \times n$ input matrix X over $\widehat{k[V]}$ and an m-vector v over $\widehat{k[V]}$ such that the functions of v only take the values 0 and 1. (In application, the vector v will be the diagonal of the tRREF R_A.) Form a new matrix X' by adjoining v as a column to the left side of X:

$$X' = \begin{pmatrix} v_1 & x_{11} & \cdots & x_{1n} \\ v_2 & x_{21} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ v_m & x_{m1} & \cdots & x_{mn} \end{pmatrix}.$$

Recall the program Γ^E for functions that exchange the rows of an input matrix X to produce an output matrix $Y = (y_{ij})$ with $y_{11} \neq 0$ if possible. Let $Y = \mathrm{Out}(\Gamma^E, X')$; the first row of the output will be the first row indicated by v. Record $\mathbf{y}'_1 := (y_{12}, \ldots, y_{1,n+1})$. Then recursively apply Γ^E to the last $m-1$ rows of Y, to obtain row vectors $\mathbf{y}'_1, \ldots, \mathbf{y}'_m$. Let Γ^B denote the resulting concatenation of the Γ^E's, so that the output of Γ^B on input v and X is an $m \times n$ matrix Y' whose rows are $\mathbf{y}'_1, \ldots, \mathbf{y}'_m$. The rows of X indicated by v appear first in Y', in their original order. Recall that the length of Γ^E applied to an $s \times (n+1)$ matrix is $O(sn)$. Summing sn for $s = 1, \ldots, m$ yields that the program Γ^B has length $O(m^2 n)$.

3. Degree bounds for orbit closures

Let $V \subseteq \mathbb{A}^n$ be an affine variety whose components all have codimension m. Define the *degree* of V to be

$$\deg(V) = \#H \cap V,$$

where H is a generic linear subspace of dimension m. In this section we relate the degree of a variety to degree bounds of polynomials that can define that variety. By then bounding the degree of an orbit closure $\overline{G \cdot p}$, we can bound the degree of the defining polynomials. The following denotes the vanishing sets, in a variety V, of polynomials or an ideal I in $k[V]$:

$$\mathbb{V}(f_1, \ldots, f_n) = \{p \in V \mid f_i(p) = 0 \ \forall i\},$$
$$\mathbb{V}(I) = \{p \in V \mid f(p) = 0 \ \forall f \in I\}.$$

The next lemma is well known.

Lemma 3.1. *Let $V = \mathbb{V}(f_1, \ldots, f_r)$ have codimension m in $\mathbb{A}^n$. Then there exist m generic linear combinations $g_i = \sum a_{ij} f_j$ such that*

$$W := \mathbb{V}(g_1, \ldots, g_m) \supseteq V$$

and W has codimension m.

Here, "generic" means that the set of coefficients a_{ij} for which the lemma holds is Zariski dense in the affine space over k of choices for all the coefficients. Heintz [10] proves a stronger version of the following statement (his Proposition 3), as well as many related results.

Proposition 3.1 *(Heintz). Let $V \subseteq \mathbb{A}^n$ be an irreducible, Zariski closed subset of degree d. Then there exists an ideal $\mathfrak{q}$, generated by polynomials of degree $\leqslant d$, such that $\mathbb{V}(\mathfrak{q}) = V$.*

Now consider a linear algebraic group G acting on affine n-space. When we can bound the degree of an orbit closure $\overline{G \cdot x}$, then we can produce a degree bound for polynomials f_i such that $\overline{G \cdot x} = \mathbb{V}(f_1, \ldots, f_r)$. For an overview of bounds for the degrees of orbits and the (polynomial) degrees of generating invariants, see Derksen [5]. The proof below follows the strategy of the proof of his Proposition 1.2. Henceforth set the following constants:

- $m := \dim G$.
- Fix an embedding of G in $\mathbb{A}^\ell$.
- Fix generators $h_1, \ldots, h_s$ for the ideal $\mathbb{I}(G)$ of functions vanishing on G, and set $M = \max\{\deg(h_i)\}$.
- Let G act on $V = k^n$ with representation

$$\rho : G \to GL_n(k) \quad \text{defined by } \rho : g \mapsto \big(\rho_{ij}(g)\big), \quad \text{for } \rho_{ij} \in k[z_1, \ldots, z_\ell],$$

- $N := \max\{\deg(\rho_{ij})\}$.

Proposition 3.2. *Let G be a linear algebraic group and m, ℓ, M, N as above. If $\overline{G \cdot x}$ is an orbit closure with dimension r, then*

$$\deg(\overline{G \cdot x}) \leqslant N^r M^{\ell - m}.$$

Proof. Let $d = \deg(\overline{G \cdot x})$. For a generic $(n-r)$-dimensional linear subspace $H \subseteq \mathbb{A}^n$, by definition $d = \#(\overline{G \cdot x} \cap H)$. Let $\sigma : g \mapsto g \cdot x$ be the orbit map. Then the degrees of the polynomials defining σ are bounded by N. Hence $\sigma^{-1}(H) = \mathbb{V}(u_1, \ldots, u_r) \subseteq G$ has $\deg(u_i) \leqslant N$ and has at least d irreducible components.

By Lemma 3.1, there exist generic linear combinations f_j of the generators of $\mathbb{I}(G)$ such that $\mathbb{V}(f_1, \ldots, f_{\ell-m})$ is a complete intersection and contains G. Thus

$$\sigma^{-1}(H) \subseteq \mathbb{V}(u_1, \ldots, u_r, f_1, \ldots, f_{\ell-m}) \subset \mathbb{A}^{\ell}.$$

Consider the vanishing of the homogenized polynomials

$$\mathbb{V}(\overline{u}_1, \ldots, \overline{u}_r, \overline{f}_1, \ldots, \overline{f}_{\ell-m}) \subset \mathbb{P}^{\ell}.$$

By a generalization of Bézout's theorem (see Fulton's book [15, Section 12.3.1]), the number of irreducible components of this variety is (generously) bounded by

$$\prod_i \deg\big(\mathbb{V}(\overline{u}_i)\big) \cdot \prod_j \deg\big(\mathbb{V}(\overline{f}_j)\big) = \prod_i \deg(\overline{u}_i) \cdot \prod_j \deg(\overline{f}_j) \leqslant N^r M^{\ell-m}.$$

This number then also bounds d. □

Corollary 3.1. *With the hypotheses of Proposition* 3.2*, there exist polynomials $f_1, \ldots, f_t$ such that $\overline{G \cdot x} = \mathbb{V}(f_1, \ldots, f_t)$ and*

$$\deg(f_i) \leqslant \deg(\overline{G \cdot x}) \leqslant N^r M^{\ell-m}.$$

We denote this degree bound $\delta(N) = N^r M^{\ell-m}$.

Example 3.1. Return to the action of the multiplicative group k^* on $V = k^2$ via $g \cdot (a, b) = (ga, gb)$. We can choose $k[G] = k[t, \frac{1}{t}] \cong k[z_1, z_2]/(1 - z_1 z_2)$ via $t \mapsto z_1$, and hence choose $\ell = 2$ to put $G \subset \mathbb{A}^2$. Note $m = 1$ and $M = 2$. Then

$$\rho(g) = \rho(z_1, z_2) = \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}$$

and $N = 1$. The maximal dimension of an orbit is $r = 1$, so Corollary 3.1 yields a degree bound of $N^r M^{\ell-m} = 1 \cdot 2^{2-1} = 2$ for polynomials defining the orbits. Of course, since the orbits are the origin and the lines through the origin, we can define the orbits with linear polynomials.

4. The algorithm for orbit-separating functions

Fix $G \hookrightarrow \mathbb{A}^{\ell}$, and let $G \xhookrightarrow{\rho} GL_n(k)$ act on $V = k^n$ with constants m, ℓ, M, and N as in Section 3. Write down the map $\sigma_p : G \to V$ for any $p \in V$ and the ring map $\sigma_p^* : k[V] \to k[G]$. Then the orbit closure $\overline{G \cdot p}$ is defined by polynomials of degree bounded by $\delta(N) = N^r M^{\ell-m}$ inside $\ker \sigma_p^*$. Below, we write straight line programs for the set $\mathcal{C}$ of invariant, orbit-separating functions by computing a k-vector space basis for $\ker \sigma_p^*$ up to degree $\delta(N)$. The length of these programs will be polynomial in the dimension n and the degree N of the representation.

4.1. Pre-computations for fixed G

The algorithm to write straight line programs for orbit-separating functions requires the following preliminary calculations related to the fixed linear algebraic group G. Since we are interested in how the length of our separating functions depend on the representation, we ignore the computational complexity of these pre-computations, which we perform once and utilize for all representations of G. First, for the fixed embedding $G \hookrightarrow \mathbb{A}^\ell$, choose a monomial order for the monomials spanning $k[z_1, \ldots, z_\ell]$. Compute a Gröbner basis for $\mathbb{I}(G)$. Now, we will compute elements in $\ker \sigma_p^*$ by identifying relations among *images* $\sigma_p^*(u)$ of monomials $u \in k[V]$. To perform this linear algebra in $k[G]$, we will require bounds for the k-vector space dimension of $k[G]_{\leqslant d}$, which is the space of functions in $k[G]$ of degree $\leqslant d$.

Lemma 4.1. *Let $m = \dim G$. There exists a function $H(d)$, computable from a Gröbner basis for $\mathbb{I}(G)$, such that $H(d) = \dim_k k[G]_{\leqslant d}$ for all $d \geqslant 0$, and $H(d) \leqslant O(d^m)$.*

Proof. Suppose $R = k[G]$ is generated as a k-algebra by $f_1, \ldots, f_r$ of degree 1. Define $S = k[f_1 t, \ldots, f_r t, t] \subseteq R[t]$, and claim $S_d = R_{\leqslant d} \cdot t^d$, where S is graded by t-degree. The inclusion $\supseteq$ is clear, and if $h \in S_d$ is a homogeneous polynomial in t, then the coefficients of t^d can have R-degree no greater than d (less, for example, in the term $f_1 t \cdot t^{d-1}$). Let $H(d)$ be the dth coefficient of the Hilbert series of S, which we may compute from a Gröbner basis for $\mathbb{I}(G)$; see, for example, Eisenbud's book [16, p. 359]. Then $H(d) = \dim_k R_{\leqslant d}$. Since S has dimension bounded by $m+1$, the Hilbert polynomial for S has degree bounded by m. Thus $H(d) \leqslant O(d^m)$. □

4.2. Description of the algorithm

With the Gröbner basis for $\mathbb{I}(G)$ and function $H(d)$ in hand, now choose a representation $\rho : G \to GL_n(k)$, so that G acts on $V = k^n$. Recall we will compute elements in $\ker \sigma_p^*$ by identifying relations among *images* in $k[G]$ of monomials $u \in k[V] = k[x_1, \ldots, x_n]$. The maximal degree among $\sigma_p^*(x_1), \ldots, \sigma_p^*(x_n)$ is $N = \max\{\deg \rho_{ij}\}$, because $\sigma_p(g) = \rho(g) \cdot p$. Thus to compute relations among $\sigma_p^*(u) \in k[G]$ for monomials $u \in k[x_1, \ldots, x_n]$ up to the degree bound $\delta(N)$, we require a k-vector space basis of $\mathbb{I}(G)$ up to degree $N \cdot \delta(N) = N^{r+1} M^{\ell - m}$. Let $B(d)$ denote a basis for $\mathbb{I}(G)$ up to degree i, written as vectors of coefficients relative to the monomial basis for $k[z_1, \ldots, z_\ell]$. So after choosing a representation of G, the first step of the algorithm to write separating functions will be to compute $B(N^{r+1} M^{\ell - m})$.

The idea of the algorithm to write separating functions is as follows; we present pseudo-code for the algorithm further below, and we discuss computational complexity in Section 4.2 below. We regard the embedding $G \hookrightarrow \mathbb{A}^\ell$, the generators $(h_1, \ldots, h_s) = \mathbb{I}(G)$, and the function $H(d)$ as fixed, with parameters $\ell, m = \dim G$, and $M = \max\{\deg h_i\}$. The input is the representation map $\rho : G \to GL_n(k)$ with parameters n, N, and r; the latter is the maximal dimension of an orbit. Note that if we cannot predict r, we may take $r = m$. First compute $B(N^{r+1} M^{\ell - m})$. Next, considering the n coordinates $(p_1, \ldots, p_n)$ of $p \in V = k^n$ as indeterminates, compute the n-vector $\rho(z_1, \ldots, z_\ell) \cdot p$. Then for $i = 1, \ldots, n$, we have $f_i := \sigma_p^*(x_i)$ as the ith entry of $\rho(z_1, \ldots, z_\ell) \cdot p$. Let w_i be the vector of coefficients of f_i relative to the monomial basis for $k[z_1, \ldots, z_\ell]$, and let $w_0 := (1, 0, \ldots, 0)$ represent the constant 1. Let W_1 be the ordered list $(w_0, \ldots, w_n)$. Since we leave p indeterminate, the entries of the w_i are elements of $k[V]$.

We first compute $\ker \sigma_p^*$ in degree one. The degree one polynomials in $\ker \sigma_p^*$ correspond to linear dependence relations among the f_i and 1 in $k[G]$, that is, linear combinations of the f_i and 1 (which have degree $\leqslant N$) that lie in $\mathbb{I}(G)$. To find these combinations, let X_1 be a matrix whose first $s_1 := n + 1$ columns are the w_i and whose remaining columns are the vectors (over k) in $B(N)$. The kernel of X_1 would give a basis for the dependence relations among the columns of X, that is, linear combinations of the w_i that (a) equal zero or (b) equal elements of $\mathbb{I}(G)$ of degree $\leqslant N$. As X_1 is a matrix over $k[V]$, we have an algorithm in Section 2 to write straight line programs for the tRREF of X_1 over $\widehat{k[V]}$, and from the tRREF write functions in $\widehat{k[V]}$ for the vectors in $\ker X_1$. These functions

on V for $\ker X_1$ make up the first elements of our set $\mathcal{C}$ of separating functions. Now, since we only care about relations among the w_i (the elements of $B(N)$ are independent, anyway), we need only record in $\mathcal{C}$ the first n entries of the vectors in $\ker X_1$.

To find degree two polynomials in $\ker \sigma_p^*$, the naive next step is to compute the n^2 products $f_i \cdot f_j$ for all i, j and find linear combinations of these and the original f_i that lie in $\mathbb{I}(G)$. So in degree 2 we would compute relations among $1+n+n^2$ polynomials, in degree 3 we would obtain $1+n+n^2+n^3$ polynomials, and in degree d we would obtain $1+n+n^2+\cdots+n^d = \frac{n^{d+1}-1}{n-1}$ polynomials. Hence the number of columns in our matrices and the length of our straight line programs would grow exponentially in the degree d, up to $d = \delta(N) = N^r M^{\ell-m}$. Consequently, to ensure the length of the straight line programs remains polynomial in n and N, we must control the growth of the products of the f_i in high degree.

To do so, we use the function $H(d) = \dim_k k[G]_{\leqslant d}$: Because $\deg f_i \leqslant N$ for all i, no more than $H(N)$ of the w_i are linearly independent. Let D be the diagonal of the tRREF of X_1, and let Y be the matrix whose rows are $w_0, \ldots, w_n$. Recalling the program Γ^B from Section 2.4 that collects matrix rows indicated by a vector D, we have $Y' = \mathrm{Out}(\Gamma^B, D, Y)$ is a matrix whose first $t_2 := \min\{n, H(N)\}$ rows have the same span as $\{w_0, \ldots, w_n\}$. Let $L_1 := \{w_{i_1}, \ldots, w_{i_{t_2}}\}$ denote these first t_2 rows of Y'. Thus to compute the degree-two polynomials in $\ker \sigma_p^*$, it suffices to start with the set

$$L_1 \cup \{f_{i_T} \cdot f_j \mid T = 1 \ldots t_2,\ j = 1 \ldots n\}.$$

Let W_2 be set of vectors of coefficients of these polynomials relative to the monomial basis for $k[z_1, \ldots, z_\ell]$ up to degree $2N$. The size of the set W_2 is bounded by $H(N) + n \cdot H(N)$, which is polynomial in n and N as desired.

Let X_2 be the matrix whose first $s_2 := |W_2|$ columns are the vectors in W_2 and whose remaining columns are the vectors in $B(2N)$. As above, we write down straight line programs for functions in $\widehat{k[V]}$ that compute the tRREF and kernel of X_2. We next adjoin to $\mathcal{C}$ the first s_2 entries of each kernel vector. We use the diagonal of the tRREF of X_2 and the program Γ^B to again identify a spanning set L_2 among the polynomials represented by W_2. We then proceed with L_1 and with the products $\{f \cdot f_i \mid f \in L_2, i = 1, \ldots, n\}$ to find the degree-three polynomials in $\ker \sigma_p^*$. The algorithm iterates in this way through degree $\delta(N)$.

The pseudo code below describes precisely the algorithm to compute orbit-separating functions in $\widehat{k[V]}$. Notice the algorithm adds detail ensuring various matrices and vectors are the appropriate size. Here, if v is a vector of length $t \geqslant s$, then $\pi_s(v)$ is the projection to the first s entries of v. The algorithm for the fixed algebraic group G has built-in parameters m and ℓ for the embedding of G, generators for the ideal of $\mathbb{I}(G)$, and the function $H(d)$. Further comments follow.

Algorithm 4.1 *(Separating functions for a representation of fixed G).* PRE-COMPUTATIONS: Generators $(h_1, \ldots, h_t) = \mathbb{I}(G)$, the function $H(d)$, constants $m = \dim G$ and ℓ from the embedding $G \hookrightarrow \mathbb{A}^\ell$.
INPUT: Representation map $\rho : G \to GL_n(k)$, parameters $n = \dim V$, $N = \max\{\deg \rho_{ij}\}$, and $r =$ maximal orbit dimension, if known, otherwise $r = m$.
OUTPUT: A set $\mathcal{C} \subset \widehat{k[V]}$ of invariant, orbit-separating functions.

1. Compute $B(N^{r+1} M^{\ell-m})$.
2. For $d = 2$ to $d = N^r M^{\ell-m}$, set $t_d := \min\{\frac{n^{d+1}}{n-1}, H(dN)\}$, or set $t_d = \min\{1+d, H(dN)\}$ if $n = 1$.
3. For $i = 1, \ldots, n$, compute $f_i := \sigma_p^*(x_i)$, where the coordinates $(p_1, \ldots, p_n) = p$ are indeterminates.
4. For $i = 1, \ldots, n$, let w_i be the vector of coefficients of $\sigma_p^*(x_i)$ with respect to the monomial basis of $k[z_1 \ldots, z_\ell]$, where 1 is represented by $w_0 = (1, 0, \ldots, 0)$. The coefficients lie in $k[V]$.
5. $\mathcal{C}_0 := \emptyset$, $W_1 := \{w_0, \ldots, w_n\}$, $s_1 := n+1$.
6. FOR $d = 1$ TO $d = N^r M^{\ell-m}$, DO
   (a) Let X_d be he matrix whose first columns are the vectors of W_d and whose remaining columns are the vectors of $B(dN)$.
   (b) Compute $R_d := \mathrm{Out}(\Gamma^{tR}, X_d)$, the tRREF of X_d.
   (c) Compute $\beta := \mathrm{Out}(\Gamma^K, R_d)$, a basis for $\ker X_d$.

(d) Let $\mathcal{C}_d := \mathcal{C}_{d-1} \cup \{\pi_{s_d}(v) \mid v \in \beta\}$.
(e) Let Y be the matrix whose rows are the vectors in W_d. Let D be the first s_d entries on the diagonal of the tRREF X_d.
(f) Compute $Y' := \mathrm{Out}(\Gamma^B, \{Y, D\})$, collecting together the rows of Y indicated by D.
(g) Let $L_d = \{w_{i_1}, \ldots, w_{i_{t_d}}\}$ be the first t_d rows of Y'.
(h) $s_{d+1} := t_d + n \cdot (t_d - t_{d-1})$.
(i) For $j = t_{d-1} + 1, \ldots, t_d$, compute the product of the polynomial represented by w_{i_j} with each of $f_1, \ldots, f_n$. Label the vectors of coefficients of these products $w_{t_d+1}, \ldots, w_{s_{d+1}}$.
(j) $W_{d+1} := \{w_{i_1}, \ldots, w_{i_{t_d}}, w_{t_d+1}, \ldots, w_{s_{d+1}}\}$.
7. $\mathcal{C} := \mathcal{C}_d$.
8. OUTPUT $\mathcal{C}$.

In Step 2, we compute in advance the maximal number t_d of image vectors $w_j \in W_d$ that can be linearly independent, that is, the number of vectors the algorithm should preserve from W_d in Steps (e)–(i). This number is bounded by $H(dN)$, but $|W_d|$ may be less than $H(dN)$. Indeed, recall from the discussion above that at first the number of image vectors is $s_d = |W_d| = 1 + n + n^2 + \cdots + n^d = \frac{n^{d+1}-1}{n-1}$. While s_d follows this pattern we use $t_d = s_d$, and Steps (e)–(i) just carry over all the images vectors to the next iteration. Once $H(dN) = t_d > |W_d|$, we know that at most $H(dN)$ vectors in W_d can be linearly independent, and Steps (e)–(i) carry over only a spanning set for the images of monomials in degree $\leqslant d$.

Note also that Step (i) multiplies only the polynomials represented by the most recent image vectors in $k[z_1, \ldots, z_\ell]$, those with indices greater than t_{d-1}, by the f_i to obtain new image polynomials in higher degree. The coefficients of all the polynomials that arise remain functions of the indeterminate coordinates $p = (p_1, \ldots, p_n)$. We will discuss algorithms for efficient polynomial multiplication in the next section. Now we present proof of the correctness of the output of Algorithm 4.1.

Proposition 4.1. *The functions defined by the set $\mathcal{C} \subset \widehat{k[V]}$ are constant on the orbit of any $p \in V$ and separate the orbits of G.*

Proof. We first show that the functions in $\mathcal{C}$ are constant on orbits, that is, invariant under the action $g \cdot f(x) = f(g^{-1} \cdot x)$ for $g \in G$. Choose $p \in \mathbb{V}$ and $q \in G \cdot p$. Let $X_d(p)$ be the matrix X_d of the algorithm, with its entries evaluated at p. Let $X_d^W(p)$ be the first $|W_d| = s_d$ columns of $X_d(p)$, that is, those containing the vectors in $W_d(p)$. Now, $X_1^W(p)$ and $X_1^W(q)$ have the same kernel, because (1) as maps $k[x_1, \ldots, x_n]_{\leqslant 1} \to k[G]_{\leqslant N}$ they are written relative to same basis $x_1, \ldots, x_n$ for their domain, and because (2) the kernel of each matrix must span $\mathbb{I}(G \cdot p)$ in degree one and smaller. Thus $X_1^W(p)$ and $X_1^W(q)$ have the same tRREF and so have the same linearly independent columns in the same locations.

So letting $\mathcal{C}_d(p)$ denote the kernel vectors obtained on input $p \in V$ in the dth iteration, we conclude $\mathcal{C}_1(p) = \mathcal{C}_1(q)$. As well, let $L_d(p)$ denote the set L_d of the algorithm with functions evaluated at p, but for clarity we write the *polynomials* represented by the vectors in $L(p)$. Hence from the previous paragraph we also conclude that $L_1(p) = \{1, \sigma_p^*(x_{j_1}), \ldots, \sigma_p^*(x_{j_r})\}$ and $L_1(q) = \{1, \sigma_q^*(x_{j_1}), \ldots, \sigma_q^*(x_{j_r})\}$ for the same indices $j_1, \ldots, j_s$.

Proceed by induction on d: we assume $X_d^W(p)$ and $X_d^W(q)$ have the same tRREF and hence $\mathcal{C}_d(p) = \mathcal{C}_d(q)$. We may also assume the columns of $X_d^W(p)$ and $X_d^W(q)$ represent the images of the same set of monomials $\{x^{I_1}, \ldots, x^{I_s}\}$, for multi-indices I_j. Then the lists $W_{d+1}(p)$ and $W_{d+1}(q)$ also represent the images of the same monomials under σ_p^* and σ_q^*, respectively. Claim again that $X_{d+1}^W(p)$ and $X_{d+1}^W(q)$ have the same tRREF. By the induction hypothesis, the two matrices are written relative to the same basis for their domain, and the kernel of each must span $I(G \cdot p)_{d+1}$. These facts prove the claim, as in the base case. Thus $\mathcal{C}_{d+1}(p) = \mathcal{C}_{d+1}(q)$, and the functions in $\mathcal{C}$ are invariant.

To show the functions in $\mathcal{C}$ separate orbits, we prove the converse of the paragraphs above: choose $p, q \in V$ such that the functions in $\mathcal{C}$ take the same values at both points, and show that $p \in G \cdot q$. The key is that the matrices $X_1^W(p)$ and $X_1^W(q)$ are written relative to the same bases for their do-

main space $k[x_1,\dots,x_n]_1$, namely, $x_1,\dots,x_n$. Therefore $\ker X_1(p) = \mathcal{C}_1(p) = \mathcal{C}_1(q) = \ker X_1(q)$ implies $\ker \sigma_p^* = \mathbb{I}(G\cdot p)$ and $\ker \sigma_q^* = \mathbb{I}(G\cdot q)$ agree in degree $\leqslant 1$. By proving this fact for all degrees $d \leqslant \delta(N)$, we will prove $\overline{G\cdot p} = \overline{G\cdot q}$. Since G is a linear algebraic group, it would follow $G\cdot p = G\cdot q$.

First observe that since $\ker X_1(p) = \ker X_1(q)$, then in fact $X_1(p)$ and $X_1(q)$ have the same tRREF. It follows, with notation as above, that

$$L_1(p) = \left\{1, \sigma_p^*(x_{j_1}), \dots, \sigma_p^*(x_{j_s})\right\} \quad \text{and} \quad L_1(q) = \left\{1, \sigma_q^*(x_{j_1}), \dots, \sigma_q^*(x_{j_s})\right\}$$

for the same indices $j_1,\dots,j_s$, because $X_1(p)$ and $X_1(q)$ have linearly independent columns in the same locations. Thus $W_2(p)$ and $W_2(q)$ list the images of the same set of monomials x_jx_k under σ_p^* and σ_q^*, respectively. Proceeding by induction, assume $X_d(p)$ and $X_d(q)$ have the same tRREF and list the images of the same monomials. Then $X_{d+1}^W(p)$ and $X_{d+1}^W(q)$ also list the images of the same monomials. This last fact and the assumption $\ker X_d(p) = \mathcal{C}_{d+1}(p) = \mathcal{C}_{d+1}(q) = \ker X_d(q)$ together yield that $\ker\sigma_p^*$ and $\ker\sigma_q^*$ agree in degrees through $d+1$, completing the induction. It follows $G\cdot p = G\cdot q$. □

4.3. Computational complexity bounds

The bookkeeping that follows confirms that the number and total length of the orbit-separating functions in $\mathcal{C}$ is polynomial in n and N.

4.3.1. Polynomial multiplication

Step (i) of Algorithm 4.1 requires computing the products of polynomials via straight line programs on their coefficients. In computations we will represent polynomials as vectors of coefficients relative to the monomial basis of $k[z_1,\dots,z_\ell]$. So in this section we drop the distinction between polynomials and their vector representations. Again we write $f_1 = \sigma_p^*(x_1),\dots,f_n = \sigma_p^*(x_n)$. Hence the products we compute are of the form $f_{i_1}\cdots f_{i_d}$, and we consider them as polynomials in $(k[V])[z_1,\dots,z_\ell]$, that is, polynomials in variables z_j with coefficients in $k[V]$ that are functions of $p\in V$. Since the matrices X_d include the coefficients in $k[V]$ of the polynomials $f_{i_1}\cdots f_{i_d}$, the coefficients contribute to the length of functions in the $\ker X_d$ and hence of functions in $\mathcal{C}$. Therefore we must bound the length of the straight line programs for these coefficients as functions of p.

Recall that in the FOR loop of Algorithm 4.1, Steps (e)–(h) select a linearly independent subset $L_d = \{w_{i_1},\dots,w_{i_{t_d}}\}$ of the set of polynomials W_d under consideration in iteration d. As $|L_d| = t_d$, each of the last $t_d - t_{d-1}$ elements of L_d is a product $f_{i_1}\cdots f_{i_d}$. Step (i) then multiplies each of these last elements with each of $f_1,\dots,f_n$.

Let f and g be polynomials in $k[z_1,\dots,z_\ell]$, each of degree (at most) d. Fast algorithms for computing the coefficients of fg with no more than $O(\ell\cdot d^\ell\log d)$ steps exist by employing Fast Fourier Transforms or evaluation homomorphisms; see Moenck [17] for a survey, Pan [18] for a fast method, and the book [12] for a exploration of the univariate case. We instead employ the naive classical or "grade school" approach for the sake of clarity; the $O(d^{2\ell})$ operations required for this procedure will suit our goals for polynomial-length straight line programs

To outline the classical algorithm in ℓ variables, first consider the univariate case $f,g\in k[z_1]$, each of degree d. Then computing fg requires $(d+1)^2$ multiplications and d^2 additions. Now let $f,g\in k[z_1,\dots,z_\ell]$, but consider f and g as univariate polynomials in z_1 with coefficients in $k[z_2,\dots,z_\ell]$, that is, consider $f,g\in k[z_1][z_2,\dots,z_\ell]$. Then apply the univariate case recursively; Moenck in [17] yields that we obtain straight line programs of total length $O(d^{2\ell})$ for the coefficients of fg.

In iteration d of the FOR loop of Algorithm 4.1, Step (i) performs $n\cdot(t_d - t_{d-1})$ polynomial multiplications over $(k[V])[z_1,\dots,z_\ell]$. Recall that for large d, $t_d = H(dN) = O(d^mN^m)$. The degree of the polynomials to multiply in iteration d is at most dN. Thus the length of the straight line programs to perform the multiplications in iteration d is at most $O(d^mN^m\cdot d^{2\ell}N^{2\ell}) = O(d^{m+2\ell}N^{m+2\ell})$.

4.3.2. Counting functions and their length

We now have a bound on the contribution of the polynomial multiplication steps to the length of functions in $\mathcal{C}$. The remaining straight line programs in Algorithm 4.1 perform linear algebra, and

their length depends upon the dimensions of the input matrices. Throughout this section we employ the fact $\sum_{d=1}^{D} d^k = O(\frac{1}{k+1} D^{k+1})$ for a fixed positive integer k.

We first bound the size of the matrices X_d that occur. Recall that for large d, $t_d = H(dN) \leqslant O(d^m N^m)$, and

$$\begin{aligned} |W_d| = s_d &= t_{d-1} + n(t_{d-1} - t_{d-2}) \\ &= H\big((d-1)N\big) + n\big[H\big((d-1)N\big) - H\big((d-2)N\big)\big] \\ &= O\big(nd^m N^m\big). \end{aligned}$$

The columns of X_d are the elements of W_d and of $B(dN)$, the k-basis for $\mathbb{I}(G)$ through degree dN. To bound $|B(dN)|$, we note that the space of degree-j monomials in $k[z_1, \ldots, z_\ell]$ has dimension

$$\binom{j+\ell-1}{\ell} = \frac{(j+\ell-1)!}{\ell!(j-1)!} \leqslant \frac{(j+\ell-1)^\ell}{\ell!} = O\big(j^\ell\big).$$

Summing this dimension from $j = 1$ to $j = dN$ yields $|B(dN)| = O(d^{\ell+1} N^{\ell+1})$. Adding the bound for $|W_d|$, we conclude X_d has $O(nd^m N^m + d^{\ell+1} N^{\ell+1}) \leqslant O(nd^{\ell+1} N^{\ell+1})$ columns, because $m \leqslant \ell$.

Likewise, X_d has $O(d^{\ell+1} N^{\ell+1})$ rows corresponding to the monomial basis for $k[z_1, \ldots, z_\ell]_{\leqslant dN}$. Now, the program Γ^{tR} in Section 2.2 for the tRREF of an $s \times t$ matrix has length $O(st^2 + t^3)$. Thus the program for the tRREF of X_d has length bounded by $O((nd^{\ell+1} N^{\ell+1})^3)$.

Recall that the program Γ^K in Section 2.3 for the kernel of an $t \times t$ matrix in tRREF has length $O(t^2)$. Hence the above count of the columns of X_d also yields that the program for the kernel of tRREF(X_d) has length $O((d^{\ell+1} N^{\ell+1})^2)$

In Steps (e)–(g) of the algorithm, the program Γ^B in Section 2.4 collects a spanning subset of W_d. On an $s \times t$ matrix, this program has length $O(s^2 t)$. The input Y is the matrix whose rows are the elements of W_d. Thus Y has $O(nd^m N^m)$ rows and $O(d^{\ell+1} N^{\ell+1})$ columns, and the length of Γ^B in iteration d is $O(n^2 d^{2m+\ell+1} N^{2m+\ell+1})$.

In Section 4.3.1 we computed the bound $O(d^{m+2\ell} N^{m+2\ell})$ for the length of the programs for polynomial multiplication in Step (i) of iteration d. After reviewing the above estimates, we observe that the fastest-growing contribution to the length of the programs in iteration d comes from the tRREF program Γ^{tR}, with length $O(n^3 d^{3\ell+3} N^{3\ell+3})$. This program dominates the polynomial multiplication and vector selection Γ^B because $m \leqslant \ell$. So we conclude that the length of the programs for functions in $\mathcal{C}$ computed in iteration d is $O(n^3 d^{3\ell+3} N^{3\ell+3})$. Summing this length from $d = 1$ to the degree bound $d = N^r M^{\ell-m}$ yields that the total length of the functions in $\mathcal{C}$ as straight line programs is

$$O\big(n^3 \big(N^r M^{\ell-m}\big)^{3\ell+4} N^{3\ell+3}\big) = O\big(n^3 N^{3\ell(r+1)+4r+3} M^{(3\ell+4)(\ell-m)}\big),$$

where as always, $n = \dim V$, N is the maximum degree of the polynomials defining the representation $G \to GL_n$, M is a degree bound for a generating set of $\mathbb{I}(G) \subset k[z_1, \ldots, z_\ell]$, and under this embedding G has dimension m. Since the embedding $G \hookrightarrow \mathbb{A}^\ell$ is fixed, we omit the constant power of M from the asymptotic complexity in Theorem 1.1.

Finally, we bound the number of separating functions in $\mathcal{C}$ that Algorithm 4.1 computes. These functions are the entries in the kernel vectors of X_d computed by Γ^K in each iteration d. For a $t \times t$ tRREF input matrix, the program Γ^K computes t vectors of dimension t. Since the tRREF of X_d has $O(nd^{\ell+1} N^{\ell+1})$ columns, it follows that iteration d of the algorithm computes $O(n^2 d^{2\ell+2} N^{2\ell+2})$ separating functions. Summing this count from $d = 1$ to the degree bound $d = N^r M^{\ell-m}$ yields

$$O\big(n^2 \big(N^r M^{\ell-m}\big)^{2\ell+3} N^{2\ell+2}\big) = O\big(n^2 N^{2\ell(r+1)+3r+2} M^{(2\ell+3)(\ell-m)}\big)$$

functions in $\mathcal{C}$. By omitting the powers of M, Theorem 1.1 follows.

5. Conclusion

Given a representation of any fixed linear algebraic group, Algorithm 4.1 computes invariant functions in the ring extension $\widehat{k[V]}$ that separate the orbits of the group action. What is more, there are polynomial bounds, in the parameters of the representation, for the number of the functions and their total length as straight line programs. It emerges that the "quasi-inverse" provides an extension of the polynomial functions sufficient to write separating functions with a polynomial measure of their complexity.

Acknowledgments

I thank my doctoral adviser, Harm Derksen, for his indispensable ideas and guidance, Frank Sottile and the anonymous referee for their many comments on style and clarity, and Evelyne Hubert, for pointing out the effective nullstellensatz and quantifier elimination approaches to this problem.

References

[1] H. Derksen, G. Kemper, Computational Invariant Theory, Invariant Theory and Algebraic Transformation Groups, I, Encyclopaedia Math. Sci., vol. 130, Springer-Verlag, Berlin, 2002.
[2] G. Kemper, Computing invariants of reductive groups in positive characteristic, Transform. Groups 8 (2) (2003) 159–176.
[3] V.L. Popov, Constructive invariant theory, in: Young Tableaux and Schur Functors in Algebra and Geometry, Toruń, 1980, in: Astérisque, vol. 87, Soc. Math. France, Paris, 1981, pp. 303–334.
[4] D.L. Wehlau, Constructive invariant theory for tori, Ann. Inst. Fourier (Grenoble) 43 (4) (1993) 1055–1066.
[5] H. Derksen, Polynomial bounds for rings of invariants, Proc. Amer. Math. Soc. 129 (4) (2001) 955–963 (electronic).
[6] M. Domokos, Typical separating invariants, Transform. Groups 12 (1) (2007) 49–63.
[7] G. Kemper, Separating invariants, J. Symbolic Comput. 44 (9) (2009) 1212–1222.
[8] M. Giusti, J. Heintz, J. Sabia, On the efficiency of effective Nullstellensätze, Comput. Complexity 3 (1) (1993) 56–95.
[9] S. Puddu, J. Sabia, An effective algorithm for quantifier elimination over algebraically closed fields using straight line programs, J. Pure Appl. Algebra 129 (2) (1998) 173–200.
[10] J. Heintz, Definability and fast quantifier elimination in algebraically closed fields, Theoret. Comput. Sci. 24 (3) (1983) 239–277.
[11] H. Kadish, Complexity in invariant theory, PhD thesis, University of Michigan, ProQuest Dissertations and Theses, 2011.
[12] P. Bürgisser, M. Clausen, M.A. Shokrollahi, Algebraic Complexity Theory, Grundlehren Math. Wiss., vol. 315, Springer-Verlag, Berlin, 1997, with the collaboration of Thomas Lickteig.
[13] S.J. Berkowitz, On computing the determinant in small parallel time using a small number of processors, Inform. Process. Lett. 18 (3) (1984) 147–150.
[14] K. Mulmuley, A fast parallel algorithm to compute the rank of a matrix over an arbitrary field, Combinatorica 7 (1) (1987) 101–104.
[15] W. Fulton, Intersection Theory, Ergeb. Math. Grenzgeb. (3) (Results in Mathematics and Related Areas (3)), vol. 2, Springer-Verlag, Berlin, 1984.
[16] D. Eisenbud, Commutative Algebra, Grad. Texts in Math., vol. 150, Springer-Verlag, New York, 1995.
[17] R.T. Moenck, Another polynomial homomorphism, Acta Inform. 6 (2) (1976) 153–169.
[18] V.Y. Pan, Simple multivariate polynomial multiplication, J. Symbolic Comput. 18 (3) (1994) 183–186.