GITFANLIB — A PACKAGE ON GIT-FANS

(VERSION 0.1)

SIMON KEICHER

1. Introduction

This is the documentation of the package `gitfanlib`, which implements the algorithms presented in [3]. More precisely: given an affine variety $X = V(\mathbb{K}^r; \mathfrak{a})$ with the action of a torus H that is given by a grading matrix Q, our packages provides functions to compute

- the GIT-fan $\Lambda(\mathfrak{a}, Q)$ of the H-action on X,
- GIT-chambers $\lambda(w)$ corresponding to weight vectors w,
- $(\mathbb{K}^*)^r$-orbits intersecting X.

See [1, 3] for an explanation of these notions and further references.

At the moment, `gitfanlib` is available for `Maple` 8 or newer; it uses `convex` [2]. An implementation in other software-systems is in preparation.

This package comes with absolutely no warranty.

Contents

2. Using the software

Download from [4] the files `gitfanlib.lib` and `gitfanlib.ind` into a directory where `Maple` can find them, e.g. in the `lib/` directory of your `Maple`-folder. The `convex`-package [2] must be installed as well. In a new `Maple`-session you can load these packages as usual:

```
> with(convex);
> with(gitfanlib);
```

...

gitfanlib, v0.1, May 2012

A package on GIT-fans.

...

Date: May 2012.

3. Procedures

We always assume that the H-action on X is given by a grading matrix Q and that X is given by equations, i.e. by a list of generators.

In the examples, we will use the data for $X = G(2,5)$ with its maximal torus action, i.e.

```
> Q := linalg[matrix]([[1, 1, 1, 1, 0, 0, 0, 0, 0, 0], [1, 0, 0, 0,
1, 1, 1, 0, 0, 0], [0, 1, 0, 0, 1, 0, 0, 1, 1, 0], [0, 0, 1, 0, 0, 1,
0, 1, 0, 1], [0, 0, 0, 1, 0, 0, 1, 0, 1, 1]])
```

$$Q := \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

```
> RL := [T[5]*T[10]-T[6]*T[9]+T[7]*T[8],
T[1]*T[10]-T[3]*T[7]+T[4]*T[6], T[1]*T[9]-T[2]*T[7]+T[4]*T[5],
T[1]*T[8]-T[2]*T[6]+T[3]*T[5], T[2]*T[10]-T[3]*T[9]+T[4]*T[8]];
```

$$\begin{aligned} RL := [&T_5T_{10} - T_6T_9 + T_7T_8, T_1T_{10} - T_3T_7 + T_4T_6, \\ &T_1T_9 - T_2T_7 + T_4T_5, T_1T_8 - T_2T_6 + T_3T_5, \\ &T_2T_{10} - T_3T_9 + T_4T_8] \end{aligned}$$

```
> TT := vars(10);
```

$$TT := [T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}]$$

3.0.1. *afaces.* Returns all $\mathfrak{a}$-faces.

Input:

- A list `RL` , a list of vars `TT` .
- optional: `'radical'` or `'pushed'`; compare 3.0.2.

Output: A set of `sets`.

Example: `> af := afaces(RL, TT, 'pushed');`

$$af := \{\{\}, \{1,2,5,6,8\}, \{1,3,5,6,8\}, \{2,3,5,6,8\}, \{5,6,8\}, \ldots\}$$

3.0.2. *choosealgos.* Globally sets a variable that tells our package which algorithms should be preferred.

- $\mathfrak{a}$-faces: `'radical'`: use [3, Rem. 3.1], `'pushed'`: use [3, Alg. 3.7].
- GIT-chambers: `'naive'`: use [3, Alg. 2.7], `'walls'`: use [3, Alg. 2.4].

Input: Optional: `'radical'` or `'pushed'` and `'naive'` or `'walls'`

Output: Nothing; prints a string.

Example: `> choosealgos('radical', 'naive');`

```
Currently enabled algorithms:
* a-faces: 'radical',
* gitchamber: 'naive'
```

3.0.3. *isaface.* Tests whether the given face $\gamma_0 \preceq \mathbb{Q}^r_{\geq 0}$ is an $\mathfrak{a}$-face.

Input:

- A set of indices `gam0`, a list `RL` , a list of vars `TT` .
- Optional: `'radical'` or `'pushed'`; compare 3.0.2.

Output: A boolean.

Example: Let `RL` and `TT` be as above.

```
> isaface(seq(1 ..  10), RL, TT, 'naive');
```

$$true$$

3.0.4. *gitchamber.* Computes a single GIT-chamber.

Input:

- A matrix `Q`, a list of integers `w`, a list of polynomials `RL` and a list of variables `TT`.
- Optional: `'naive'` or `'walls'`: choose specific algorithms of [3] as in 3.0.2.

Output: A `CONE`.

Example: Let `Q`, `RL` and `TT` be as above.

```
> w := [1,1,1,1,1];
```

$$w := [1, 1, 1, 1, 1]$$

```
> lambdaw := gitchamber(Q, w, RL, TT, 'walls');
```

$$lambdaw := CONE(5, 5, 0, 10, 10)$$

3.0.5. *gitfan.* Returns the GIT-fan of the H-action on $V(\mathfrak{a})$. The generators of $\mathfrak{a}$ are given in a list `RL` and a list of variables `TT`. The third parameter is the grading matrix Q.

Input:

- A matrix `Q`, a list of polynomials `RL` and a list of variables `TT`. If only `Q` is specified, GKZ(Q) is returned.
- Optional: `'radical'`, `'naive'`, `'pushed'` or `'walls'`: choose specific algorithms of [3] as in 3.0.2.
- Optional: `'fan'`; returns a `FAN` instead of a list of maximal `CONE`s.

Output: A list of CONEs or a `FAN`..

Example: Let `Q`, `RL` and `TT` be as above.

```
> GIT := gitfan(Q, RL, TT, 'fan');
```

$$GIT := FAN(5, 0, [0, 0, 0, 0, 76])$$

```
> GKZ := gitfan(Q, 'fan');
```

$$GKZ := FAN(5, 0, [0, 0, 0, 0, 345])$$

3.0.6. *vars.* Returns the list `[T[1],...,T[r]]`. These can be used as variables if `T` hasn't been assigned yet.

Input: An integer r

Output: A list of symbols

Example: `> TT := vars(3);`

$$TT := [T_1, T_2, T_3]$$

References

[1] I. Arzhantsev, U. Derenthal, J. Hausen, A. Laface: *Cox rings*. Preprint, `arXiv:1003.4229`; see also the authors' webpages.
[2] M. Franz: *Convex – a Maple package for convex geometry*. (Available at `http://www.math.uwo.ca/~mfranz/convex/`)
[3] S. Keicher: *Computing the GIT-fan*. Preprint, `arXiv:1205.4204`.
[4] S. Keicher: *gitfanlib – a package for GIT-fans*. (Available at `http://www.mathematik.uni-tuebingen.de/~keicher/gitfanlib/`).