ARTICLE IN PRESS

Journal of Symbolic Computation ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Journal of Symbolic Computation

journal homepage: www.elsevier.com/locate/jsc

Implementation of Stanley's algorithm for projective group imbeddings

Roy Joshua [a,1], Shaun Van Ault [b]

[a] Department of Mathematics, Ohio State University, Columbus, OH, 43210, USA

[b] Department of Mathematics, Fordham University, Bronx, NY, 10458, USA

ARTICLE INFO

Article history:
Received 25 May 2008
Accepted 6 November 2008
Available online xxxx

Keywords:
Stanley's algorithm
Toric varieties
Group imbeddings
GAP
LiE
Polymake

ABSTRACT

Computing intersection cohomology Betti numbers is complicated by the fact that the usual long exact localization sequences in Borel–Moore homology do not carry over to the setting of intersection homology. Nevertheless, about 20 years ago, Richard Stanley had formulated a remarkable algorithm for computing the intersection cohomology Betti numbers of toric varieties. During the last few years, Michel Brion and the first author were able to extend this to a much larger class of spherical varieties. This algorithm has been implemented as an interactive script written for the programming package `GAP` (using also `LiE` and `polymake`) by the authors. This paper is an exposition of this implementation.

1. Introduction

We begin with the following elementary example to illustrate why computing intersection homology is quite difficult (and why, in contrast, computing Borel–Moore homology is often quite easy).

Let $Grass_2(\mathbb{C}^4)$ denote the Grassmannian of 2-planes in $\mathbb{C}^4$. Let Sch_2 denote the Schubert variety defined by $\{V \in Grass_2(\mathbb{C}^4) \mid dim(V \cap \mathbb{C}^2) \geq 1\}$. This variety has the following explicit cell decomposition into complex cells C_i, $i = 0, 1, 2, 3$, and C_2' with $dim(C_i) = i$ and $dim(C_2') = 2$:

$$C_0 = \{V \in Grass_2 \mid dim(V \cap \mathbb{C}^i) = i, i = 1, 2\}$$
$$C_1 = \{V \in Grass_2 \mid dim(V \cap \mathbb{C}^i) = 1, i = 1, 2, dim(V \cap \mathbb{C}^3) = 2\}$$
$$C_2 = \{V \in Grass_2 \mid dim(V \cap \mathbb{C}^i) = i - 1, i = 2, 3, dim(V \cap \mathbb{C}^1) = 0\}$$

E-mail addresses: joshua@math.ohio-state.edu (R. Joshua), ault@fordham.edu (S. Van Ault).

[1] Current address: Max Planck Institute for Mathematics, Vivatsgasse 7, 53111 Bonn, Germany.

0747-7171/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsc.2008.11.002

$$C_2' = \{V \in Grass_2 \mid dim(V \cap \mathbb{C}^i) = 1, i = 1, 2, 3\}.$$
$$C_3 = \{V \in Grass_2 \mid dim(V \cap \mathbb{C}^2) = 1, dim(V \cap \mathbb{C}^3) = 1, dim(V \cap \mathbb{C}^1) = 0\}.$$

Observe that the cells are all complex cells, and therefore of even real dimension. Therefore, the usual localization sequences in Borel–Moore homology associated with the filtration provided by the above cell decomposition break up into short exact sequences. Consequently one may read off the Borel–Moore homology as $H_0(Sch_2;\mathbb{Q}) \cong \mathbb{Q} \cong H_6(Sch_2;\mathbb{Q})$, $H_2(Sch_2;\mathbb{Q}) \cong \mathbb{Q}$ and $H_4(Sch_2;\mathbb{Q}) \cong \mathbb{Q} \oplus \mathbb{Q}$. The last two isomorphisms follow from the fact there is exactly one 1-cell whereas there are two 2-cells. In particular, the above computations show the failure of Poincaré duality which, in turn, implies the Schubert variety Sch_2 is singular.

Computing the intersection cohomology of the above Schubert variety is quite involved; it makes use of a small resolution of singularities given explicitly as a variety of partial flags. (See Beilinson et al. (1982) and Joshua (1987) for basic results on intersection cohomology.) In this particular case, we are fortunate in having the existence of such a small resolution, but clearly this is a very special situation. It should be in this context that one should view the remarkable algorithm conjectured by Richard Stanley (see Stanley (1987)) that computes the intersection cohomology Betti numbers of all projective toric varieties. This was proved shortly thereafter by several groups independently: see for example, Denef and Loeser (1991) and Fieseler (1991). We may summarize this result as the following theorem.

Theorem 1.1. *Let X be a complex projective toric variety of dimension d, and with dense orbit T. Let $IP_X(t)$ ($IP_{X,x}(t)$) be the Poincaré polynomial for global intersection cohomology (for the stalks of the intersection cohomology sheaves at $x \in X$, respectively). Then*

$$IP_X(t) = \sum_x (t^2 - 1)^{d-r_x} IP_{X,x}(t) \tag{1.0.1}$$

(sum over representatives of T-orbits in X) and

$$IP_{X,x}(t) = IP_{\mathcal{S}_x,x}(t) = \begin{cases} \tau_{\leq d_x - 1}((1-t^2) IP_{\mathbb{P}(\mathcal{S}_x)}(t)), & d_x \geq 2, \\ 1 & otherwise. \end{cases} \tag{1.0.2}$$

$\mathcal{S}_x$ is a slice at x to Tx of dimension d_x; $\mathbb{P}(\mathcal{S}_x) = (\mathcal{S}_x - x)/\mathbb{C}^*$ is the corresponding link for an attractive action of the multiplicative group, and $\tau_{\leq d_x - 1}$ denotes the truncation of a polynomial to degrees $\leq d_x - 1$. (Making use of Poincaré duality one has $t^{2d} IP_X(1/t) = IP_X(t)$, so one may rewrite the first formula above as $IP_X(t) = \sum_x (1-t^2)^{d-r_x} t^{2d_x} IP_{X,x}(1/t)$ where $d_x = d - r_x =$ the dimension of the slice at x and r_x is the dimension of the stabilizer at x.)

In the late 1990s, Michel Brion and the first author discovered a new technique that extended the above algorithm of Stanley to a large class of spherical varieties and also provided a more conceptual framework for understanding the already known results for toric varieties. In Brion and Joshua (2001), the authors proved that certain orbit stratifications were equivariantly perfect for intersection cohomology, in the sense that the long exact sequences in equivariant intersection cohomology associated with this stratification broke up into short exact sequences. This provides new proofs of the vanishing of odd dimensional intersection cohomology sheaves in many cases. More importantly it was able to provide a computational tool extending Stanley's algorithm to a large class of spherical varieties. In Brion and Joshua (2004), the authors work this out in detail for the family of projective spherical varieties called projective reductive varieties. The goal of the present paper is to discuss the implementation of this algorithm for the important subclass of reductive varieties given by projective group imbeddings (or compactifications of algebraic groups). The implementation is written as a `GAP` package (see The GAP Group, 1986), which will call the program `LiE` (see van Leeuwen et al., 1996) to compute the purely Lie-theoretic invariants and `polymake` for questions about convex polytopes. A key use is made of the observation that the Poincaré polynomial in intersection cohomology for these varieties can be computed from the combinatorial data given by a polytope and the Lie-theoretic data associated with the specified algebraic group. (See Theorem 2.3.)

The *organization of the paper* is as follows. As the title of the paper indicates, we concentrate on the implementation aspects of the above algorithm. The second section sets up the framework for the rest

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

of the paper. We summarize all the necessary background information on reductive varieties there. We conclude that section with the theorem extending Theorem 1.1 to projective reductive varieties as proven in Brion and Joshua (2004). The next section is devoted to an algorithmic reformulation of the above theorem for group imbeddings. There we also discuss several examples and trace the flow of the algorithm for these examples. We discuss implementation-specific details in the next section. The final section gives a sample of the traces of the program on the examples considered earlier.

1.1. Notation and conventions

In the rest of this section, we will establish the terminology and conventions that will be adopted for the rest of the paper. Throughout the paper we will restrict to complex algebraic varieties. We denote by G a complex linear algebraic group, and by G^0 the connected component of the identity in G. A separated reduced scheme X of finite type over $\mathbb{C}$ provided with an action by G will be called a *G-variety*; observe that varieties need not be irreducible.

Consider a G-variety X and a point $x \in X$; let Gx be its G-orbit and G_x its isotropy group. A *slice* to Gx at x is a locally closed subvariety $\mathcal{S}$ of X containing x and satisfying the following two conditions: (i) there exists a maximal torus T_x of G_x such that $\mathcal{S}$ is stable under T_x and (ii) the map $G \times \mathcal{S} \to X$ sending (g, x) to gx is *smooth* at (e, x), and the dimension of $\mathcal{S}$ is the codimension of Gx in X. Observe that $\mathcal{S}$ always exists since we have restricted to complex algebraic varieties.
Let T denote a torus acting on a variety X with a fixed point x. We say that x is *attractive* if there exists a one-parameter subgroup $\lambda : \mathbb{C}^* \to T$ such that, for all y in a Zariski neighborhood of x, we have $\lim_{t\to 0} \lambda(t)y = x$. We say that $\mathcal{S}$ is an *attractive slice*, if x is an attractive fixed point for the action of T_x on $\mathcal{S}$. (See Brion and Joshua (2001, Appendix (A.1)) for further details on attractive fixed points.) In this case, the geometric quotient $\mathbb{P}(\mathcal{S}_x) = (\mathcal{S}_x - x)/\mathbb{C}^*$ exists, and we call it the *link* at x. This is a projective variety, since $\mathcal{S}_x$ is assumed to be affine.

2. Reductive varieties and their intersection cohomology computations

This section serves as a framework for the rest of the paper. We will summarize the key properties of reductive varieties and group imbeddings from Brion and Joshua (2004).

Let G denote a complex connected reductive group with B, B^- opposite Borel subgroups, i.e., $T = B \cap B^-$ is a maximal torus of G; let $U = R_u(B)$, $U^- = R_u(B^-)$ be the unipotent radicals of B, B^-. The character group of T will be denoted by Λ and called the *weight lattice*; we put

$$\Lambda_{\mathbb{R}} = \Lambda \otimes_{\mathbb{Z}} \mathbb{R}. \tag{2.0.1}$$

W will denote the Weyl group of (G, T); it acts on Λ and hence on $\Lambda_{\mathbb{R}}$. Φ will denote the root system of (G, T), with the subset Φ^+ of positive roots (the roots of (B, T)), and Π of simple roots.

For any subset $I \subseteq \Pi$, Φ_I will denote the corresponding subsystem of Φ, and W_I (resp. $P_I \supseteq B$ and $P_I^- \supseteq B^-$) will denote the corresponding parabolic subgroup of W (opposite parabolic subgroups of G, respectively). $L_I = P_I \cap P_I^-$; this is a Levi subgroup of P_I and P_I^-, with root system Φ_I and Weyl group W_I. Let ℓ be the length function of W, and let W^I be the subset of representatives of minimal length of W/W_I. Then the Poincaré polynomial of G/P_I equals $\sum_{w\in W^I} t^{2\ell(w)}$.

Next consider the connected reductive group $G \times G$, with Borel subgroup $B^- \times B$ and maximal torus $T \times T$. diag T will denote the diagonal in $T \times T$. An affine $G \times G$-variety X is called *reductive* (for G) if it satisfies the following conditions: (i) X is normal, (ii) there exists $x \in X$, fixed by diag T, such that the orbit $(B^- \times B)x$ is dense in X and (iii) the isotropy group $(G \times G)_x$ is connected. Note that the set of all $x \in X$ satisfying (ii) is a unique $T \times T$-orbit: any such point will be called a *base point*.

The Bruhat decomposition implies that multiplication of G yields an open immersion $U^- \times T \times U \to G$. Thus the group G, viewed as a $G \times G$-variety via left and right multiplication, is an affine reductive variety. More generally, all affine $G \times G$-equivariant embeddings of G, or of quotients of G by connected normal subgroups, are reductive varieties for G. One may consult Brion and Joshua (2004, Section 5) or Alexeev and Brion (2004) for further details on affine reductive varieties.

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

Given a complex connected reductive group G as above, next we will consider projective $G \times G$-reductive varieties as in Alexeev and Brion (2004) and Brion and Joshua (2004, Section 5). We will recall their definition from Brion and Joshua (2004) presently.

Consider a projective irreducible $G \times G$-variety X equipped with an ample $G \times G$-linearized line bundle L. Let $R = \bigoplus_{n=0}^{\infty} \Gamma(X, L^{\otimes n})$, where $\Gamma(X, L)$ is the vector space of sections of the line bundle $L \to X$; this is a graded, finitely generated reduced algebra and defines an affine variety $\tilde{X}$ where $\mathbb{C}^* \times G \times G$ acts. Further, the action of $\mathbb{C}^*$ is attractive, and the corresponding link identifies with X. We say that the pair (X, L) is a *linearized projective $G \times G$-variety*. Let $\tilde{G} = \mathbb{C}^* \times G$. This is a connected reductive group with weight lattice $\tilde{\Lambda} = \mathbb{Z} \times \Lambda$. We may regard $\tilde{X}$ as a $\tilde{G} \times \tilde{G}$-variety, where $\mathbb{C}^* \times \mathbb{C}^*$ acts via its morphism $(t_1, t_2) \mapsto t_1 t_2^{-1}$ to $\mathbb{C}^*$. Now *projective $G \times G$-reductive varieties* may be defined as those linearized projective $G \times G$-varieties (X, L) for which the affine cone $\tilde{X}$ is an affine $\tilde{G} \times \tilde{G}$-reductive variety.

Examples 2.1. 1. Take $G = T$, a complex torus. Then the projective $G \times G$-varieties identify with the projective toric varieties associated with both T and all quotient tori of T.

2. One can require in addition that the projective reductive varieties contain G as a dense open orbit: we then obtain compactifications of algebraic groups or group imbeddings considered in De Concini and Procesi (1983).

Next we recall that projective $G \times G$-reductive varieties are determined combinatorially from certain polytopes just as in the case of toric varieties. Let $\sigma \subseteq \tilde{\Lambda}_{\mathbb{R}} = \mathbb{R} \times \Lambda_{\mathbb{R}}$ be the cone associated with $\tilde{X}$, and put $\delta = \sigma \cap (1 \times \Lambda_{\mathbb{R}})$. Then δ is a lattice polytope in $\Lambda_{\mathbb{R}}$, and σ is the cone over δ. We require that δ satisfies the following conditions:
(i) The relative interior δ^0 meets $\Lambda^+_{\mathbb{R}}$. (ii) The distinct translates $w\delta^0$ $(w \in W)$ are disjoint.

A lattice polytope $\delta \subset \Lambda_{\mathbb{R}}$ satisfying (i) and (ii) is called a *W-admissible polytope*. These classify polarized reductive varieties; we will denote by (X_δ, L_δ) the linearized reductive variety with polytope δ. The closure in X of the orbit of base points, equipped with the restriction of L, is the linearized toric variety with polytope δ. The $G \times G$-orbit closures in X_δ are the X_φ, where $\varphi \subseteq \delta$ is a W-admissible face, which is defined as follows:

Definition 2.2. A face φ is W-admissible if the distinct translates $w\varphi^0$, $(w \in W)$ are disjoint and it satisfies one of the following conditions: (i) its relative interior intersects with the interior of the positive Weyl chamber or (ii) if it lies in one of the walls of the positive Weyl chamber, then no translate $w\varphi$ $(w \in W)$ satisfies the condition in (i).

For each W-admissible face φ of the polytope δ, we let $I(\varphi)$ $(J(\varphi))$ denote the simple roots so that reflections in a plane orthogonal to them leave φ stable (pointwise fixed, respectively). The description of the isotropy group $(G \times G)_x$ as in Brion and Joshua (2004, (5.1.1)) carries over to this projective setting, with Λ_σ (in the affine case) being replaced by the lattice Λ_δ spanned by the *differences* of any two elements of $\Lambda \cap \delta$.

As a consequence, the description of orbits as fibered spaces as in Brion and Joshua (2004, (5.1.2)) carries over as well with σ being replaced by δ. Further, projective imbeddings of a quotient of G by a connected normal subgroup (resp. of G) correspond to W-invariant lattice polytopes (resp. with non-empty interior).

2.1. Combinatorial structure in reductive varieties

One also obtains a *combinatorial description of slices and links* in reductive varieties as in Brion and Joshua (2004, Section 5). Consider a W-admissible polytope $\delta \subset \Lambda_{\mathbb{R}}$, and a W-admissible face $\varphi \subseteq \delta$. These correspond to a linearized reductive variety (X_δ, L_δ) together with a $G \times G$-orbit $\mathcal{O} = \mathcal{O}_\varphi$: the open orbit in $X_\varphi \subseteq X_\delta$. We describe the local structure of X along $\mathcal{O}$ as follows.

Let x be a base point of $\mathcal{O}$; then $(B^- \times B)x$ is open in $(G \times G)x = \mathcal{O}$. Further, it follows from Brion and Joshua (2004, (5.1.1)) that the normalizer P of $(B^- \times B)x$ in $G \times G$ equals $P_J^- \times P_J$, where $J = J(\varphi)$. Since x is fixed by $\operatorname{diag} T$, the Levi subgroup L of P equals $L_J \times L_J$. In this case it is shown in Alexeev and Brion (2004, Lemma 2.8) that $R_u(P_J^-) \times R_u(P_J) \times \Sigma$ is an affine open neighborhood of $(B^- \times B)x$ in X

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

for a locally closed subvariety Σ of X. It is also shown in Alexeev and Brion (2004, Lemma 2.8) that the variety Σ is an affine reductive variety for L_J; one readily checks that the corresponding W_J-admissible cone is generated by the differences $\lambda - \mu$, where $\lambda \in \delta$ and $\mu \in \varphi$.

Now Brion and Joshua (2004, (5.1.1)) once again shows $L_x = G_\varphi \times G_\varphi$, where G_φ is a connected reductive subgroup of G, normalized by T; further, T_φ is a maximal torus of G_φ, so that the weight lattice of G_φ equals Λ/Λ_φ. The set of simple roots of G_φ is $J = J(\varphi)$, with Weyl group $W_J = C_W(\varphi)$; the latter will be denoted by W_φ.

By Alexeev and Brion (2004, Lemma 4.1), the slice $\mathcal{S}_x$ is an affine reductive variety for G_φ. Denote its W_φ-admissible cone by $\sigma = \sigma_\varphi$; this cone is the image in Λ_φ of the cone of Σ. So we may regard σ as the normal cone to δ along its face φ.

To describe the link $\mathbb{P}(\mathcal{S}_x)$, note first that the closed convex cone σ contains no lines. Thus, we may find a linear form f on $\Lambda_{\mathbb{R}}/\Lambda_{\varphi,\mathbb{R}}$ that takes positive values at all non-zero points of σ. We may assume in addition that f takes rational values at all points of Λ/Λ_φ, and is invariant under the normalizer of σ in W_φ. Then by Alexeev and Brion (2004, 3.2, 4.1), f yields a positive $G_\varphi \times G_\varphi$-invariant grading of the algebra of regular functions on $\mathcal{S}_x$. In other words, f defines an attractive $\mathbb{C}^*$-action on $\mathcal{S}_x$ that commutes with the action of $G_\varphi \times G_\varphi$. Now $\mathbb{P}(\mathcal{S}_x)$ is the reductive variety for G_φ associated with the polytope $\sigma \cap (f = n)$, where n is a suitable positive integer. We may regard this polytope as the link of δ along its face φ.

2.2. The closure property of the class of group imbeddings

If X_δ is an embedding of a quotient of G by a connected normal subgroup, then the polytope δ is W-invariant, so that σ_φ is invariant under W_φ. Therefore, $\mathcal{S}_x$ will be an embedding of a quotient of G_φ by a connected normal subgroup. *It follows that the class of imbeddings of connected reductive groups is stable under taking slices and, likewise, links. Therefore, it is possible to restrict to this class of reductive varieties as is done in the present paper*: this also simplifies the algorithm given in the next section.

We end this section by recalling the key theorem from Brion and Joshua (2004): this is the extension of Stanley's algorithm to reductive varieties, and in particular, to group imbeddings.

Theorem 2.3. *Let X be a projective reductive $G\times G$-variety; let $IP_X(t)$ ($IP_{X,x}(t)$) be the Poincaré polynomial for global intersection cohomology (for the stalks of the intersection cohomology sheaves at $x \in X$, respectively). Then*

$$IP_X(t) = \sum_x (1-t^2)^{(2r-r_x)} \frac{P_{G/T}(t)^2}{P_{(G\times G)_x/(T\times T)_x}(t)} t^{2d_x} IP_{X,x}\left(\frac{1}{t}\right) \tag{2.2.1}$$

(sum over representatives of $G \times G$-orbits in X), and

$$IP_{X,x}(t) = IP_{\mathcal{S}_x,x}(t) = \begin{cases} \tau_{\le d_x-1}((1-t^2) IP_{\mathbb{P}(\mathcal{S}_x)}(t)), & d_x \ge 2, \\ 1 & otherwise. \end{cases} \tag{2.2.2}$$

Here T is a maximal torus of G, of dimension r; $(T \times T)_x$ is a maximal torus of $(G \times G)_x$, of dimension r_x; $\mathcal{S}_x$ is a slice at x to $(G \times G)x$, of dimension d_x; $\mathbb{P}(\mathcal{S}_x) = (\mathcal{S}_x - x)/\mathbb{C}^*$ is the corresponding link for an attractive action of the multiplicative group, and $\tau_{\le d_x-1}$ denotes the truncation of a polynomial to degrees $\le d_x - 1$. Moreover, $(1-t^2)^{(2r-r_x)} \frac{P_{G/T}(t)^2}{P_{(G\times G)_x/(T\times T)_x}(t)} = (-1)^{r_x} P_{(G\times G)x}(t)$ where $(G \times G)x$ denotes the $G \times G$-orbit of x.

3. Extension of Stanley's algorithm to projective group imbeddings

In this section we will convert the last theorem into an algorithm for computing the intersection cohomology Betti numbers for all projective reductive varieties using ascending induction on the dimension of the varieties. We will also conclude by considering a few simple examples. For simplicity we will only consider the case of *group imbeddings*: according to Brion and Joshua (2004, Section 5) the corresponding W-admissible polytopes are stable with respect to the action of W and have non-empty interiors. (The last condition means that their dimension is the same as the dimension of $\Lambda_{\mathbb{R}}$.)

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

3.1. Step by step algorithm

In what follows all function names are written in `typewriter` font, while non-literal input variables are written in *italics*.

Initial processing

(1) Enter the reductive group. Certain computations are made automatically, including an enumeration of the positive roots, fundamental roots, Weyl group, Cartan matrix, a basis for the lattice Λ, etc.

(2) Read in the (W-stable) polytope by giving its vertices in the root lattice. Let r = the dimension of root lattice. (Observe this is the dimension of the maximal torus T of G.) The dimension of the corresponding reductive variety is $d = dim(G) = 2l(w_0) + r$ where w_0 is the longest element of the Weyl group.

The recursive or iterative part

For each W-admissible face φ of δ one calls the following steps recursively or iteratively until the polytope in step 5 (namely, δ_φ) reduces to a point.

(3) For each W-admissible face φ of the polytope δ, compute the sets $I(\varphi)$ ($J(\varphi)$) of simple roots so that reflections in a plane orthogonal to them leaves φ stable (pointwise fixed, respectively). The lattices Λ_φ generated by the differences $\lambda - \mu$, λ, μ in the face φ and the quotient Λ/Λ_φ will be considered as appropriate sublattices of Λ. This is possible because there is no torsion present in Λ/Λ_φ. Compute the Weyl subgroup W_φ generated by the simple roots in $J(\varphi)$.

(4) Compute σ_φ = the normal cone to φ in δ. Obtain the polytope $\delta_\varphi = \sigma_\varphi \cap (f = c)$, for some linear functional that is invariant by the normalizer of the normal cone σ_φ in W_φ, positive and rational on the cone σ_φ.

(5) Compute the Poincaré polynomial $P_{(G\times G)x}(t) = (t^2-1)^{dim(\varphi)}\big(\sum_{w\in W^{I(\varphi)}} t^{2l(w)}\big)^2\big(\sum_{w\in W_K(\varphi)} t^{2l(w)}\big)$ where the point x is in the orbit corresponding to φ. $W^{I(\varphi)}$ is the set of coset representatives of minimal length for $W/W_{I(\varphi)}$ where $W_{I(\varphi)}$ denotes the subgroup of W generated by the simple roots in $I(\varphi)$. Moreover, r_x (which is the semi-simple rank of the stabilizer $(G\times G)_x$ at x) is given by $2r - dim(\Lambda_\varphi)$ and $d_x = d - (|\Phi - \Phi_{J(\varphi)}| + dim(\varphi))$ where $\Phi_{J(\varphi)}$ is the subroot system generated by $J(\varphi)$. Substitute this into the formula (2.2.1). Replace W by W_φ (by retaining only those roots that generate W_φ) and the old Λ by Λ/Λ_φ which is represented by a sublattice normal to φ. (Here we retain the ambient lattice Λ as our framework, but record and use the dimension Λ/Λ_φ in subsequent computations.)

(6) Repeat steps 3 through 5 until the polytope in the last step is a point. When the normal polytope δ_φ is a point, the corresponding reductive variety will be also a point, since we are restricting to only group imbeddings.

Of particular importance for us is the manner in which algebraic groups are represented in the package `LiE`. Recall that (see Bourbaki (1975)) every complex connected reductive group is the homomorphic image of a direct product of simple groups and a torus. The above homomorphism has a finite kernel which is contained in the center. Recall that any simple algebraic group is represented by its Lie type: if it is classical, it is of type A_n, B_n, C_n or D_n or of it is exceptional it is of the form E_6, E_7, E_8, F_4 or G_2.

Proposition 3.1. *In Theorem 2.3, it is possible to assume that all the groups G and the various stabilizers are direct products of simple groups and a torus.*

Proof. This is essentially a consequence of the formula proved in Theorem 2.3. First observe that all the stabilizers are connected. We may assume using ascending on the dimension of the group imbedding that the proposition is true for all algebraic groups of dimension strictly smaller than the dimension of G. Next observe that replacing these stabilizers $(G\times G)_x$ by the corresponding reductive groups $(G\times G)_x/R_U((G\times G)_x)$ does not change the cohomology of the corresponding flag variety $(G\times G)_x/(T\times T)_x$. In a similar way we may take the quotient by the central torus without changing the flag variety. Next one may take the simply connected cover of these semi-simple groups: then there will be a surjective homomorphism from these simply connected covers to the original groups

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

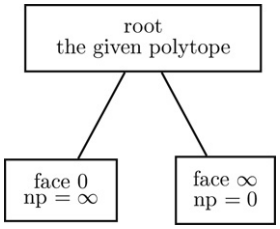
with finite kernel. It is straightforward to see that the flag varieties associated with these groups are isomorphic. This essentially follows from the observation that if $f : G' \to G$ is a surjective homomorphism of (reductive) connected algebraic groups with finite kernel, then the Borel subgroups correspond one to one. (It is known that every Borel subgroup in G is the image of a Borel subgroup in G'. Conversely $f^{-1}(B)$ for any Borel subgroup in G must be parabolic and must contain a Borel subgroup B' of G'. Since parabolic subgroups are connected it follows readily that $f^{-1}(B) = B'$.)

Now consider the various summands that appear in the formula in Theorem 2.3. It is clear from the above observations that the term corresponding to the dense orbit is $(1-t^2)^r P_{G/T}(t)$ for which the above observations apply. For each of the remaining orbits $\mathcal{O}$ containing a point x, one has a term $(1-t^2)^{(2r-r_x)} \frac{P_{G/T}(t)^2}{P_{(G\times G)_x/(T\times T)_x}(t)} t^{2d_x} IP_{X,x}(\frac{1}{t})$ as a summand of $IP_X(t)$. Now the term $IP_{X,x}(t) = IP_{\mathcal{S}_x,x}(t) = \tau_{\leq d_x-1}((1-t^2)IP_{\mathbb{P}(S_x)}(t))$ if $d_x \geq 2$ and $IP_{X,x}(t) = 1$ if $d_x \leq 1$. Since $\mathbb{P}(S_x)$, which is the link, is a group imbedding for a smaller group, we may use ascending induction on the dimension of the algebraic groups whose imbeddings we are considering to complete the proof. □

The implementation-specific details are discussed in detail in the next section. For the rest of this section, we will trace the above algorithm for several examples. See the last section for a computation of the Poincaré polynomials in these examples by our program.

Example 3.2. The most basic example is that of the toric variety $\mathbb{P}^1$. $\mathbb{P}^1$ turns up as the *link* in many of the examples that we consider. In view of these and also to illustrate our techniques we will begin by considering this example.

Observe that $\mathbb{P}^1$ may be viewed as the group imbedding associated with $\mathbb{C}^*$. One has three orbits, namely the dense orbit $\cong \mathbb{C}^*$ and the two fixed points which we will denote as 0 and ∞. Clearly these two are W-admissible since W is trivial. Therefore, the tree diagram associated with the algorithm in 3.1 will be the following:



Therefore, one may compute the Poincaré series for $\mathbb{P}^1$ as $1 - t^2 + 2t^2 = 1 + t^2$.

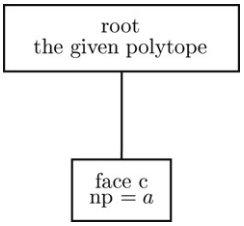
In the above example and the following ones, the tree diagram drawn will be a simplified one convenient for discussing the flow of the algorithm. The actual tree constructed by the program will have several more nodes, needed for the recursive calculation of the Poincaré polynomials.

Next we will consider several non-toric examples beginning with reductive groups of *type* A_1. For type A_n the reductive group will be either PGL_{n+1} or SL_{n+1}. The semi-simple rank of these groups is n, the weight lattice is of dimension n, and the Weyl group W identifies with the symmetric group S_n. Projective group imbeddings of G are given by W-symmetric polytopes in $\Lambda_{\mathbb{R}}$ whose interior meets the positive Weyl chamber $\Lambda^+_{\mathbb{R}}$. Observe that in this case the Poincaré polynomial $P_{G/B}(t) = \sum_{i=0}^{n} t^{2i}$. For the sake of simplicity we will not consider the complication arising from our representation of vectors as coordinate vectors with respect to the basis of simple roots (and not as the actual vectors.)

Example 3.3. We will presently work out in detail the $G = PGL_2$ case considered also in Brion and Joshua (2004). The weight lattice for PGL_2 is one dimensional. We draw this with the positive simple root α along the positive x-axis. Now the Weyl group $W \cong \mathbb{Z}/2\mathbb{Z}$ acts in the obvious manner. Clearly the only choice is a polytope with vertices at $x = \pm k$ for some non-zero k.

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

For convenience, we have represented the polytope as the line segment with faces $a = (-2)$ and $c = (2)$ (Euclidean coordinates). The simple root α is drawn as the vector with coordinate (1). In our algorithm, all vectors are represented by their coordinate vectors with respect to the simple roots: therefore a and c would be denoted as $[-2]$ and $[2]$, respectively. In this case the only W-admissible faces are the entire polytope and the face c. Clearly the face given by the entire polytope corresponds to the dense orbit. There is only one other orbit, namely the closed orbit corresponding to the face c. Now the normal polytope to the face c is a, which is the reflection of c in the y-axis. The following is the tree-diagram for the iterations of the algorithm given in 3.1.

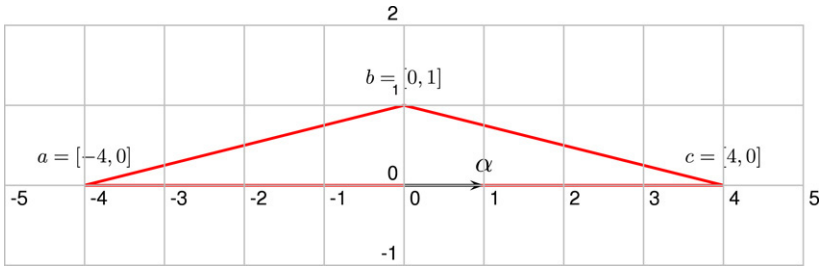


Recall that the normal polytope above corresponds to the link to the corresponding orbit. Observe that for the face $\varphi = c$, the set $J(\varphi = c)$ of simple roots that fix c is empty. Therefore, we make the following observations:

(i) The link associated with c, and which we identified with c, is in fact a *point*. (ii) The Levi subgroup $L_{J(\varphi=c)}$ of PGL_2 identifies with the maximal torus $\mathbb{C}^*$. Moreover, the torus T_φ is trivial since its character group is given by Λ/Λ_φ: see Brion and Joshua (2004, pages 476 and 478). Therefore, the stabilizer corresponding to the closed orbit may be identified with a Borel subgroup of $PGL_2 \times PGL_2$.

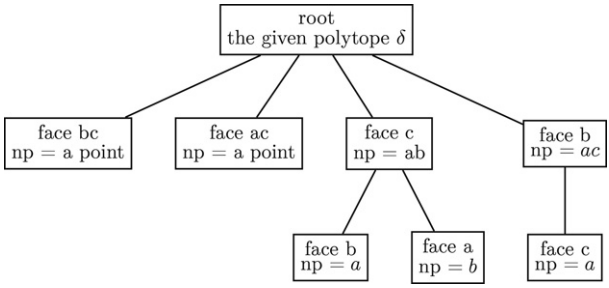
At this point all the data on the right hand side of the formula in Theorem 2.3 (2.2.1) are completely determined. In fact in this case, the reductive variety is none other than $\mathbb{P}^3$ provided with the action of $PGL_2 \times PGL_2$ where the left-factor (right-factor) acts in the obvious manner on the left (right, respectively). This is an example of *wonderful group compactifications* considered in De Concini and Procesi (1983). There are exactly two orbits for this action, namely the open dense orbit $\cong PGL_2$ and the closed orbit isomorphic to $\mathbb{P}^1 \times \mathbb{P}^1$. The slice to the closed orbit is an affine space and the link is a point.

Example 3.4. Next we consider group imbeddings of GL_2. The root lattice for GL_2 is obtained from the lattice of PGL_2 by extending by an orthogonal component. Here the positive simple root (which will be denoted as α) is again drawn along the positive x-axis so that the positive Weyl chamber is the right half-plane. As the polytope δ, we take the triangle with vertices at a, b and c.



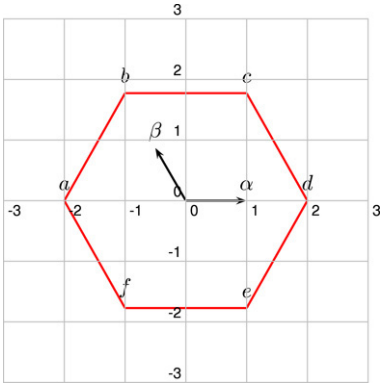
Of the faces, the one-dimensional faces bc and ac, and the vertices b and c are W-admissible. The vertex b is fixed by the action of the Weyl group which is to reflect about the y-axis. The following is the corresponding *tree diagram* as the algorithm explores the various admissible faces iteratively.

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

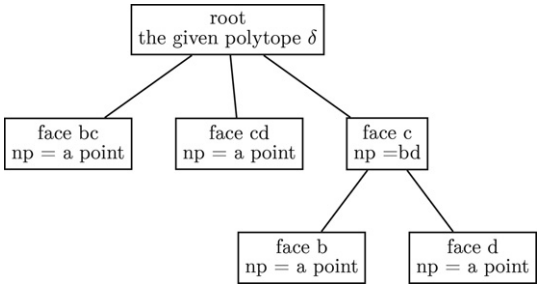


The structure of the various orbits and their transverse slices in this example have been analyzed in detail in Brion and Joshua (2004, p. 480); therefore we skip these details here.

Example 3.5. The next example that we consider is the group $G = PGL_3$ which is of type A_2. Therefore, the root lattice is two dimensional with the simple roots denoted as α and β. We consider the polytope given by the following hexagon: the coordinate vector of the point a with respect to the root vectors α and β is $[-2, 0]$. The action by the Weyl group determines the remaining vertices so that one obtains a polytope symmetric with respect to this action.



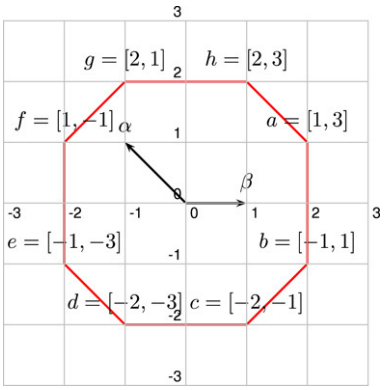
In this case the admissible faces are bc, cd, c. The vertices b and d lie in the walls of the positive Weyl chamber and may be obtained from the vertex c by translating by a suitable Weyl group element. Therefore they are not W-admissible. The *tree* for the iterated calls of our algorithm is as follows. (Here np denotes the normal polytope at a face.)



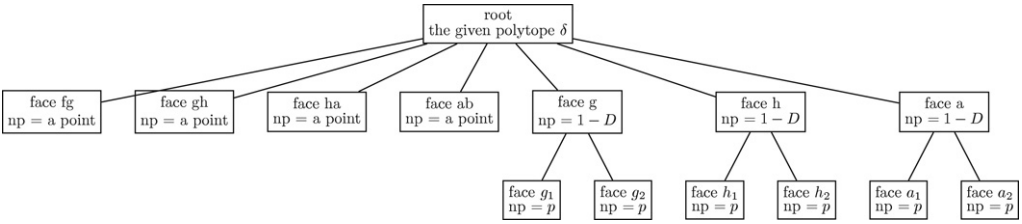
(i) For the face $\varphi = cd$, the sets $J(\varphi)$, $I(\varphi)$ and $K(\varphi)$ are empty. The normal polytope is a point. (ii) For the face $\varphi = bc$, the set $J(\varphi)$ is empty since there are no roots that leave the face bc *pointwise fixed*. However, $I(\varphi) = K(\varphi) = \{\alpha\}$. (iii) Next we consider the face $\varphi = c$. Now $J(\varphi) = I(\varphi) = K(\varphi)$ is empty. The normal polytope is bd.

Example 3.6. Next we consider compactifications of the group SO_5. This is of type B_2. We consider an octagon in the root lattice which looks as follows.

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002



The coordinates given to the vertices are all the components of their coordinate vectors with respect to the simple roots, with the first root denoting the longer root. In this example, the W-admissible faces are gf, gh, ha, ab, g, h and a. The normal polytopes to each of the vertices g, h and a are one dimensional: in the tree below, we have denoted these by $1 - D$. These normal polytopes are not stabilized by any of the given roots so that each of the vertices of these normal polytopes is W-admissible. We denote vertices of the normal polytope associated with the vertex g by g_i, $i = 1, 2$, and similarly for the vertices of the normal polytopes corresponding to h and a. Whenever the normal polytopes are points, we denote these by p.



4. Implementation details

4.1. Input and output

We have chosen to write an outer shell that is entirely in the GAP package; it calls on the package LiE as necessary to perform various calculations that are more Lie theoretic and involve the structure of the weight lattice. The main program also calls on the package polymake to perform certain computations on polytopes. Certain features of LiE and GAP have dictated the choice of several aspects of our program: for example, the representation of polytopes is based on how vectors in the Lie algebra are represented in LiE.

Since the actual simple roots often involve irrational numbers as their Cartesian coordinates, representing the actual simple roots in the usual Cartesian coordinates would essentially involve numerical approximations. To avoid this and other related issues, vectors are represented in LiE by their *coordinate vectors with respect to the basis of simple roots*. For example, if the group is PGL_3 or SL_3, its Lie algebra is of type A_2 and the simple roots are α and β, with α along the x-axis and β in the second quadrant making an angle of $2\pi/3$ radians with α. In this case the vector $\alpha + \beta$ will be represented as `[1, 1]`. The lengths of the simple roots are *normalized* in LiE somewhat differently than the standard conventions: for example, for type A_n, all simple roots have length $\sqrt{2}$ and *not* 1.

In view of Proposition 3.1, one may assume without loss of generality that the algebraic groups that we need to consider are all products of simple groups and a torus. Therefore, whenever the given algebraic group is a product of simple groups (or more generally, semi-simple) any *admissible* (see Definition 2.2) polytope whose vertices lie in the associated weight lattice is allowed. In the case where

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

the algebraic group is the product of a central torus and simple groups (or more generally, a central torus and a semi-simple group), then the weight lattice is *extended* by taking its product with a lattice equal to the lattice of characters of the given central torus. In this case any admissible polytope whose vertices lie in this extended lattice is allowed. Checking for W-admissibility is a rather difficult task and we discuss below (see 4.2) how this is implemented in our program.

One starts the program by first starting GAP and then by reading the script in using the command: `Read("BJAlgorithm.g");`. Next one needs to specify which algebraic group we are considering: this is done by the command: `SetDefaultGroup(`*groupsymbol*`);`, where *groupsymbol* is a string of symbols such as "*AnBmCp*...*Tk*", following how connected reductive algebraic groups may be represented in LiE as a product of simple groups and a torus, ignoring the isogeny considered above. For example, PGL_2 will be denoted as "A1" and GL_2 will be denoted as "A1T1". Next one enters a W-symmetric polytope in the root lattice. Any convex polytope is represented by the list of its vertices. In order to enter the polytope conveniently, we have written a small routine that will take as input a small list of vertices and will produce a list of vertices that can be generated from the given ones using the Weyl group action. This is called `GenPolytope`: this will produce a W-symmetric polytope as a list of its vertices. At this point, all of the necessary global variables are set in order to compute $IP_X(t)$ corresponding to this group and polytope. The function `IP_X` (with no input arguments) accomplishes this computation and stores the result in the global variable `polynomial`.

4.1.1. The initial processing

The reductive group is entered via the command `SetDefaultGroup` using the classification scheme in LiE. Certain information about the group is then generated and the results stored in global variables. Among the initial computations are:

1. The list of positive roots, obtained from LiE via the GAP command `Liepos_roots`, as well as a fundamental system of roots.
2. An enumeration of the Weyl group as ordered lists of simple roots. The current version of LiE does not have a function that does this directly. Instead, we obtain all the group elements generated by the fundamental roots. Each potential new group element is put into canonical form by `Liecanonical` and added to the list if not already present. This process terminates whenever the size of the list equals the previously computed size of the Weyl group (from `LieW_order`. As a by-product, the *length* of each non-identity group element is exactly the length of the list of generators representing that element.
3. A list of matrices representing the actions of group elements on vectors in the weight lattice.
4. The Cartan matrix of the group (for use in computing inner products), obtained from LiE via the `LieCartan` command.
5. A basis for the lattice Λ, which is a direct sum of the weight lattice corresponding to the group and a free lattice of rank equal to the toral dimension.

Since we need to compute the Poincaré series, $P_{G_x/B_x}(t)$ for the various stabilizer subgroups G_x appearing in (2.2.2), we have written a routine that takes any subgroup W' of the Weyl group W of the group G and computes the Poincaré polynomial $P_{W'}(t) = \sum_{w' \in W'} t^{2\ell(w')}$. This is handled by `WeylSubgroup(x)`, where `x` is a (possibly empty) subset of the set of simple roots. (Here, $\ell(w)$ is the length of the Weyl word w).

4.1.2. The recursive part

Here the main effort is in analyzing the given polytope, constructing admissible *normal* polytopes, δ_φ, associated with various W-admissible faces φ and also constructing the corresponding smaller weight lattices associated with the groups G_x.

4.2. Algorithm for deciding on W-admissibility

First, we have written a routine that takes each face and *tests if it is W-admissible* in the sense of Definition 2.2. This is handled by the following function: `W_admissible(`V, ρ`)`, where

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

V is the polytope (as a list of its vertices) and ρ is the list of simple roots for which we are testing "ρ-admissibility", that is, W-admissibility when W is generated by the roots in ρ. The function iterates over the simple components of the original group. Let ρ_r be the subset of ρ corresponding to the roots belonging to the rth simple component. Then for each such ρ_r, determine the position of V with respect to the positive ρ_r-chamber. This is done via the auxiliary function `W_admissible_simple`(V, ρ_r). If V intersects the positive ρ_r-chamber, `W_admissible_simple` returns the integer 2. If V lies entirely within the ρ_r-chamber, the function returns 1, and in all other situations, the function returns 0. Now, after having iterated through all simple component groups, we have a list of integers in {0, 1, 2}. If the list contains a 0, then V is **not** W-admissible; hence `W_admissible` will return `false`. If the list consists entirely of 2's, then V is W-admissible, and `true` is returned. In all other cases, the polytope lies in a chamber wall. This case can be quite subtle, and further tests are necessary.

If the list consists entirely of 1's, then V lies in the chamber walls corresponding to all simple components, and so is not considered W-admissible *unless* no translate of V by a non-trivial group element intersects the positive ρ-chamber. In order to test this condition, the function iterates through all non-identity group elements w generated by ρ and tests the resulting polytope, wV', using the function `W_admissible_simple`(V', ρ). The name of the function is a little deceiving, as it works the same whether the list of roots come from a simple component or a composite of many components. If this function returns 2, then we know that V' intersects the positive ρ-chamber, so that the original V is *not* W-admissible; return `false`. On the other hand, if no such V' intersects the positive chamber, return `true`.

Now, if there is a mixture of 1's and 2's, then V is W-admissible *unless* some translate of V by a group element intersects the positive ρ-chamber. Therefore, the function iterates through all non-identity group elements generated by ρ and tests the resulting polytope, V', using `W_admissible_simple`(V', ρ). If this function returns 2, then we do not consider V to be W-admissible; hence return `false`.

Presently, the function `W_admissible_simple`(V, ρ) will be described. This function makes the following two quick checks before proceeding further:

Quick check 0: If V or ρ is empty, return 2. (That is, in this case, V is ρ-admissible by default.)

Quick check 1: If the lattice $\Lambda_{\mathbb{R}}$ is one dimensional, we can check ρ-admissibility of the polytope V by verifying that at least one point of V lies to the right of 0 — in which case, return 2. If no points lie to the right of 0, but there is at least one point to the left of 0, return 0. The degenerate case of a single point lying at 0 causes the function to return 1.

Next consider the (rational) vector space, $\Lambda_{\mathbb{R}}$, spanned by the generators of the lattice. Let the vectors in ρ be considered part of a basis of $\Lambda_{\mathbb{R}}$, and fill out $\Lambda_{\mathbb{R}}$ with additional vectors orthogonal to those of ρ. Write the points of the polytope V as position vectors in terms of the basis for $\Lambda_{\mathbb{R}}$. Call this new polytope V_0.

Quick check 2: If V_0 is a single point, simply check the ρ-components of the vector for that point. If all are positive, then V_0 lives in the positive ρ-chamber; return 2. If all of the ρ-components are non-negative, with at least one 0, then V_0 lives in a chamber wall; return 1. Otherwise, there is a negative component; return 0.

Quick check 3: Check whether any point of V_0 lies in the positive ρ-chamber (by checking the ρ-coordinates as above). If there is such a point, return 2. Otherwise, further tests are necessary.

If the polytope has not been decided by any of the quick checks, we proceed to invoke the now well known program called `polymake` (see Polymake (1997)). Convert the points (i.e. list of vertices) in V_0 into a `polymake`-style polytope. Use `polymake` functions to obtain the inequalities and equations that define the polytope. Create an intersection of this polytope with the non-negative ρ-chamber by appending the inequalities $v_i \geq 0$ for all ρ-coordinates v_i. Use `polymake` to determine first whether the intersection polytope exists (`polymake_FEASIBLE`). If not, return 0. If so, find the dimension of the intersection polytope (`polymake_DIM`). If this dimension is strictly less, then at best a proper face of V lies in the ρ-chamber wall, so return 0. Next, test whether V lies entirely in a chamber wall by intersecting with the walls and determining the dimension of the resulting polytope. If there is a wall for which the intersection has the same dimension as V, then V lives entirely within the wall; return

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

1. Finally, if none of the preceding tests have returned a value, then we know that V must intersect the positive ρ-chamber non-trivially, and hence we return 2.

The above algorithm decides W-admissibility, since if the polytope is not handled by any of the quick checks, it passes to a question of dimensionality of the intersection polytope defined above.

4.3. Algorithm for constructing the normal cone

Next, for each W-admissible face, we have another routine that finds a set of generators for the associated *normal cone*, and from it the corresponding *normal polytope*. We first discuss the algorithm for determining the normal cone of a W-admissible face φ in a polytope δ. Recall that both δ and φ will be given as a list of vertices. Let v_0 denote the first vertex in the list for φ. Now we will let V_φ denote the span of the vectors $\{v_i - v_0 \mid i > 0, v_i \in \varphi\}$. Next we find the orthogonal complement of V_φ in the ambient space $\Lambda_{\mathbb{R}}$. Let this subspace be denoted as V_0. Here the orthogonality is determined by the inner product defined on the root vectors. Next project all the vectors $w_i - v_0$ onto V_0 where $w_i \in \delta - \varphi$. Note that if the face φ is a point, then the normal cone consists of all vectors $w_j - v_0$, $w_j \neq v_0$. Finally, we return the list of vectors obtained as the above projections along with the chosen vertex v_0. The next step is the construction of the normal polytope.

4.4. Algorithm for computing the normal polytope

The function that performs this is called as follows: `NormalPolytope(`δ, φ, R`)` where δ is the given polytope given as a list of its vertices, φ is a W-admissible face given as a list of its vertices and R is a subset of roots acting on δ. Let R_φ denote the subset of roots that fix the face φ pointwise, and let R_P denote the subset of roots in R_φ that stabilize the normal cone. Next we need to find a hyperplane that is stable with respect to the Weyl subgroup corresponding to R_P and intersecting all normal cone vectors. The hyperplane is determined by its normal vector v, and it suffices to make sure that $\langle w_i, v\rangle > 0$ for all vectors w_i in a generating set for the normal cone. As w_i varies over the generating vectors of the normal cone, this provides a system of inequalities. We further assume that the entries of v are all bounded in $[-1, 1]$. The resulting set of inequalities is solved using `polymake` to obtain a vector v that satisfies the above conditions. Finally one makes this vector invariant under the action of the Weyl subgroup corresponding to R_P by averaging over all its translates by the elements of the same Weyl subgroup. The *normal polytope* is the intersection of the normal cone with the above hyperplane. The above function returns the normal polytope and the set of roots R_P.

4.5. The main algorithm

The main algorithm is a *recursive* implementation of the algorithm discussed above in 3.1. This is implemented as the function `IP_X()`. It first produces a list of W-admissible faces. It examines one face at a time and the recursion proceeds by replacing the given polytope by the normal polytope associated with a fixed face: when the normal polytope turns out to be a point, it terminates the recursive call and starts examining another W-admissible face. When all the W-admissible faces have been explored the algorithm terminates and returns the value of the Poincaré polynomial computed. The global variable `tree` will contain the resulting tree that is constructed as the algorithm proceeds. One may examine the nodes of the tree to determine various intermediate values. For example, it stores the value of the face and the corresponding normal polytope constructed above.

4.6. Scope of the program and its limitations

We have successfully used the program on groups that are tori (up to dimension 4, so far) and on groups of type $A_1, A_1A_1, (A_1)^3, (A_1)^4, A_1T_1, A_1T_2, A_2, A_2A_1, A_2A_2, A_2T_1, A_3, B_2, B_2A_1, B_2A_2, B_2T_1, B_3, C_3$ and G_2. In the toric case, the current tree constructed is exponential in the dimension of the toric variety. For fairly simple toric varieties of dimension 4, the current tree constructed has nodes close to 1000. This is not surprising, since one way to effectively prune the tree is to make use of the Weyl group

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

action. Therefore, in the absence of such Weyl group actions, the tree tends to be large. Despite the tree-pruning effect of the use of W-admissibility, some of the Poincaré polynomials that have been generated for (non-toric) projective reductive varieties are of degrees 20–30.

One limitation of the present program is the exponential tree growth in the toric case. This could be improved by avoiding duplicate computations. Another issue that we have run into is with groups with a relatively large number of roots: for example, we have run into problems for groups of type A_n and B_n, with $n \geq 4$. These are essentially limitations in resources (running time, memory, etc.), and there may be ways to expand our range of effective computation by using parallel computing or storage of previous calculations in tables.

4.7. The script and examples

The files necessary to run the algorithm along with many examples that we have worked out are available as a tar file: http://www.math.ohio-state.edu/~joshua/pub.html/BJA-1-21-08.tar. This includes rpm files of GAP, LiE and polymake that one may install on any i386-PC running Linux. These examples are grouped based on the Lie type of the algebraic group. The programs `LiE` and `polymake` must be installed (in addition to `GAP`). Furthermore, a "liegap" interface is required: for this, the files "lie.g" and "liegap.g" included in the above tar file must be read in at the start of the `GAP` session. We have automated this, so that these files are automatically read in first when the main program "BJAlgorithm.g" is read into `GAP`.

4.8. Installation and running the program

We have successfully installed `GAP`, `LiE` and `polymake` on a PC running Enterprise Linux (or CenTos) 5.1. (One may consult the respective home-pages of these packages for advice on such installations. Alternatively, one may simply install the rpm files of these packages that we have included in the first tar file. These should work on any i386 machine running Linux: we had our program installed on such a machine running Enterprise Linux/Centos 5.1.) Once these packages have been installed, all one needs to do is create a directory where the file BJAlgorithm.etc.tar will be untarred. Now one may read in each of the examples included in the file of examples or one may work out new examples.

4.9. Auxiliary functions of independent interest

In the course of building a program to compute the Poincaré polynomials, we have also written several routines that may be of independent interest. We recall here a few of them.

1. `GenPolytope(L);` — This function produces a list of vertices that form a W-symmetric polytope in the given weight lattice. Here L is a small list of a few vertices.
2. `W_admissible(V, rho);` — This function returns true if the polytope V is rho-admissible, where rho is a list of simple roots.
3. `Find_W_adm_faces(V, rho);` — This function generates all the rho-admissible faces of the given polytope, V.
4. `tree;` — This variable stores the tree that has been constructed by the program for computing the Poincaré polynomial.
5. `Size(tree);` — This is a particularly useful command, as it gives the number of nodes in the tree constructed above.

5. A few sample sessions

The default session

Example 5.1.
```
gap> Read("BJAlgorithm.g");
Loading LiE and polymake modules...
```

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

```
Default Group is A1
Weyl group order = 2
Semisimple rank = 1
Toral dimension = 0
Lattice dimension = 1
Root vectors:
[ 1 ]
Dimension of spherical imbedding = 3

Polytope cleared.
Please input a polytope using 'SetPolytope' or 'GenPolytope'
Please set the default group using:
 SetDefaultGroup(<groupname>)

 Create a W-symmetric polytope by first creating a small list
of vertices <L> and then calling GenPolytope(<L>) or SetPolytope(<L>)
The global variable <polytope> will have the updated list of vertices

For a list of available routines, call:   BJ_Functions()
For a list of global variables, call:  BJ_Variables()
```

Imbedding of SO_5

Example 5.2. `gap> Read("b2_ex.g");`

```
Default Group is B2
Weyl group order = 8
Semisimple rank = 2
Toral dimension = 0
Lattice dimension = 2
Root vectors:
[ 1, 0 ]
[ 0, 1 ]
Dimension of spherical imbedding = 10

Polytope cleared.
Please input a polytope using 'SetPolytope' or 'GenPolytope'
 Octagon generated by [[2,1]] over B2:  B2_P1()
 Octagon generated by [[1,3]] over B2:  B2_P2()
gap> B2_P2();
 Example: Octagon Over B2
polytope updated: [ [ -2, -3 ], [ -2, -1 ], [ -1, -3 ], [ -1, 1 ], [ 1, -1 ],
  [ 1, 3 ], [ 2, 1 ], [ 2, 3 ] ]
Calculating IP_X...
t^20+4*t^18+12*t^16+24*t^14+35*t^12+40*t^10+35*t^8+24*t^6+12*t^4+4*t^2+1
[ [ 1, 0, 4, 0, 12, 0, 24, 0, 35, 0, 40, 0, 35, 0, 24, 0, 12, 0, 4, 0, 1 ], 0
 ]
gap>Size(tree);
17
```

A sample of other higher dimensional examples

Example 5.3. `gap> Read("b2a1_ex.g");`

```
Default Group is B2A1
```

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

. . . (Some output is skipped here.)

```
Dimension of spherical imbedding = 13

Polytope cleared.
Please input a polytope using 'SetPolytope' or 'GenPolytope'
 Polytope generated by [[2,1,0],[0,0,1]] over B2A1:  B2A1_P1()
 Octagonal Prism generated by [[2,1,1]] over B2A1:  B2A1_P2()
gap> B2A1_P2();
 Example: Octagonal Prism over B2A1
polytope updated:
```

. . . (Some output is skipped here.)

```
Calculating IP_X...
t^26+5*t^24+17*t^22+41*t^20+75*t^18+111*t^16+134*t^14+134*t^12+111*t^10
     +75*t^8\
+41*t^6+17*t^4+5*t^2+1
[ [ 1, 0, 5, 0, 17, 0, 41, 0, 75, 0, 111, 0, 134, 0, 134, 0, 111, 0, 75, 0,
      41, 0, 17, 0, 5, 0, 1 ], 0 ]
gap>Size(tree);
85
```

Example 5.4. `gap> Read("g2_ex.g");`

```
Default Group is G2
```

. . . (Some output is skipped here.)

```
Dimension of spherical imbedding = 14

Polytope cleared.
Please input a polytope using 'SetPolytope' or 'GenPolytope'
 Dodecagon generated by [[1,2]] over G2:  G2_P1()
gap> G2_P1();
 Example: Dodecagon Over G2
polytope updated:
```

. . . (Some output is skipped here.)

```
Calculating IP_X...
t^28+6*t^26+20*t^24+40*t^22+60*t^20+80*t^18+99*t^16+108*t^14+99*t^12
     +80*t^10+6\
0*t^8+40*t^6+20*t^4+6*t^2+1
[ [ 1, 0, 6, 0, 20, 0, 40, 0, 60, 0, 80, 0, 99, 0, 108, 0, 99, 0, 80, 0, 60,
      0, 40, 0, 20, 0, 6, 0, 1 ], 0 ]
gap>Size(tree);
27
```

Example 5.5. `gap> Read("a3_ex.g");`

```
Default Group is A3
```

. . . (Some output is skipped here.)

```
Dimension of spherical imbedding = 15

Polytope cleared.
Please input a polytope using 'SetPolytope' or 'GenPolytope'
```

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

```
 Polytope generated by [[2,0,1]] over A3:  A3_P1()
 Polytope generated by [[7,1,2]] over A3:  A3_P2()
gap> A3_P1();
 Example:  14-face polytope over A3
polytope updated:
```

. . . (Some output is skipped here.)

```
Calculating IP_X...
t^30+6*t^28+25*t^26+74*t^24+163*t^22+284*t^20+405*t^18+482*t^16+482*t^14
     +405*t\
^12+284*t^10+163*t^8+74*t^6+25*t^4+6*t^2+1
[ [ 1, 0, 6, 0, 25, 0, 74, 0, 163, 0, 284, 0, 405, 0, 482, 0, 482, 0, 405, 0,
      284, 0, 163, 0, 74, 0, 25, 0, 6, 0, 1 ], 0 ]
gap> Size(tree);
132
```

Example 5.6. `gap> Read("c3_ex.g");`

```
Default Group is C3
```

. . . (Some output is skipped here.)

```
Dimension of spherical imbedding = 21

Polytope cleared.
Please input a polytope using 'SetPolytope' or 'GenPolytope'
 Polytope generated by [[5,3,1]] over C3:  C3_P1()
gap> C3_P1();
 Example:  over C3
polytope updated:
```

. . . (Some output is skipped here.)

```
Calculating IP_X...
t^42+13*t^40+66*t^38+210*t^36+503*t^34+983*t^32+1649*t^30+2441*t^28
     +3240*t^26+3900\
*t^24+4274*t^22+4274*t^20+3900*t^18+3240*t^16+2441*t^14+1649*t^12
     +983*t^10+503*t^8\
+210*t^6+66*t^4+13*t^2+1
[ [ 1, 0, 13, 0, 66, 0, 210, 0, 503, 0, 983, 0, 1649, 0, 2441, 0, 3240, 0, 3900,
     0, 4274, 0, 4274, 0, 3900, 0, 3240, 0, 2441, 0, 1649, 0, 983, 0, 503, 0,
      210, 0, 66, 0, 13, 0, 1 ], 0 ]
gap> Size(tree);
377
```

Acknowledgements

Both authors were supported by a grant from the NSA. The first author also thanks the MPI(Bonn) and IHES for support.

References

Alexeev, V., Brion, M., 2004. Stable reductive varieties. I, affine varieties. Invent. Math. 157 (2), 227–274.
Alexeev, V., Brion, M., 2004. Stable reductive varieties: Projective varieties. Adv. Math. 184 (2), 380–408.

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002

Beilinson, A., Bernstein, J., Deligne, P., 1982. Faisceaux Pervers, vol. 100. Société Mathématiques de France, Asterisque.
Bourbaki, N., 1975 Groupes et Algébres de Lie. Hermann, Paris, (Chapters 7& 8).
Brion, M., Joshua, R., 2001. Vanishing of odd dimensional intersection cohomology II. Math. Ann. 321 (2), 399–437.
Brion, M., Joshua, R., 2004. Intersection cohomology of reductive varieties. J. European Math. Soc. 6, 465–481.
Denef, J., Loeser, F., 1991. Weights,exponential sums, intersection cohomology and Newton polyhedra. Invent. Math. 106, 275–294.
De Concini, C., Procesi, C., 1983. Complete symmetric varieties. In: Invariant Theory (Montecatini, 1982). In: Lect. Notes in Math., vol. 996. Springer.
Fieseler, K.H., 1991. Rational intersection cohomology of projective toric varieties. J. Reine Angew. Math. 413, 88–98.
Joshua, R., 1987. Vanishing of odd dimensional intersection cohomology. Math. Z. 195, 239–253.
Polymake, 1997. www.math.tu-berlin.de/polymake/.
Stanley, R., 1987. Generalized H-vectors, intersection cohomology of toric varieties and related results. Adv. Stud. Pure Math. 11, 187–213.
van Leeuwen, M.A.A., Cohen, A.M., Lisser, B., 1996. LiE manual. See http://www.cwi.nl/~maavl/liE.
The GAP Group, 1986. The GAP package. See http://www.gap-system.org/.

Please cite this article in press as: Joshua, R., Van Ault, S., Implementation of Stanley's algorithm for projective group imbeddings. Journal of Symbolic Computation (2009), doi:10.1016/j.jsc.2008.11.002