Gfan version 0.5: A User's Manual

Anders Nedergaard Jensen *

January 25, 2011

Abstract

Gfan is a software package for computing Gröbner fans and tropical varieties. These are polyhedral fans associated to polynomial ideals. The maximal cones of a Gröbner fan are in bijection with the marked reduced Gröbner bases of its defining ideal. The software computes all marked reduced Gröbner bases of an ideal. Their union is a universal Gröbner basis. The tropical variety of a polynomial ideal is a certain subcomplex of the Gröbner fan. Gfan contains algorithms for computing this complex for general ideals and specialized algorithms for tropical curves, tropical hypersurfaces and tropical varieties of prime ideals. In addition to the above core functions the package contains many tools which are useful in the study of Gröbner bases, initial ideals and tropical geometry. The full list of commands can be found in Appendix B. For ordinary Gröbner basis computations Gfan is not competitive in speed compared to programs such as CoCoA, Singular and Macaulay2.

Contents

*Research partially supported by the Faculty of Science, University of Aarhus, Danish Research Training Council (Forskeruddannelsesrådet, FUR) , Institute for Operations Research ETH, grants DMS 0222452 and DMS 0100141 of the U.S. National Science Foundation and the American Institute of Mathematics.

1 Introduction

Gfan is a software package for computing *Gröbner fans* [18] and *tropical varieties* [20] of polynomial ideals. It is an implementation of the algorithms appearing in [9] and [5]. These two papers are joint work with Tristram Bogart, Komei Fukuda, David Speyer, Bernd Sturmfels and Rekha Thomas. A combined presentation can be found in [15]. For toric and lattice ideals, Gröbner fan programs already existed: TiGERS [12] and CaTS [13]. Gfan works on any ideal in $\mathbb{Q}[x_1, \ldots, x_n]$.

Gfan is based on Buchberger's algorithm [6] and the local basis change procedure [7]. For traversal of Gröbner fans the simplex method, the reverse search technique [3] and symmetry exploiting algorithms are used. This allows enumeration of fans with millions of cones. For tropical computations these methods have been developed further.

Gfan has been used for studying the structure of the Gröbner fan. Among the new results is an example of a Gröbner fan which is not the normal fan of a polyhedron [16].

The software is intended to be run in a UNIX style environment. In particular, the software works on GNU/Linux and on Mac OS X (with some effort). Gfan uses the GNU multi-precision arithmetic library [11] and cddlib [8] for doing exact arithmetics and solving linear programming problems, respectively. A new feature of version 0.4 is the possibility to use the SoPlex [23] linear programming solver which does its computations in floating point arithmetics. Gfan verifies LP certificates in exact arithmetics and falls back on cddlib in case of a rounding error.

The first section of this manual is a very short introduction to Gröbner fans and algorithms for computing them. The second section describes the installation procedure of the software and the third gives some examples of how to use it. Section 4 explains how Gfan can be used for computing tropical varieties, prevarieties and tropical bases. More details on the data formats and programs are given in Appendix A and B.

Note for the reader: As opposed to scientific journals the World Wide Web has the advantage that its contents can be changed after publication. If you have suggestions for improvements of this manual do not hesitate to let me know. Suggestions for the installation instructions are of particular interest since I only have access to / experience with a limited number of computer systems.

Acknowledgments: The first version of this software was written in the fall 2003 during the authors visit to the Institute for Operations Research, ETH Zürich. Many features have been added since then. Rekha Thomas and Komei Fukuda have been involved in the development of the Gröbner fan algorithms, see the joint paper [9]. The tropical algorithms were developed in the joint paper [5] with

Tristram Bogart, David Speyer, Bernd Sturmfels and Rekha Thomas. The author is thankful to the following people and institutions for supporting the research: Komei Fukuda and Hans-Jakob Lüthi (Institute for Operations Research, ETH Zürich), Douglas Lind and Rekha Thomas (University of Washington, Seattle) and the American Institute of Mathematics. In recent years the research has also been supported by University of Aarhus, University of Minnesota, TU-Berlin and the German Research Foundation (DFG) through the institutional strategy of Georg-August-Universität Göttingen. The author would also like to thank his advisor Niels Lauritzen and the many people who have been testing, been using and helped improving the software.

1.1 The Gröbner fan of an ideal

The Gröbner fan of an ideal $I \subseteq k[x_1, \ldots, x_n]$ in a polynomial ring over a field k is a polyhedral complex consisting of cones in $\mathbb{R}^n$. We provide a short definition and refer the reader to the papers mentioned above for details.

Definition 1.1 Let $\omega \in \mathbb{R}^n$ and $a \in \mathbb{N}^n$. We define $x^a := x_1^{a_1} \cdots x_n^{a_n}$. The *$\omega$-weight* of αx^a with $\alpha \in k \setminus \{0\}$ is $\omega \cdot a$. For $f \in k[x_1, \ldots, x_n]$ we define its *initial form* $\mathrm{in}_\omega(f)$ to be the sum of all terms in f with maximal ω-weight. For an ideal $I \subseteq k[x_1, \ldots, x_n]$ we define the *initial ideal* to be $\mathrm{in}_\omega(I) := \langle \mathrm{in}_\omega(f) : f \in I \rangle$.

Notice that initial ideals might not be monomial ideals. If for some $\omega \in \mathbb{R}^n_{>0}$ we have $\mathrm{in}_\omega(I) = I$ then we say that I is *homogeneous* in the ω-grading. We now fix the ideal $I \subseteq k[x_1, \ldots, x_n]$ and consider the equivalence relation:

$$u \sim v \Leftrightarrow \mathrm{in}_u(I) = \mathrm{in}_v(I)$$

on vectors $u, v \in \mathbb{R}^n$. If I is homogeneous then any equivalence class contains a positive vector. Any equivalence class containing a positive vector is convex. Moreover, its closure is a polyhedral cone. We use the notation

$$C_\omega(I) := \overline{\{u \in \mathbb{R}^n : \mathrm{in}_u(I) = \mathrm{in}_\omega(I)\}}$$

to denote the closure of the equivalence class containing ω.

Definition 1.2 [9, Definition 2.8] Let $I \subseteq k[x_1, \ldots, x_n]$ be an ideal. The *Gröbner fan* of I is the collection of cones $C_\omega(I)$ where $\omega \in \mathbb{R}^n_{>0}$ together with all their non-empty faces.

Any cone in the Gröbner fan is called a *Gröbner cone*. The relative interior of any Gröbner cone is an equivalence class. The equivalence class containing 0 is a subspace of $\mathbb{R}^n$ called the *homogeneity space* of I. The Gröbner fan is a polyhedral fan; see [21] or [9]. The *support* of the Gröbner fan i.e. the union of its cones is called the *Gröbner region* of I. If I is homogeneous then the

Gröbner region is $\mathbb{R}^n$ and, moreover, the Gröbner fan is the normal fan of the *state polytope* of I; see [21] for a construction of this polytope. The *lineality space* of a polyhedral cone is defined as the largest subspace contained in the cone. The common lineality space of all cones in the Gröbner fan equals the homogeneity space of I.

Remark 1.3 Definition 1.2 was chosen since it gives the nicest Gröbner cones. In general our Gröbner fan does not coincide with the "restricted" Gröbner fan nor the "extended" Gröbner fan defined in [18]. The common refinement (i.e. "intersection") of $\mathbb{R}^n_{\geq 0}$ and our Gröbner fan is the restricted Gröbner fan. For homogeneous ideals our definition coincides with [21, page 13] (which only contains a definition for homogeneous ideals).

1.2 Gröbner bases

Given a *term order* $\prec$ the *initial term* $\text{in}_\prec(f)$ of a polynomial f is defined and, analogously to the ω-initial ideal above, so is the *initial ideal* $\text{in}_\prec(I)$ of an ideal I. We remind the reader that given generators for and ideal $I \subseteq k[x_1, \ldots, x_n]$ and a term order $\prec$ Buchberger's Algorithm produces a *reduced* Gröbner basis $\mathcal{G}_\prec(I)$. This basis is unique. It is useful to introduce the notion of a *marked* polynomial and a *marked* reduced Gröbner basis. A polynomial is marked if one of its terms has been distinguished. When writing such a polynomial we may either underline the distinguished term or we may by convention write the distinguished term as the first one listed. Gfan uses this second convention. A Gröbner basis $\mathcal{G}_\prec(I)$ is marked if the initial term $\text{in}_\prec(f)$ of every polynomial $f \in \mathcal{G}_\prec(I)$ has been marked i.e. distinguished.

Example 1.4 The polynomial ideal $I = \langle x + y \rangle \subseteq \mathbb{Q}[x, y]$ has two marked reduced Gröbner bases: $\{\underline{x} + y\}$ and $\{x + \underline{y}\}$. Gfan would write these Gröbner bases as $\{x + y\}$ and $\{y + x\}$.

By definition of Gröbner bases the initial ideal $\text{in}_\prec(I)$ is easily read off from the marked (reduced) Gröbner basis $\mathcal{G}_\prec(I)$, namely, it is generated by the marked terms. In fact, for $I \subseteq k[x_1, \ldots, x_n]$ fixed the follow three finite sets are in bijection:

- The set of marked reduced Gröbner bases for I.
- The set of monomial initial ideals $\text{in}_\prec(I)$ with respect to term orders.
- The set of n-dimensional Gröbner cones in the Gröbner fan of I.

The map from the first set to the second set has already been described. A monomial ideal $\text{in}_\prec(I)$ in the second set is mapped to $\overline{\{v \in \mathbb{R}^n : \text{in}_v(I) = \text{in}_\prec(I)\}}$ in the third set. Going from the first set to the third is easy, namely the inequalities can be read off from the exponents of the marked reduced Gröbner basis. Thus a useful way to represent the Gröbner fan of an ideal is by the set of its marked reduced Gröbner bases.

1.3 Algorithmic background

We briefly describe the algorithms implemented in Gfan for computing Gröbner fans. The algorithms are divided into two parts, the local algorithms and the global algorithms. For more details we refer to [9] and [14].

1.3.1 Local computations

There are two local computations that need to be done:

- Given a full-dimensional Gröbner cone by its reduced Gröbner basis, we need to find its facets. To be precise we need to find a normal for each facet.
- Given a full-dimensional Gröbner cone represented by its reduced Gröbner basis and a normal for one of its facets we need to compute the other full-dimensional cone having this facet as a facet (if one exists). Again, the computed cone should be represented by a reduced Gröbner basis.

To do the first computation we need the following theorem telling us how to read of the cone inequalities from the reduced Gröbner basis:

Theorem 1.5 *Let $\mathcal{G}_\prec(I)$ be a reduced Gröbner basis. For any vector $u \in \mathbb{R}^n$*

$$\mathrm{in}_u(I) = \mathrm{in}_\prec(I) \Leftrightarrow \forall g \in \mathcal{G}_\prec(I) : \mathrm{in}_u(g) = \mathrm{in}_\prec(g)$$

Each g introduces a set of strict linear inequalities on u. By making these inequalities non-strict we get a description of the closed Gröbner cone of $\mathcal{G}_\prec(I)$. This gives us a list of possible facet normals of the cone. Linear programming techniques are now applied to find the true set of normals among these.

Suppose we know a reduced Gröbner basis $\mathcal{G}_\prec(I)$ and a normal of one of its facets. If ω is a vector in the relative interior of the facet we can compute a Gröbner basis of $\mathrm{in}_\omega(I)$ with respect to $\prec$ by picking out a certain subset of the terms in $\mathcal{G}_\prec(I)$, see [21, Corollary 1.9]. The initial ideal $\mathrm{in}_\omega(I)$ has at most two reduced Gröbner bases since it is homogeneous with respect to any grading given by vectors in the $n-1$ dimensional subspace spanned by the facet. The other Gröbner basis of $\mathrm{in}_\omega(I)$ can be computed using a term order represented by the outer normal of the facet. A lifting step will take the Gröbner basis for $\mathrm{in}_\omega(I)$ to a Gröbner basis for I representing the neighbouring cone. See [21, Subroutine 3.7]. The method described above is the local change procedure due to [7]. The procedure simplifies in our case since:

- We only walk through facets. Thus, the ideal $\text{in}_\omega(I)$ has at most two reduced Gröbner bases.
- We know the facet normal. Thus, there is no reason for computing ω.

1.3.2 Global computations

We define the graph G whose set of vertices consists of all reduced Gröbner bases of I with two bases being connected if their cones share a common facet containing a strictly positive vector. With the two subroutines in the previous section it is easy to do a traditional vertex enumeration of G starting from some reduced Gröbner basis. However, for such algorithm to work it would need to store the boundary of the already enumerated vertices to guarantee that we do not enumerate the same vertex more than ones. For a planar graph this might not seem too bad but as the dimension grows the boundary can contain a huge number of elements. Storing these elements would require a lot of memory and sometimes more memory than the size of the computers RAM which would cause the computation to slow down.

A better way to do the enumeration is by the reverse search strategy [3]. If there is an easy rule for orienting the edges of a graph so that it has a unique sink and no cycles it is also easy to find a spanning tree for the graph. The reverse search will traverse this spanning tree. The method works well for enumerating vertices of polytopes since an orientation of the edges with respect to a generic vector will have a unique sink and no cycles. A proof in [9] shows that a similar orientation orienting G with respect to a term order will also give an acyclic orientation with a unique sink and thus allow enumeration by reverse search. Reverse search is the default enumeration method in Gfan .

If the ideal is symmetric we may want to do the Gröbner basis enumeration up to symmetry. For example the ideal $I = \langle a-b \rangle \subseteq k[a,b]$ is invariant under the exchange of a and b. The ideal has two marked Gröbner bases $\{\underline{a}-b\}$ and $\{\underline{b}-a\}$, each defining a full dimensional Gröbner cone in $\mathbb{R}^2$. Up to symmetry they are equal. We only want to compute one of them. In general $I \subseteq k[x_1, \ldots, x_n]$ is invariant under all permutations of some subgroup $\mathbf{G} \subseteq S_n$. Applying a permutation in $\mathbf{G}$ to a marked reduced Gröbner basis of I we get another marked reduced Gröbner basis of I. Hence, $\mathbf{G}$ acts on the set of marked reduced Gröbner bases of I. We wish to compute only one representative for each orbit. We apply techniques similar to the ones used in [19] for computing regular triangulations of point configurations up to symmetry. Often the number of orbits is much smaller than the number of reduced Gröbner bases and we save a lot of time by not computing them all.

2 Installation

If you are using MacOS the easiest way to install Gfan is to use precompiled executables: go to the Gfan webpage, go to the binaries.html subpage, and follow the instructions there.

If you are using Linux the following might work

```
sudo apt-get install gfan
```

or

```
sudo emerge gfan
```

depending on your distribution and package manager. If you succeed, it is good to know which version was installed. Run

```
gfan _version
```

Should this command fail, then you are using an old version of gfan.

The rest of this section eplains how to install Gfan by compiling it from source on a Linux/Unix-like system with a modern version of gcc. Gfan has been compiled successfully with gcc version 4.1.1. **Two libraries are needed in order to compile Gfan : cddlib and gmp.** Users of Microsoft Windows may be able to use these installation instructions if they first install Cygwin. A new feature in Gfan version 0.4 is the possibility to link to the SoPlex [23] library. This does not add to the functionality of Gfan but improves speed of the polyhedral computations. In an attempt to keep the installation instructions simple, instructions for how to use SoPlex are given in a separate section, Subsection 2.4. If you are a lucky Linux user it will suffice to follow the red part of these instructions.

2.1 Installation of the gmp library

GMP stands for GNU Multi Precision arithmetic library. This library must be installed on your system before you can install cddlib and gfan. **On ~~most~~ some GNU/Linux systems the library is already installed.** If your system does not already have gmp installed (which is the case if you have a usual Mac OS X installation) follow the directions in this section.

IF YOU ARE USING Mac OS X AND YOU ARE NOT AN EXPERT FOLLOW THE INSTRUCTIONS IN SECTION 2.1.1 INSTEAD.

Make a new directory and download `gmp-4.2.2.tar.gz` from

`http://gmplib.org/`

for example by typing

```
cd ~
mkdir tempdir
cd tempdir
wget http://ftp.sunet.se/pub/gnu/gmp/gmp-4.2.2.tar.gz
```

Extract the file and go to the thereby created directory:

```
tar -xzvf gmp-4.2.2.tar.gz
cd gmp-4.2.2
```

Run the configure script and specify the installation directory:

```
./configure --prefix=$HOME/gfan/gmp
```

The above line specifies the installation directory which in this case will be the folder `gfan/gmp` in your home directory. If you already have a directory by that name its content may be destroyed by the subsequent commands.

Compile the gmp library and install it:

```
make
make install
```

Finally, a very important step when working with gmp: Let the program perform a self-test:

```
make check
```

The gmp installation is now complete. The gmp files can be found in your home directory under `gfan/gmp`.

2.1.1 Installing the gmp library on Mac OS X using fink

Current versions of Mac OS X and the gmp library have a compatibility problem causing gmp to be compiled with errors if compiled without modifications. There exist packages of gmp for Mac OS X on the internet which have been compiled incorrectly. We recommend that Mac OS users use the packages provided by fink.

Install fink by following the instructions given on the page
`http://www.finkproject.org/download/index.php?phpLang=en`
Having installed fink now simply type

```
fink install gmp
```

The gmp library is now installed in the directory `/sw`.

2.2 Installation of the cddlib library

Cddlib [8] is a library for doing exact polyhedral computations, including solving linear programming problems. Gfan can be compiled with cddlib version 094. Older versions of cddlib will not work with Gfan version 0.2 or later. Notice that cddlib itself needs gmp to compile. We give instructions on how to install cddlib.

Make a directory for the compilation process if you did not do that already:

```
cd ~
mkdir tempdir
cd tempdir
```

Download the file `cddlib-094f.tar.gz` from

`http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html`

into that directory. Decompress the file and change directory to the directory being created:

```
tar -xzvf cddlib-094f.tar.gz
cd cddlib-094f
```

Run the configure script. Here you have the chance of telling cddlib where to find gmp and where to install itself.

```
./configure --prefix="$HOME/gfan/cddlib"
      CFLAGS="-I$HOME/gfan/gmp/include -L$HOME/gfan/gmp/lib"
```

(On a single line). The above options say that cddlib should be installed in your home directory under `gfan/cddlib` and where to look for gmp. If gmp was installed by fink (see Section 2.1.1) you should run

```
./configure --prefix="$HOME/gfan/cddlib"
      CFLAGS="-I/sw/include -L/sw/lib"
```

instead. **If gmp was already installed on your system in its default location run**

```
./configure --prefix="$HOME/gfan/cddlib"
```

The content of `gfan/cddlib` might be destroyed by the subsequent commands. Compile and install cddlib:

```
make
make install
```

You can now find the installed cddlib library files in your home directory under `gfan/cddlib`.
If you had super user access you could also just have run

```
./configure
```

when you configured cddlib. This would cause cddlib to be installed in its default place.

2.3 Gfan installation

Download the file `gfan0.5.tar.gz` from the Gfan homepage located at:

`http://www.math.tu-berlin.de/~jensen/software/gfan/gfan.html`

to your folder tempdir. Extract the file and enter the new directory by typing

```
cd ~
cd tempdir
tar -xzvf gfan0.5.tar.gz
cd gfan0.5
```

Gfan does not have a configure script, so you tell Gfan where to find gmp and cdd when you compile the program. For example you should type

```
make
```

or

```
make cddpath=$HOME/gfan/cddlib
```

or

```
make cddpath=$HOME/gfan/cddlib gmppath=$HOME/gfan/gmp
```

or

```
make cddpath=$HOME/gfan/cddlib gmppath=/sw
```

depending on where you installed the libraries to compile the program.

The final step is to install the compiled program. Type

```
make PREFIX=$HOME/gfan install
```

or

```
make install
```

depending on where you want Gfan installed. (The second line attempts to install it in `/usr/local` by default). If you chose to install in the directory `gfan` in your home folder you will now find the file `gfan` in the subdirectory `gfan/bin` of your home folder together with a set of symbolic links, for example `gfan_buchberger`. You can go to the subdirectory and type `./gfan --help` and `./gfan_buchberger --help` in the shell to test them. Or you can ask Gfan to compute the reduced Gröbner bases of an ideal by typing

```
./gfan_bases
```

followed by, for example,

```
Q[a,b,c]
{a^3+b^2c-a,c^2-2/3b}
```

Remark 2.1 If for some reason you did get `gfan` compiled but did not get the symbolic links made like `gfan_buchberger` you can still run that program by typing `gfan _buchberger` instead of `gfan_buchberger`.

2.4 SoPlex (for the advanced user only)

Linking Gfan to SoPlex can lead to huge performance improvements. Notice however, that the strict license of SoPlex propagates through the software to your paper, requiring that you cite SoPlex appropriately if you choose to publish results based on SoPlex. Furthermore, with the standard SoPlex license you are only allowed to use SoPlex for non-commercial, academic work.

Download SoPlex here (version 1.3.2 has been used successfully):

`http://soplex.zib.de/download.shtml`

After download, follow the installation instructions

`http://www.zib.de/Optimization/Software/Soplex/html/INST.html`

After having installed SoPlex, you must tell Gfan where SoPlex is located. Do this by editing the lines

```
SOPLEX_PATH = $(HOME)/math/software/soplex-1.3.2
SOPLEX_LINKOPTIONS = -lz $(SOPLEX_PATH)/lib/libsoplex.darwin.x86.gnu.opt.a
```

of the file `Makefile` in your Gfan directory. Most likely you need to change `darwin` to `linux` in the last line. Finally you need to recompile Gfan . First run `make clean` and then `make` with the options from Subsection 2.3 together with the option `soplex=true`. Then do a `make install` as described in Subsection 2.3.

3 Using the software

In this section we will explain by examples how to use the software for the most common computations. Gfan consist of a set of subprograms with names like `gfan_bases` and `gfan_buchberger` each with a different purpose. See Appendix A for an explanation of the data formats and Appendix B for a full list of the various functions and their help files.

3.1 Computing the Gröbner fan

The program `gfan_bases` computes the set of reduced Gröbner bases of an ideal. To use it type in the name in the UNIX shell [1]

```
gfan_bases
```

and type in a polynomial ring followed by a set of generators for the ideal

```
Q[a,b,c]
{aab-c,bbc-a,cca-b}
```

For compatibility reasons the polynomial ring can be left out in which case the ring is assumed to be the polynomial ring over the rationals with variable names $a, b, c, \ldots$. The program will output the polynomial ring and the list of reduced Gröbner bases of the input ideal. In this example there are 33 such bases.

Often it is convenient to store your generators in a text file instead of typing them in every time you use the program. You can redirect the standard input for the program to read from a file instead of the keyboard. For example, if your ring and generators are stored in the file `myinputfile.txt` you would type:

```
gfan_bases <myinputfile.txt
```

If you want to store the output in the file `myoutputfile.txt` you can redirect the standard output as well:

```
gfan_bases <myinputfile.txt >myoutputfile.txt
```

The list of reduced Gröbner bases can be transformed into a polyhedral representation of the Gröbner fan by using the program `gfan_topolyhedralfan` as explained in Section 3.2.

Here is another example of a polynomial ring and an ideal:

```
Z/3Z[x_1,x_2,x_3]
{x_1^2x_2-x_3,x_2^2x_3-x_1,x_3^2x_1-x_2}
```

[1] It is actually much more convenient to use the Emacs shell. In Emacs press Meta-x and type shell. When you are in the Emacs shell Ctrl-up will allow you to easily reinput old polynomial data to Gfan .

3.1.1 Exploiting symmetry

As explained in Subsection 1.3.2 the program can do its computations up to symmetry. In the example above we may cycle the three variables without changing the ideal. Hence the subgroup $G \subseteq S_n$ in Subsection 1.3.2 is the group generated by a three cycle. A way to write down the subgroup is by writing a list of permutations that generate the subgroup:

```
{(0,1,2),(1,2,0)}
```

The first permutation is the identity (which can be left out). The second permutation is three-cycle. Together they generate G. See Appendix A for more information on how to specify the permutations.

The option `--symmetry` tells `gfan` to do its computations up to symmetry. For example,

```
gfan_bases --symmetry <anotherinputfile.txt
```

will read the generators for the ideal and the generators for the group and perform the computation up to symmetry. The input file would have to look like this:

```
Q[a,b,c]
{aab-c,bbc-a,cca-b}
{(0,1,2),(1,2,0)}
```

The output will be a list of reduced Gröbner basis - one for each orbit.

3.2 Combining the programs

The various Gfan programs can be combined. For example, if we are interested in the combinatorics of the Gröbner fan rather than the Gröbner bases, we can run the command:

```
gfan_bases <myinputfile.txt | gfan_topolyhedralfan
```

The output is a polyhedral fan in the format explained in Appendix A.7.

Similarly, the command line

```
gfan_buchberger <myinputfile.txt | gfan_groebnercone
```

produces the polyhedral cone (Appendix A.8) of the computed reduced Gröbner basis, and

```
gfan_buchberger <myinputfile.txt | gfan_groebnercone --asfan
```

computes the cone as a polyhedral fan (Appendix A.7) with all faces of the cone listed.

As another example, if we are interested in the list of monomial initial ideals rather than the complete list of reduced Gröbner bases of an ideal we will pipe the output of `gfan_bases` through the program `gfan_leadingterms`:

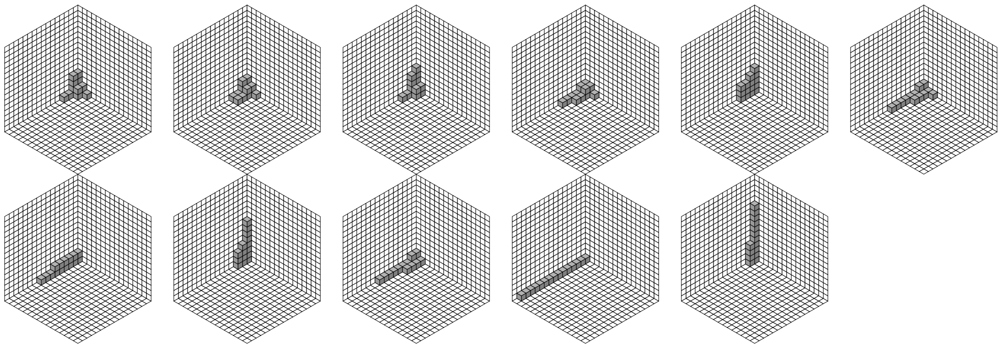

Figure 1: Staircase diagrams of the monomial initial ideals in the example - up to symmetry.

```
gfan_bases <myinputfile.txt | gfan_leadingterms -m
```

We need to use the option `-m` to tell `gfan_leadingterms` that it should expect a list of Gröbner bases rather than a single Gröbner basis on its input.

If we want the union of the Gröbner bases instead we should type:

```
gfan_bases <myinputfile.txt | gfan_polynomialsetunion >myoutputfile.txt
```

This will compute a *universal Gröbner basis.*

In three variables, if we want to draw staircase diagrams of the initial ideals we may use the program `gfan_renderstaircase`:

```
gfan_bases --symmetry <anotherinputfile.txt |
                              gfan_renderstaircase -m -w6 -d16 >out.fig
```

The output file is the xfig file in Figure 1. To save paper we used the `--symmetry` option and gave the program the file also containing the group generators as input.

In three variables, if we want to draw the Gröbner fan - or rather draw the intersection of the 2-dimensional standard simplex with the Gröbner fan we may use the program `gfan_render`:

```
gfan_bases <myinputfile.txt | gfan_render >myoutputfile.fig
```

The output is shown in Figure 2. If there are more than three variables in the polynomial ring this program can still be used but it is more difficult. See Appendix B.29.

3.3 Interactive mode

To study the local structure of the Gröbner fan the program `gfan_interactive` is useful. It allows the user to walk along an arbitrary path of full dimensional

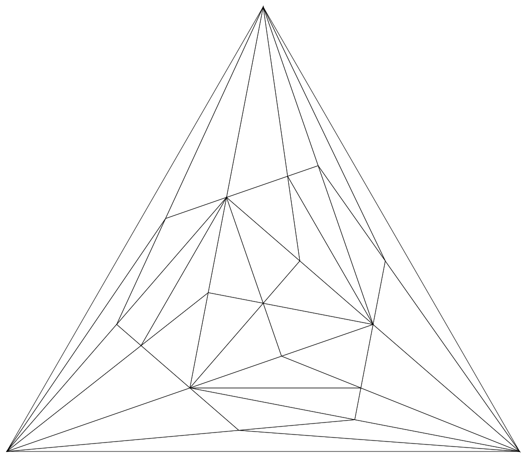

Figure 2: The Gröbner fan of the ideal intersected with the standard simplex.

Gröbner cones in the Gröbner fan of the ideal. At each step the user will specify which facet to walk through. The input must be a marked Gröbner basis. The program will minimise and autoreduce if necessary to get the reduced Gröbner basis. For example running the program

```
gfan_interactive
```

with input

```
Q[a,b,c]
{
c^15-c,
b-c^11,
a-c^9}
```

will give us a list of facets to walk through. (One way to get a starting Gröbner basis is by using the program `gfan_buchberger`.) In this case only two flips are possible, since the third wall does not lead to a new Gröbner cone. – The wall is on the boundary of the Gröbner region. We may choose any of the two remaining facets by typing in an index (a number) followed by $<$ enter $>$. See Appendix B.16 for more options.

3.4 Integers and p-adics

Gfan handles two settings in which the usual division and Buchberger algorithms do not suffice. These are ideals in $\mathbb{Z}[x_1, \ldots, x_n]$ and $\mathbb{Q}[x_1, \ldots, x_n]$, where, in the latter setting, the p-adic valuation is taken into account when defining initial ideals.

At the moment these two settings are handled by the commands `gfan_overintegers` and `gfan_padic`. They allow the computation of Gröbner bases, initial ideals, Gröbner cones (or polyhedra) and Gröbner fans (or complexes). In this section we give two examples. Use the `--help` option to get the full documentation.

Example 3.1 To compute the Gröbner fan of [21, Example 3.9], with the ideal considered in the ring $\mathbb{Z}[a, b, c]$ we run the command

```
gfan_overintegers --groebnerFan -g --log1
```

on

```
Q[a,b,c]
{a^5+b^3+c^2-1, b^2+a^2+c-1, c^3+a^6+b^5-1}
```

Since the type-system of Gfan does not understand Z[a,b,c] we need to trick Gfan by specifying the ring $Q[a, b, c]$ when running gfan_overintegers. From the output we conclude that the ideal has 1659 reduced Gröbner bases over the integers (as opposed to 360 over a field of characteristic 0).

Example 3.2 To compute the reduced Gröbner basis of $I = \langle x_1+2x_2-3x_3, 3x_2-4x_3+5x_4\rangle \subseteq \mathbb{Q}[x_1, \ldots, x_4]$ with respect to the vector $(1, 0, 0, 1)$ (tie-broken lexicographically) and with $\mathbb{Q}$ having the 2-adic valuation, we run

```
gfan_padic --groebnerBasis -p2
```

on the input

```
Q[x1,x2,x3,x4]
{
x1+2x2-3x3,
3x2-4x3+5x4
}
(1,1,0,0,1)
```

The first coordinate of the input vector is a 1, since _padic requires "homogenized" weight vectors. This is Example 2.4.3 in the upcoming book by Maclagan and Sturmfels on tropical geometry. The division algorithm implemented in Gfan used for this computation was proposed by Maclagan. To find all initial ideals, we can use a combination of `gfan_padic --groebnerBasis`, `gfan_combinerays --section CONES -i filename` and `gfan_padic --initialIdeals -m`.

Notice that the Gröbner complex of I, where valuation is taken into account (see Maclagan-Sturmfels), is not a fan. The output of `gfan_groebnerComlex`, however, will be a fan. To get the Gröbner complex we need to intersect the fan with the hyperplane $\omega_0 = 1$.

NOTICE THAT `_padic` USES THE MINIMUM CONVENTION AT THE MOMENT - in order to be consistent with Maclagan and Sturmfels.

3.5 Toric ideals and secondary fans

Gfan is a replacement of the software CaTS [13] which computes Gröbner fans of toric and lattice ideals. For convenience a program for computing lattice ideals has been added to Gfan. To compute the lattice ideal of the lattice generated by $(2, -1, 0)$ and $(3, 0, -1)$ we run:

```
gfan_latticeideal
{(2,-1,0),(3,0,-1)}
```

Gfan will transform the generators into binomials and compute the saturation of the ideal they generate by the product of all variables. The computation is independent of the characteristic of the field.

If on the other hand we wish to compute the toric ideal of a *vector configuration* given by the columns of the 1×3-matrix $(1, 2, 3)$ we run

```
gfan_latticeideal -t
{(1,2,3)}
```

More rows can be added to the matrix if we want.

The choice of the term “vector configuration” is intentional and nonstandard. The reason for this will become clear later in this section. In Gfan terminology a *point configuration* is reserved for the collection of points we have before we add a row of ones to construct a projective toric variety. By adding the row of ones the point configuration is turned into a vector configuration. Notice that scaling a vector of a vector configuration may change its toric ideal.

Computing toric ideals Gfan is not optimal. If one needs to do big examples the software 4ti2 [1] is recommended.

For a toric ideal the radical of a monomial initial ideal is the Stanley-Reisner ideal of a regular triangulation of the point configuration, see [21]. Hence the toric Groebner fan is a refinement of the *secondary fan*, indexing all regular triangulation of the point configuration.

The secondary fan of the vector configuration $\{(1, 0), (1, 1), (1, 2), (1, 3)\}$ can be computed by typing

```
gfan_secondaryfan
{(1,0),(1,1),(1,2),(1,3)}
```

Comparing this to the finer Gröbner fan of the corresponding toric ideal which you get by doing

```
gfan_transposematrix  | gfan_latticeideal -t | gfan_bases | gfan_topolyhedralfan
{(1,0),(1,1),(1,2),(1,3)}
```

you realise that three monomial initial ideals of the toric ideal have the same radical, while four monomial initial ideals pairwise have the same radical.

The secondary fan computation was added for convenience. An alternative is to use TOPCOM [19]. Notice however, that the vector configurations for gfan_secondaryfan do not have to be pointed. This means that all combinatorial types of polytopes with a fixed set of normals can be easily enumerated. This is not possible with TOPCOM.

4 Doing tropical computations

This section follows the max convention for tropical arithmetic. For the non-constant coefficient case tropical varieties are defined as in [17] and [15].

In this section we explain how to use Gfan to do tropical computations. For a fixed ideal $I \subseteq k[x_1, \ldots, x_n]$ the set of all faces of all full-dimensional Gröbner cones is a polyhedral complex which we call the Gröbner fan of I. For tropical computations the lower dimensional cones of the complex will be of our interest. In general every Gröbner cone is of the form:

$$C_\omega(I) := \overline{\{\omega' \in \mathbb{R}^n : \mathrm{in}_{\omega'}(I) = \mathrm{in}_\omega(I)\}}.$$

We define the tropical variety $\mathcal{T}(I)$ of an ideal I to be the the set of all ω such that $\mathrm{in}_\omega(I)$ does not contain a monomial. If the ideal I is homogeneous with respect to a positive grading, then the Gröbner cones cover all of $\mathbb{R}^n$ and $\mathcal{T}(I)$ is a union of Gröbner cones. Thus for a homogeneous ideal the tropical variety gets the structure of a polyhedral fan which it inherits from the Gröbner fan. We therefore also define the tropical variety $\mathcal{T}(I)$ to be the collection of all Gröbner cones $C_\omega(I)$ such that $\mathrm{in}_\omega(I)$ is monomial-free.

We start by noticing that for computational purposes it is no restriction to only consider the case of a homogeneous ideal:

Lemma 4.1 *[15, Lemma 6.2.5] Let $I \subseteq k[x_1, \ldots, x_n]$ be an ideal generated by $f_1, \ldots, f_m \in k[x_1, \ldots, x_n]$. Let $J = \langle f_1^h, \ldots, f_m^h \rangle \subseteq k[x_0, \ldots, x_n]$. Then $\mathrm{in}_\omega(I)$ is monomial-free if and only if $\mathrm{in}_{(0,\omega)}(J)$ is monomial-free where $\omega \in \mathbb{R}^n$. In particular we have the following identity of sets in $\mathbb{R}^{n+1}$:*

$$\{0\} \times \mathcal{T}(I) = \mathcal{T}(J) \cap (\{0\} \times \mathbb{R}^n).$$

Here f^h denotes the homogenization of the polynomial f. The homogenization of a list of polynomials can be computed by the program `gfan_homogenize`. Notice that the lemma only requires the generators to be homogenized as a set of polynomials and not in the sense of a polynomial ideal.

The tropical algorithms implemented in Gfan are explained in [5]. Notice that Gfan follows the usual conventions for signs of weight vectors defining initial forms while [5] uses opposite signs. This means that Gfan is compatible with the max-plus convention whereas [5] is compatible with the min-plus convention.

4.1 Tropical variety by brute force

The command `gfan_tropicalbruteforce` will compute all Gröbner cones of a homogeneous ideal and for each check if its initial ideal contains a monomial. The output is the tropical variety of the ideal. Since the tropical variety is usually much smaller than the Gröbner fan this is a rather slow method for computing the tropical variety. The line

```
gfan_buchberger | gfan_tropicalbruteforce
```

run on the input

```
Q[a,b,c,d,e,f,g,h,i,j]
{
bf-ah-ce,
bg-ai-de,
cg-aj-df,
ci-bj-dh,
fi-ej-gh
}
```

produces a tropical variety of the input ideal in a few minutes as a polyhedral fan, see Section A.7. We use `gfan_buchberger` since `gfan_tropicalbruteforce` requires its input to be a marked reduced Gröbner basis.

Remark 4.2 Notice that if $k' \supseteq k$ is a field extension and $I \subseteq k[x_1, \ldots, x_n]$ an ideal then $\mathcal{T}(I) = \mathcal{T}(\langle I \rangle_{k'[x_1,\ldots,x_n]})$ as a polyhedral fan. This identity follows since both objects can be computed by Gröbner basis methods and Gröbner bases are independent of such field extensions. The same argument of course also applies to the Gröbner fans of the two ideals.

4.2 Traversing tropical varieties of prime ideals

Let $I \subseteq \mathbb{C}[x_1, \ldots, x_n]$ be a homogeneous monomial-free prime ideal of dimension d. By the Bieri Groves Theorem [4] the tropical variety of I is a pure d-dimensional polyhedral fan. It is connected in codimension one ([5, Theorem 14]) and can be traversed by Gfan. Let ω be a relative interior point of a d-dimensional Gröbner cone in the tropical variety of I. Fix some term order $\prec$. Gfan represents $C_\omega(I)$ by the pair of marked reduced Gröbner bases $(\mathcal{G}_{\prec_\omega}(\mathrm{in}_\omega(I)), \mathcal{G}_{\prec_\omega}(I))$. To compute the tropical variety of an ideal we must begin by finding a starting d-dimensional Gröbner cone. For this `gfan_tropicalstartingcone` is used. This programs guesses a starting cone by heuristic methods. The guessing might fail. In that case the program will terminate with an error message. After having computed a starting cone we use the program `gfan_tropicaltraverse` to traverse the tropical variety. We illustrate the procedure with an example.

Remark 4.3 Gfan does its computations over $\mathbb{Q}$ and thus the input should be an ideal generated by polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$. The assumption that I is an ideal in $\mathbb{C}[x_1, \ldots, x_n]$ is needed since by "prime ideal" in the above we mean "prime ideal in the polynomial ring over the algebraically closed field $\mathbb{C}$". If I is a prime ideal in $\mathbb{Q}[x_1, \ldots, x_n]$ we do not know that its tropical variety is connected. In Section 4.6 we address the problem of specifying non-rational coefficients.

Example 4.4 Let $I \subseteq \mathbb{Q}[a, \ldots, o]$ be the ideal generated by the relations on the 2 by 2 minors of a 2 by 6 generic matrix. In $\mathbb{C}[x_1, \ldots, x_n]$ the ideal I generates a prime ideal. To get a starting cone for the traversal of $\mathcal{T}(I)$ we run the command

```
gfan_tropicalstartingcone
```

on the input

```
Q[a,b,c,d,e,f,g,h,i,j,k,l,m,n,o]
{
bg-aj-cf,  bh-ak-df,  bi-al-ef,  ck-bm-dj,  ch-am-dg,
cl-ej-bn,  ci-eg-an,  dn-co-em,  dl-bo-ek,  di-ao-eh,
gk-fm-jh,  gl-fn-ij,  hl-fo-ik,  kn-jo-lm,  hn-im-go
}
```

and get a pair of marked reduced Gröbner bases

```
Q[a,b,c,d,e,f,g,h,i,j,k,l,m,n,o]
{
l*m+j*o,  i*m+g*o,  i*k-h*l,  i*j-g*l,  h*j-g*k,
e*m+c*o,  e*k+b*o,  e*j-c*l,  e*h+a*o,  e*g-c*i,
c*k-b*m,  c*h-a*m,  b*i-a*l,  b*h-a*k,  b*g-a*j}
{
l*m-k*n+j*o, i*m-h*n+g*o, i*k-h*l+f*o, i*j-g*l+f*n, h*j-g*k+f*m,
e*m-d*n+c*o, e*k-d*l+b*o, e*j-c*l+b*n, e*h-d*i+a*o, e*g-c*i+a*n,
c*k-d*j-b*m, c*h-d*g-a*m, b*i-e*f-a*l, b*h-d*f-a*k, b*g-c*f-a*j}
```

This takes a minute or so. We store the output in the file `grassmann2_6.cone` for later use. Since I has many symmetries we add the following lines describing the symmetry group to the end of the file:

```
{
(0,8,7,6,5,4,3,2,1,14,13,11,12,10,9),
(5,6,7,8,0,9,10,11,1,12,13,2,14,3,4)
}
```

We are ready to traverse $\mathcal{T}(I)$. We run the following command

```
gfan_tropicaltraverse --symmetry <grassmann2_6.cone
```

The computation takes a few (two - three) minutes. The output looks like this:

```
_application PolyhedralFan
_version 2.2
_type PolyhedralFan

AMBIENT_DIM
15

DIM
9

LINEALITY_DIM
6

RAYS
0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 # 0
0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 # 1
0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 # 2
0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 # 3
0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 # 4
0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 # 5
0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 # 6
0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 # 7
0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 # 8
0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 # 9
0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 # 10
0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 # 11
0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 # 12
-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 # 13
0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 # 14
0 0 0 -1 -1 -1 -1 0 0 -1 0 0 0 0 -1 # 15
-1 -1 0 0 0 -1 0 0 0 0 0 0 -1 -1 -1 # 16
-1 0 0 0 -1 0 0 0 -1 -1 -1 0 -1 0 0 # 17
1 1 0 0 0 1 2 2 0 2 2 0 1 1 1 # 18
1 0 2 2 1 0 0 0 1 1 1 0 1 2 2 # 19
0 0 0 1 1 1 1 0 0 1 2 2 2 2 1 # 20
1 1 0 2 2 1 0 2 2 0 0 0 1 1 1 # 21
1 2 2 0 1 2 2 0 1 1 1 0 1 0 0 # 22
2 2 0 1 1 1 1 0 2 1 0 2 0 0 1 # 23
0 -1 0 -1 0 0 -1 0 -1 0 -1 0 0 -1 0 # 24

N_RAYS
25
```

```
LINEALITY_SPACE
0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
0 0 0 0 1 0 0 0 1 0 0 1 0 1 1
0 0 0 1 0 0 0 1 0 0 1 0 1 0 1
0 0 1 0 0 0 1 0 0 1 0 0 1 1 0
0 1 0 0 0 0 -1 -1 -1 0 0 0 -1 -1 -1
1 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 -1

ORTH_LINEALITY_SPACE
0 0 0 0 0 0 0 0 0 0 1 -1 -1 1 0
0 0 0 0 0 0 0 0 0 1 0 -1 -1 0 1
0 0 0 0 0 0 0 1 -1 0 0 0 -1 1 0
0 0 0 0 0 0 1 0 -1 0 0 0 -1 0 1
0 0 0 0 0 1 0 0 -1 0 0 -1 -1 1 1
0 0 0 1 -1 0 0 0 0 0 0 0 -1 1 0
0 0 1 0 -1 0 0 0 0 0 0 0 -1 0 1
0 1 0 0 -1 0 0 0 0 0 0 -1 -1 1 1
1 0 0 0 -1 0 0 0 -1 0 0 0 -1 1 1

F_VECTOR
1 25 105 105
```

After this follows a list of cones and maximal cones. Every maximal cone has an associated multiplicity which is also listed.

The output says that the tropical variety has dimension 9. Modulo the 6-dimensional homogeneity space this is reduced to a 3-dimensional complex in $\mathbb{R}^9$ and thus we may think of the tropical variety as a 2-dimensional polyhedral complex on the 8-sphere in $\mathbb{R}^9$. This complex is simplicial and has 105 maximal cones.

The extreme rays (modulo the homogeneity space) are labeled $0, \ldots, 24$. In the cone lists the cones are grouped together according to dimension and orbit with respect to the specified symmetries. See Section A.7 for more information on how to read the polyhedral fan format.

While traversing the variety the program `gfan_tropicaltraverse` only computes d and $(d-1)$-dimensional cones. The other cones are extracted after traversing. Also the symmetries are expanded. Sometimes extracting all cones is time consuming and one is only interested in the high dimensional cones up to symmetry. These can be output using the option `--noincidence`. In that case the output would be a list of orbits for maximal cones and a list of orbits for codimension one cones. It is also listed how these cones are connected taking symmetry into account. In general that format is rather difficult to read.

A final remark about `gfan_tropicaltraverse` is that the polyhedral structure of the complex comes from the Gröbner fan. For some ideals it is possible to find polyhedral fans covering the tropical variety with fewer cones.

4.3 Intersecting tropical hypersurfaces

The tropical variety of a principal ideal is called a *tropical hypersurface*. A *tropical prevariety* is a finite intersection of tropical hypersurfaces or, to be precise, the intersection of the support set of these hypersurfaces. In Gfan the intersection is represented by the *common refinement* of the tropical hypersurfaces. The program `gfan_tropicalintersection` can compute such intersections.

Example 4.5 To compute the intersection of the tropical hypersurfaces $\mathcal{T}(\langle a + b + c + 1\rangle)$ and $\mathcal{T}(\langle a + b + 2c\rangle)$ we run

```
gfan_tropicalintersection
```

on

```
Q[a,b,c]
{a+b+c+1,a+b+2c}
```

The output is a polyhedral fan whose support is the intersection. The balancing condition for this fan is not satisfied which implies that it is not a tropical variety.

4.4 Computing tropical bases of curves

In Gfan an ideal I is said to define a *tropical curve* if $k[x_1, \ldots, x_n]/I$ has Krull dimension equal to or one larger than the dimension of the homogeneity space of I. A *tropical basis* of I is a finite generating set for the ideal such that the tropical variety defined by I (as a set) is the intersection of the tropical hypersurfaces of the generators. A tropical basis always exists [5]. The program gfan_tropicalbasis computes a tropical basis for an ideal defining a tropical curve.

Example 4.6 Again we consider the ideal $\langle a + b + c + 1, a + b + 2c\rangle$. We notice that this ideal defines a curve since the Krull dimension is 1 and the dimension of the homogeneity space is 0. In the example above we saw that the listed set is not a tropical basis. We run

```
gfan_tropicalbasis -h
```

on

```
Q[a,b,c]
{a+b+c+1,a+b+2c}
```

to get some tropical basis

```
Q[a,b,c]
{
-1+c,
2+b+a}
```

We needed the option `-h` here since the ideal was not homogeneous. If we run `gfan_tropicalintersection` on the output we see that the tropical variety consists of three rays and the origin.

4.5 Tropical intersection theory

Gfan contains a few experimental programs for doing computations in tropical intersection theory. In [2, Definition 3.4] the tropical Weil divisor of a tropical rational function on a (tropical) k-cycle in $\mathbb{R}^n$ is defined. This divisor can be computed in Gfan. However, Gfan and [2] do not agree on the basic definitions in tropical geometry. For example the definition of a fan is different. Here we will adjust the necessary definitions to the Gfan conventions. A tropical k-cycle will be a pure (rational) polyhedral fan F of dimension k in $\mathbb{R}^n$ with weights which is balanced in the following sense: To every k-dimensional facet C we assign a weight (or multiplicity) $m_C \in \mathbb{Z}$. The vector space $\mathbb{R}^n$ comes with its standard lattice $\mathbb{Z}^n$. For a $k-1$-dimensional ridge $R \in F$ and a facet C in its star[2] in F corresponding to a cone L in the link[3] of R in F, the semigroup $L \cap \mathbb{Z}^n/\mathrm{span}_{\mathbb{R}}(R) \cap \mathbb{Z}^n \subseteq \mathbb{Z}^n/\mathrm{span}_{\mathbb{R}}(R) \cap \mathbb{Z}^n$ is isomorphic to $\mathbb{N}$. Define $u_{C/R} \in \mathbb{Z}^n/\mathrm{span}(R) \cap \mathbb{Z}^n$ as the element identified with $1 \in \mathbb{N}$. The balancing condition at R is that

$$\sum_{C \in F : R \subset C} m_C u_{C/R} = 0.$$

For a (weighted) fan to be a tropical cycle this must hold at every ridge R.

It remains to define what a tropical rational function is. Take a polyhedral fan F' and associate to each of its maximal cones a linear form. When evaluating a point x in the support of F' simply evaluate the linear form of cone containing x. If this gives a well-defined function we call this function a tropical rational function. When computing Weil divisors we will require that the supports satisfy $\mathrm{supp}(F) \subseteq \mathrm{supp}(F')$. There will be no further restriction on the polyhedral structure.

For a definition of the Weil divisor itself we refer to [2, Definition 3.4]. Here we just mention that it again is a cycle of dimension one lower.

To demonstrate the Gfan features we recompute [2, Example 3.10]. An easy way to generate the k-cycle of that example is to compute it as a hypersurface.

[2] the smallest polyhedral subcomplex of F containing all faces of F containing R.

[3] take an ϵ-ball around a relative interior $\omega \in R$ and intersect it with F. Translating the ball to the origin and scaling the intersection to infinity we get the link of R in F.

Since the paper is using min and Gfan is using max we need to change the polynomial from the paper such that the Newton polytope is flipped:

```
gfan_tropicalhypersurface > tmpfile1.poly
Q[x_1,x_2,x_3]
{x_2x_3+x_1x_3+x_1x_2+x_1x_2x_3}
```

The weights/multiplicities are stored in the MULTIPLICITIES section of the Polymake file.

It is harder specifying the rational function. We make the following file and call in `func.poly`.

```
_application PolyhedralFan
_version 2.2
_type PolyhedralFan

AMBIENT_DIM
3

DIM
2

LINEALITY_DIM
0

RAYS
1 0 0 # 0
0 1 0 # 1
0 0 1 # 2
-1 -1 -1 # 3
1 1 0 # 4
-1 -1 0 # 5

N_RAYS
6

LINEALITY_SPACE

ORTH_LINEALITY_SPACE
1 0 0
0 1 0
0 0 1

MAXIMAL_CONES
{3 5} # Dimension 2
{5 2}
{0 2}
{1 2}
{1 3}
{0 3}
{1 4}
```

```
{0 4}

MULTIPLICITIES
1
1
1
1
1
1
1
1

RAY_VALUES
0
0
0
1
-1
0

LINEALITY_VALUES
```

Instead of specifying the linear function on each maximal cone we have to specify its values on each of the rays in the fan and each of the generators of the lineality space. Then Gfan will automatically interpolate the function. Since the lineality space of the fan is empty we leave the LINEALITY_VALUES section empty.

We now compute the Weil divisor:

```
gfan_tropicalweildivisor -i1 tmpfile1.poly -i2 func.poly >tmpfile2.poly
```

...and compute the Weil divisor again as in [2]...

```
gfan_tropicalweildivisor -i1 tmpfile2.poly -i2 func.poly >tmpfile3.poly
```

We get a fan with the origin being the only cone. It has multiplicity -1:

```
MULTIPLICITIES
-1      # Dimension 0
```

There is another useful command for computing polyhedral fans for rational functions. The command `gfan_tropicalfunction` takes a polynomial and turns it into a fan representing its tropicalization which is a tropical rational function.

4.6 Non-constant coefficients

In tropical geometry it is common to take the valuation of $\mathbb{C}\{\{t\}\}$ into account when defining the tropical variety of an ideal in $\mathbb{C}\{\{t\}\}[x_1, \ldots, x_n]$. Here $\mathbb{C}\{\{t\}\}$ denotes the field of Puiseux series. The valuation $\mathrm{val}(p)$ of a non-zero Puiseux series p is the degree of its lowest order term.

Definition 4.7 For $\omega \in \mathbb{R}^n$ the *t-ω-degree* of a term $ct^a x^v$ with $c \in \mathbb{C}^*$, $a \in \mathbb{Q}$ and $v \in \mathbb{Z}^n$ is defined as $-\text{val}(ct^a) + \omega \cdot v = -a + \omega \cdot v$. The *t-initial form* $\text{t-in}_\omega(f) \in \mathbb{C}[x_1, \dots, x_n]$ of a polynomial $f \in \mathbb{C}\{\{t\}\}[x_1, \dots, x_n]$ is the sum of all terms in f of maximal t-ω-weight but with 1 substituted for t.

Remark 4.8 Notice that since t has t-ω-degree -1, the maximal t-ω-weight *is* attained by a term if the polynomial is non-zero. Furthermore, only a finite number of terms attain the maximum. Therefore, it makes sense to substitute $t = 1$ and consider the finite sum of terms as a polynomial in $\mathbb{C}[x_1, \dots, x_n]$.

Example 4.9 Consider $f = (1 + t) + t^2 x + tx^2 \in \mathbb{C}\{\{t\}\}[x_1, \dots, x_n]$. Let $\omega = (\frac{1}{2}) \in \mathbb{R}^1$. Then $\text{t-in}_\omega(f) = 1 + x^2$. For any other choice of ω the t-initial form is a monomial.

Definition 4.10 Let $I \subseteq \mathbb{C}\{\{t\}\}[x_1, \dots, x_n]$ and $\omega \in \mathbb{R}^n$. The *t-initial ideal* of I with respect to ω is defined as:

$$\text{t-in}_\omega(I) := \langle \text{t-in}_\omega(f) : f \in I \rangle \subseteq \mathbb{C}[x_1, \dots, x_n].$$

Definition 4.11 Let $I \subseteq \mathbb{C}\{\{t\}\}[x_1, \dots, x_n]$ be an ideal. The *tropical variety* of I is the set

$$\mathcal{T}'(I) := \{\omega \in \mathbb{R}^n : \text{t-in}_\omega(I) \text{ is monomial-free}\}.$$

We use the notation $\mathcal{T}'(I)$ to avoid contradicting our original definition of the tropical variety of an ideal in the polynomial ring over a field.

Proposition 4.12 *[17, Proposition 7.3] Let $I \subseteq \mathbb{C}[t, x_1, \dots, x_n]$ be an ideal, $J = \langle I \rangle_{\mathbb{C}\{\{t\}\}[x_1,\dots,x_n]}$ and $\omega \in \mathbb{R}^n$. Then* $\text{t-in}_\omega(I) = \text{t-in}_\omega(J)$.

Remark 4.13 For $f \in \mathbb{C}[t, x_1, \dots, x_n]$ we have $\text{t-in}_\omega(f) = (\text{in}_{(-1,\omega)}(f))|_{t=1}$. Consequently, for $I \subseteq \mathbb{C}[t, x_1, \dots, x_n]$ we have $\text{t-in}_\omega(I) = (\text{in}_{(-1,\omega)}(I))|_{t=1}$. In order to decide if $\text{t-in}_\omega(I)$ contains a monomial we may simply decide if the initial ideal $\text{in}_{(-1,\omega)}(I)$ contains a monomial. As a corollary we get

$$\mathcal{T}(I) \cap (\{-1\} \times \mathbb{R}^n) = \{-1\} \times \mathcal{T}'(J).$$

In fact this gives a method for computing the tropical variety as a set of any ideal $J \subseteq \mathbb{C}\{\{t\}\}[x_1, \dots, x_n]$ generated by elements in the polynomial ring over the field of rational functions $\mathbb{Q}(t)[x_1, \dots, x_n]$ in Gfan by clearing denominators and intersecting the result with the $t = -1$ plane. (We remind the reader that Lemma 4.1 shows that for computational purposes it is no restriction if I is not homogeneous.)

Intersecting the tropical variety with the $t = -1$ plane can with some difficulty be done by hand. If the tropical (pre)-variety has been computed with `gfan_tropicalintersection` then it is also possible to let Gfan do the intersection. What Gfan does is to compute the common refinement of the fan with the fan consisting of the halfspace $t \leq 0$ and its proper face. Of course this does not remove the cones in the $t = 0$ plane, but they are easily removed by hand. We illustrate the procedure by an example.

Example 4.14 Exercise 2 in Chapter 9 of [22] asks us to draw the variety defined by the *tropical* polynomial $f = 1x^2 + 2xy + 1y^2 + 3x + 3y + 1$. If we tropically divide this polynomial by 3 we get $f' := f/3 = -2x^2 - 1xy - 2y^2 + 0x + 0y + -2$ which defines the same tropical variety. This variety equals the variety defined by the polynomial $g = t^2x^2 + txy + t^2y^2 + x + y + t^2 \in \mathbb{C}\{\{t\}\}[x, y]$. Notice that f' is the tropicalisation of g.

According to Remark 4.13 above the we may compute $\mathcal{T}'(\langle g\rangle)$ by computing the variety of $\langle t^2x^2 + txy + t^2y^2 + x + y + t^2\rangle \subseteq \mathbb{C}[t, x, y]$ and intersecting it with the hyperplane $t = -1$. Running

```
gfan_tropicalintersection --tplane
```

on

```
Q[t,x,y]
{t^2x^2+txy+t^2y^2+x+y+t^2}
```

we get

```
RAYS
0 -1 0 # 0
-1 2 1 # 1
0 1 1 # 2
-1 1 1 # 3
-1 -2 -2 # 4
0 0 -1 # 5
-1 1 2 # 6

MAXIMAL_CONES
{3 4} # Dimension 2
{2 6}
{1 3}
{1 2}
{3 6}
{4 5}
{0 4}
{0 6}
{1 5}
```

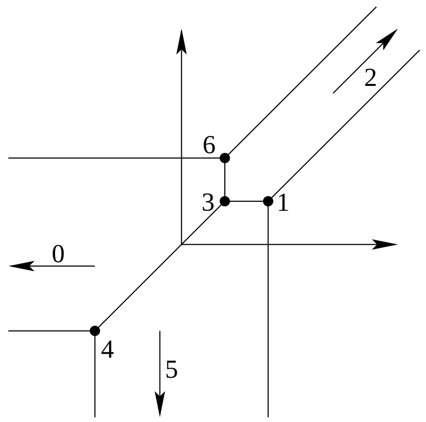

Figure 3: The tropical variety defined by the tropical polynomial in Example 4.14.

among other information. We can now draw the two-dimensional picture asked for in the exercise. The rays with non-zero first coordinate become points in the picture. (If the first coordinate is not -1 scaling is required to get the rational x, y-coordinates.) The rays with zero first coordinate become directions. The maximal cones show how to connect the rays; see Figure 3. Notice that some of the connections could have been "at infinity".

4.6.1 Algebraic field extensions of $\mathbb{Q}$

Ignoring time, memory usage and overflows Gfan can compute the tropical variety $\mathcal{T}'(I)$ of any ideal $I \subseteq \mathbb{C}\{\{t\}\}[x_1, \ldots, x_n]$ generated by elements of $\overline{\mathbb{Q}}(t)[x_1, \ldots, x_n]$. This is a consequence of the following lemma:

Lemma 4.15 *[17, Lemma 3.12] Let k be a field and $M = \langle m \rangle \subseteq k[a]$ a maximal ideal where m is not a monomial. Let $I \subseteq (k[a]/M)[x_1, \ldots, x_n]$ be an ideal. For $\omega \in \mathbb{R}^n$ we have*

$$\mathrm{in}_\omega(I) \text{ contains a monomial} \iff \mathrm{in}_{(0,\omega)}(\varphi^{-1}(I)) \text{ contains a monomial}$$

where $\varphi : k[a, x_1, \ldots, x_n] \to (k[a]/M)[x_1, \ldots, x_n]$ is the homomorphism taking elements to their cosets.

A Data formats

In this section we describe how polynomials, lists, marked Gröbner bases etc. are represented as ASCII character strings. These strings will be input to the programs by typing or by redirecting the standard input and be output by the program on the standard output which may be the screen, a pipe or a file. Usually files are used for input. For example,

```
gfan_bases < inputfile.txt > outputfile.txt
```

will read its input from `inputfile.txt` and write its output to `outputfile.txt`. The following is an example of how to use pipes for computing a universal Gröbner basis of the input:

```
gfan_bases < inputfile.txt | gfan_polynomialsetunion > outputfile.txt
```

In general spaces and newlines in the input are ignored, but for the polyhedral formats described in Section A.7 and Section A.8 the rules are different.

A.1 Fields

Two kinds of fields are supported:

- The field $\mathbb{Q}$ of rationals which is represented by the string "`Q`".
- Fields of the form $\mathbb{Z}/p\mathbb{Z}$ where p is a prime number. These fields are represented by text strings "`Z/pZ`" where `p` is the prime number. For example "`Z/3Z`" or "`Z/17Z`". In Gfan the prime number p must be less than 32749.

A.2 Variables

A variable is denoted by its name which is an string of characters. The exact rules for which names are allowed have not been decided on in this version of Gfan and therefore Gfan accepts most names. However, white spaces, commas and "`]`" are not allowed as characters in the name. Furthermore one should not choose variable names such that one name is a starting substring of an other – don't choose names such as "`x1`" and "`x`" in the same polynomial ring.

A.3 Polynomial rings

A polynomial ring is represented first by a field and then by a list of variable names. The list begins with "`[`" and ends with "`]`". Names are separated by commas. The ordering of the variables matters as this is also the ordering used for the entries of for example weight vectors. Examples: "`Z/2Z[a,b]`" and "`Q[x_1,x_2,y1,y2]`".

A.4 Polynomials

Coefficients in the field are given as fractions. A coefficient equals its numerator multiplied by the inverse of the denominator. The numerator and denominator themselves are given by an integer in $\mathbb{Z}$ which is mapped to the field by the homomorphism sending $1 \in \mathbb{Z}$ to 1 in the field. The '`/`' character and the denominator can be left out if the denominator is 1. If a field with non-zero characteristic was chosen one should be careful that the denominator is not 0.

Monomials are written in the following formats:

- `a^4dc`
- `a4dc`
- `aaadac`

The monomial 1 cannot be written without writing it as a term in the usual way "`1`". Any other term is either a monomial or a coefficient and a monomial. A polynomial is a list of terms separated by `+`. The `+` may be left out if the numerator of the next monomial is negative.

That description did not cover every detail. Here is an example:

```
hello world - 3/8 a2+23abcge^4 +1
```

In our usual notation we would write it like this: $dehl^3o^2rw + 1 + 23abce^4g - \frac{3}{8}a^2$. **It is important to note that the first term written in a polynomial is distinguished from the other terms in the polynomial. This is useful when specifying marked Gröbner bases.**

A.5 Lists

A list begins with a '`{`' or a '`(`', contains elements separated by '`,`' and is ended by a '`}`' or a '`)`'. Different types of lists may be needed when specifying input for the various programs:

An integer vector is a list of integers.

A list of integer vectors is a list of integer vectors. Such lists are used for example when specifying generators for subgroups of S_n.

A polynomial list (or a polynomial set) is a list of polynomials.

A Gröbner basis is the list of polynomials in a Gröbner basis with the leading term of each listed polynomial being the initial term with respect to a term order for which this a Gröbner basis.

A list of polynomial sets is a list of polynomial sets. Often the polynomial sets are required to be Gröbner bases.

An ideal is written as a list of polynomials generating it.

For all other lists than integer vectors the characters '{' and '}' are used to start and end the list.

A.6 Permutations

When exploiting the symmetry of an ideal one needs to input permutations to the program. Each permutation is specified by a vector. The length of the vector should equal the number of elements being permuted - for example the number of variables in the polynomial ring. The first element in the vector describes where the first element goes and so on. Of course we start indexing from 0. The following vectors specify the identity, a transposition and a 3-cycle, respectively, on an ordered set of four elements:

- `(0,1,2,3)`
- `(1,0,2,3)`
- `(0,3,1,2)`

A.7 Polyhedral fans

The output format for polyhedral fans is intended to be Polymake [10] compatible. Polymake recently switched to an XML based format. Gfan will keep outputting data in the old text style by default as it is more convienient for the command line Gfan user. The option `--xml` switches output to being XML. In the Polymake world, the output objects will have type `SymmetricFan`. The new XML files cannot be read by Gfan at the moment. In the following we describe only the text (non-XML) format. For the XML format we refer to the Polymake documentation.

The text representation of a fan begins with the lines

```
_application fan
_version 2.2
_type SymmetricFan
```

After this follows a list of properties. For example

```
LINEALITY_SPACE
0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
0 0 0 0 1 0 0 0 1 0 0 1 0 1 1
0 0 0 1 0 0 0 1 0 0 1 0 1 0 1
0 0 1 0 0 0 1 0 0 1 0 0 1 1 0
0 1 0 0 0 0 -1 -1 -1 0 0 0 -1 -1 -1
1 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 -1
```

Each property has a name and must be assigned a value of a certain type. You can read more about the philosophy of the format in the Polymake documentation.

Example A.1 The ideal $I = \langle ab - c, bc - a, ca - b\rangle$ has a cyclic symmetry. If we run the commands

```
gfan --symmetry -e | gfan_topolyhedralfan --symmetry
```

on the input

```
Q[a,b,c]
{ab-c,bc-a,ca-b}
{(1,2,0)}
```

we get a polyhedral representation of the Gröbner fan of I:

```
_application PolyhedralFan
_version 2.2
_type PolyhedralFan

AMBIENT_DIM
3

DIM
3

LINEALITY_DIM
0

RAYS
1 1 0 # 0
1 0 1 # 1
0 1 1 # 2
0 1 0 # 3
1 0 0 # 4
0 0 1 # 5
2 1 1 # 6
1 1 2 # 7
1 2 1 # 8
1 1 1 # 9

N_RAYS
10

LINEALITY_SPACE
```

```
ORTH_LINEALITY_SPACE
0 0 1
0 1 0
1 0 0

F_VECTOR
1 10 18 9

CONES
{} # New orbit # Dimension 0
{0} # New orbit # Dimension 1
{1}
{2}
{3} # New orbit
{4}
{5}
{6} # New orbit
{7}
{8}
{9} # New orbit
{0 3} # New orbit # Dimension 2
{1 4}
{2 5}
{0 4} # New orbit
{1 5}
{2 3}
{6 9} # New orbit
{7 9}
{8 9}
{0 6} # New orbit
{1 7}
{2 8}
{0 8} # New orbit
{1 6}
{2 7}
{3 8} # New orbit
{4 6}
{5 7}
{0 3 8} # New orbit # Dimension 3
{1 4 6}
{2 5 7}
{0 4 6} # New orbit
```

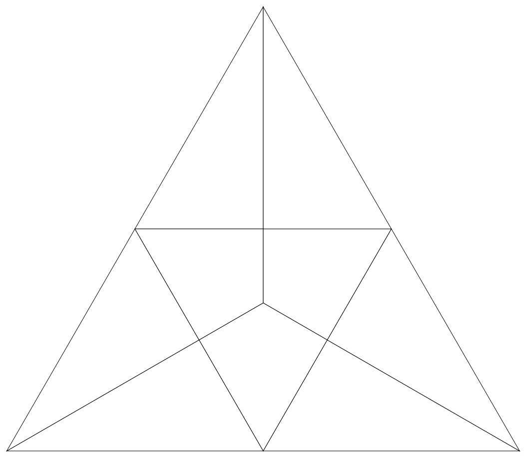

Figure 4: The Gröbner fan in Example A.1 intersected with the standard simplex in $\mathbb{R}^3$.

```
{1 5 7}
{2 3 8}
{0 6 8 9} # New orbit
{1 6 7 9}
{2 7 8 9}

MAXIMAL_CONES
{0 3 8} # New orbit # Dimension 3
{1 4 6}
{2 5 7}
{0 4 6} # New orbit
{1 5 7}
{2 3 8}
{0 6 8 9} # New orbit
{1 6 7 9}
{2 7 8 9}

PURE
1
```

The most important properties are "RAYS" and "CONES". A ray is given by a relative interior point and a cone is given by a list of indices of rays that will generate the cone. We may compare this combinatorial data to the drawing of the Gröbner fan given in Figure 4. Notice that this example is particularly simple as the dimension of the homogeneity space of I is 0.

The symbol "#" is used for writing comments in the file. The comments should not be considered a part of the file. The comments are used by Gfan to let the user know about dimensions, orbits and indices.

A detailed description of the properties follows in the following.

A.7.1 Data types

In Gfan's Polymake format the following data types are supported:

Cardinal: One non-negative integer.

Boolean: 0 or 1.

Matrix: An array of integer vectors.

IncidenceMatrix: An array of sets of integers.

Vector: An integer vector.

A.7.2 Properties

Before we describe the properties we need to make a few definitions.

We do not consider the empty set to be a cone nor a face of a cone.

Definition A.2 The *lineality space* of a polyhedral cone is the largest subspace contained in the cone.

The lineality space is the smallest face of the cone and if two cones are in the same polyhedral fan then they must have the same lineality space. We define the *lineality space* of a fan to be the common lineality space of its cones. In the special case of a (non-restricted) Gröbner fan or a tropical variety the lineality space of the fan coincides with the homogeneity space of the defining ideal.

Definition A.3 A cone (in a fan) is called a *ray* if its dimension is one larger than the dimension of its lineality space.

A ray can be represented by a vector in its relative interior. This vector is contained in the cone but not contained in any of its proper faces. The representation is not unique since the cone is invariant under translation by vectors in its lineality space.

A Polyhedral fan in Gfan can have a subset of the following properties:

AMBIENT_DIM is a *Cardinal* whose value is the dimension of the vector space in which the fan lives. If the fan is a Gröbner fan or a tropical variety then this number equals the number of variables in the polynomial ring of the defining ideal.

DIM is a *Cardinal* whose value is the dimension of the highest dimensional cone in the fan.

LINEALITY_DIM is a *Cardinal* whose value is the dimension of the lineality space of the fan.

RAYS is a *Matrix*. The rows of the matrix are vectors representing the rays of the fan — one for each ray. The rows are ordered and Gfan writes an index as a comment to make the file human readable.

N_RAYS is a *Cardinal* which equals the number of rays in the fan.

LINEALITY_SPACE is a *Matrix* whose rows form a basis for the lineality space of the fan.

ORTH_LINEALITY_SPACE is a *Matrix* whose rows form a basis for the orthogonal complement of the lineality space of the fan.

F_VECTOR is a *Vector*. The number of entries is DIM−LINEALITY_DIM+1. The ith entry is the number of cones in the fan of dimension i+LINEALITY_DIM−1.

CONES is an *IncidenceMatrix*. The section contains a line for each cone in the fan. Each line is the set of indices of the rays contained in the corresponding cone.

MAXIMAL_CONES is an *IncidenceMatrix* and similar to CONES except that only cones which are maximal with respect to inclusion are listed.

PURE is a *Boolean*. The value is 1 if the polyhedral fan is pure and 0 otherwise.

MULTIPLICITIES is a *Matrix* with one column. An entry is the multiplicity of a maximal cone. Usually cones in polyhedral fans do not have multiplicities. Thus this property only makes sense for *weighted* polyhedral fans of which tropical varieties is a special case. The ordering of the rows in this property is consistent with the ordering in MAXIMAL_CONES.

RAY_VALUES is a *Matrix* with just one column. It is used when the fan is meant to specify a piece-wise linear (or tropical rational) function. The function value on the ith ray of the fan is listed in the ith row of the matrix.

LINEALITY_VALUES is a *Matrix* with just one column. It is used when the fan is meant to specify a piece-wise linear (or tropical rational) function. The function value on the ith generator of the lineality space (stored in LINEALITY_SPACE) is listed in the ith row of the matrix.

Besides sections listed above, the sections MAXIMAL_CONES_ORBITS, CONES_ORBITS and MULTIPLICITIES_ORBITS are introduced when doing symmetric computations with the `--symmetry` option. These sections are analogous to MAXIMAL_CONES, CONES and MULTIPLICITIES except that they operate on the level of orbits of cones with respect to the symmetry rather than cones.

A.8 Polyhedral cones

The string representation of a polyhedral cone starts with

```
_application PolyhedralCone
_version 2.2
_type PolyhedralCone
```

After this follows the properties. For polyhedral cones they are as follows.

AMBIENT_DIM — see previous section.

DIM is a *Cardinal* whose value is the dimension of the cone.

IMPLIED_EQUATIONS is a *Matrix* whose rows form a basis of the space of linear forms vanishing on the cone.

LINEALITY_DIM — see previous section.

LINEALITY_SPACE — see previous section.

FACETS is a *Matrix* which contains an outer normal vector for each facet of the cone.

RELATIVE_INTERIOR_POINT is a *Vector* in the relative interior of the cone.

B Application list

This section contains the full list of programs in Gfan. For each program its help file is listed. The help file of a program can also be displayed by specifying the `--help` option when running the program. Besides the options listed in this section all programs have options **--log1**, **--log2**,... which tell Gfan how much information to write to "standard error" while a computation is running. **These options are VERY USEFUL when you wish to know if Gfan is making any progress in its computation.**

Additional options which can be used for all programs, but which are not listed in the following subsections are:

--stdin value Specify a file to use as input instead of reading from the standard input.

--stdout value Specify a file to write output to instead of writing to the standard output.

--xml To let polyhedral fans be output in an XML format instead of in the text format. (The XML files are not readable by Gfan.)

B.1 gfan_bases

This is a program for computing all reduced Gröbner bases of a polynomial ideal. It takes the ring and a generating set for the ideal as input. By default the enumeration is done by an almost memoryless reverse search. If the ideal is symmetric the symmetry option is useful and enumeration will be done up to symmetry using a breadth first search. The program needs a starting Gröbner basis to do its computations. If the -g option is not specified it will compute one using Buchberger's algorithm.

Options:

-g Tells the program that the input is already a Gröbner basis (with the initial term of each polynomial being the first ones listed). Use this option if it takes too much time to compute the starting (standard degree lexicographic) Gröbner basis and the input is already a Gröbner basis.

--symmetry Tells the program to read in generators for a group of symmetries (subgroup of S_n) after having read in the ideal. The program checks that the ideal stays fixed when permuting the variables with respect to elements in the group. The program uses breadth first search to compute the set of reduced Gröbner bases up to symmetry with respect to the specified subgroup.

-e Echo. Output the generators for the symmetry group.

--disableSymmetryTest When using --symmetry this option will disable the check that the group read off from the input actually is a symmetry group with respect to the input ideal.

--parameters value With this option you can specify how many variables to treat as parameters instead of variables. This makes it possible to do computations where the coefficient field is the field of rational functions in the parameters. This does not work well at the moment.

B.2 gfan_buchberger

This program computes a reduced lexicographic Gröbner basis of the polynomial ideal given as input. The default behavior is to use Buchberger's algorithm. The ordering of the variables is $a > b > c...$ (assuming that the ring is Q[a,b,c,...]).
Options:

-w Compute a Gröbner basis with respect to a degree lexicographic order with $a > b > c...$ instead. The degrees are given by a weight vector which is read from the input after the generating set has been read.

-r Use the reverse lexicographic order (or the reverse lexicographic order as a tie breaker if -w is used). The input must be homogeneous if the pure reverse lexicographic order is chosen. Ignored if -W is used.

-W Do a Gröbner walk. The input must be a minimal Gröbner basis. If -W is used -w is ignored.

-g Do a generic Gröbner walk. The input must be homogeneous and must be a minimal Gröbner basis with respect to the reverse lexicographic term order. The target term order is always lexicographic. The -W option must be used.

--parameters value With this option you can specify how many variables to treat as parameters instead of variables. This makes it possible to do computations where the coefficient field is the field of rational functions in the parameters. This does not work well at the moment.

B.3 gfan_combinerays

This program combines rays from the specified polymake file by adding according to a list of vectors of indices given on the standard input.
Options:

-i value Specify the name of the input file.

--section value Specify a section of the polymake file to use as input, rather than standard input.

B.4 gfan_doesidealcontain

This program takes a marked Gröbner basis of an ideal I and a set of polynomials on its input and tests if the polynomial set is contained in I by applying the division algorithm for each element. The output is 1 for true and 0 for false.
Options:

--remainder Tell the program to output the remainders of the divisions rather than outputting 0 or 1.

--multiplier Reads in a polynomial that will be multiplied to the polynomial to be divided before doing the division.

B.5 gfan_fancommonrefinement

This program takes two polyhedral fans and computes their common refinement.
Options:

-i1 value Specify the name of the first input file.

-i2 value Specify the name of the second input file.

B.6 gfan_fanhomology

This program takes a polyhedral fan and computes its reduced homology groups. Of course the support of a fan is contractible, so what is really computed is the reduced homology groups of the support of the fan after quotienting out with the lineality space and intersecting with a sphere. Notice that taking the quotient with the lineality space results in an inverted suspension which just results in a shift of the reduced homology groups.
Options:

-i value Specify the name of the input file.

--no-optimize Disable preprocessing of boundary maps before doing lattice computations.

B.7 gfan_fanlink

This program takes a polyhedral fan and a vector and computes the link of the polyhedral fan around that vertex. The link will have lineality space dimension equal to the dimension of the relative open polyhedral cone of the original fan containing the vector.
Options:

-i value Specify the name of the input file.

--symmetry Reads in a fan stored with symmetry. The generators of the symmetry group must be given on the standard input.

--star Computes the star instead. The star is defined as the smallest polyhedral fan containing all cones of the original fan containing the vector.

B.8 gfan_fanproduct

This program takes two polyhedral fans and computes their product.
Options:

-i1 value Specify the name of the first input file.

-i2 value Specify the name of the second input file.

B.9 gfan_fansubfan

This program takes a polyhedral fan and a list of vectors and computes the smallest subfan of the fan having the list of vectors in its support.
Options:

-i value Specify the name of the input file.

--symmetry Reads in the cone stored with symmetry. The generators of the symmetry group must be given on the standard input.

B.10 gfan_genericlinearchange

This program takes a list of polynomials and performs a generic linear change of coordinates by introducing nxn new variables.

B.11 gfan_groebnercone

This program computes a Gröbner cone. Three different cases are handled. The input may be a marked reduced Gröbner basis in which case its Gröbner cone is computed. The input may be just a marked minimal basis in which case the cone computed is not a Gröbner cone in the usual sense but smaller. (These cones are described in [Fukuda, Jensen, Lauritzen, Thomas]). The third possible case is that the Gröbner cone is possibly lower dimensional and given by a pair of Gröbner bases as it is useful to do for tropical varieties, see option --pair. The facets of the cone can be read off in section FACETS and the equations in section IMPLIED_EQUATIONS.
Options:

--restrict Add an inequality for each coordinate, so that the the cone is restricted to the non-negative orthant.

--pair The Gröbner cone is given by a pair of compatible Gröbner bases. The first basis is for the initial ideal and the second for the ideal itself. See the tropical section of the manual.

--asfan Writes the cone as a polyhedral fan with all its faces instead. In this way the extreme rays of the cone are also computed.

--vectorinput Compute a cone given list of inequalities rather than a Gröbner cone. The input is an integer which specifies the dimension of the ambient space, a list of inequalities given as vectors and a list of equations.

B.12 gfan_groebnerfan

This is a program for computing the Gröbner fan of a polynomial ideal as a polyhedral complex. It takes a generating set for the ideal as input. If the ideal is symmetric the symmetry option is useful and enumeration will be done up to symmetry. The program needs a starting Gröbner basis to do its computations. If the -g option is not specified it will compute one using Buchberger's algorithm.
Options:

-g Tells the program that the input is already a Gröbner basis (with the initial term of each polynomial being the first ones listed). Use this option if it takes too much time to compute the starting (standard degree lexicographic) Gröbner basis and the input is already a Gröbner basis.

--symmetry Tells the program to read in generators for a group of symmetries (subgroup of S_n) after having read in the ideal. The program checks that the ideal stays fixed when permuting the variables with respect to elements in the group. The program uses breadth first search to compute the set of reduced Gröbner bases up to symmetry with respect to the specified subgroup.

--disableSymmetryTest When using --symmetry this option will disable the check that the group read off from the input actually is a symmetry group with respect to the input ideal.

--restrictingfan value Specify the name of a file containing a polyhedral fan in Polymake format. The computation of the Gröbner fan will be restricted to this fan. If the --symmetry option is used then this restricting fan must be invariant under the symmetry and the orbits in the file must be with respect to the specified group of symmetries. The orbits of maximal cones of the file are then read in rather than the maximal cones.

--parameters value With this option you can specify how many variables to treat as parameters instead of variables. This makes it possible to do computations where the coefficient field is the field of rational functions in the parameters. This does not work well at the moment.

--nocones Tells the program not to output the CONES and MAXIMAL_CONES sections, but still output CONES_COMPRESSED and MAXIMAL_CONES_COMPRESSED if --symmetry is used.

B.13 gfan_homogeneityspace

This program computes the homogeneity space of a list of polynomials - as a cone. Thus generators for the homogeneity space are found in the section LINEALITY_SPACE. If you wish the homogeneity space of an ideal you should first compute a set of homogeneous generators and call the program on these. A reduced Gröbner basis will always suffice for this purpose.

B.14 gfan_homogenize

This program homogenises a list of polynomials by introducing an extra variable. The name of the variable to be introduced is read from the input after the list of polynomials. Without the -w option the homogenisation is done with respect to total degree. Example: Input: Q[x,y]{y-1} z Output: Q[x,y,z]{y-z}
Options:

-i Treat input as an ideal. This will make the program compute the homogenisation of the input ideal. This is done by computing a degree Gröbner basis and homogenising it.

-w Specify a homogenisation vector. The length of the vector must be the same as the number of variables in the ring. The vector is read from the input after the list of polynomials.

-H Let the name of the new variable be H rather than reading in a name from the input.

B.15 gfan_initialforms

This program converts a list of polynomials to a list of their initial forms with respect to the vector given after the list.
Options:

--ideal Treat input as an ideal. This will make the program compute the initial ideal of the ideal generated by the input polynomials. The computation is done by computing a Gröbner basis with respect to the given vector. The vector must be positive or the input polynomials must be homogeneous in a positive grading. None of these conditions are checked by the program.

--pair Produce a pair of polynomial lists. Used together with --ideal this option will also write a compatible reduced Gröbner basis for the input ideal to the output. This is useful for finding the Gröbner cone of a non-monomial initial ideal.

--mark If the --pair option is and the --ideal option is not used this option will still make sure that the second output basis is marked consistently with the vector.

--list Read in a list of vectors instead of a single vector and produce a list of polynomial sets as output.

B.16 gfan_interactive

This is a program for doing interactive walks in the Gröbner fan of an ideal. The input is a Gröbner basis defining the starting Gröbner cone of the walk. The program will list all flippable facets of the Gröbner cone and ask the user to choose one. The user types in the index (number) of the facet in the list. The program will walk through the selected facet and display the new Gröbner basis and a list of new facet normals for the user to choose from. Since the program reads the user's choices through the the standard input it is recommended not to redirect the standard input for this program.
Options:

-L Latex mode. The program will try to show the current Gröbner basis in a readable form by invoking LaTeX and xdvi.

-x Exit immediately.

-f Tell the program to list the flipped reduced Gröbner basis of the initial ideal for each flippable wall in the current Gröbner cone.

-w Tell the program to list (a Gröbner basis with respect to the current term order for) the initial ideal for each flippable wall in the current Gröbner cone.

-i Tell the program to list the defining set of inequalities of the non-restricted Gröbner cone as a set of vectors after having listed the current Gröbner basis.

-W Print weight vector. This will make the program print an interior vector of the current Gröbner cone and a relative interior point for each flippable facet of the current Gröbner cone.

--tropical Traverse a tropical variety interactively.

B.17 gfan_ismarkedgroebnerbasis

This program checks if a set of marked polynomials is a Gröbner basis with respect to its marking. First it is checked if the markings are consistent with respect to a positive vector. Then Buchberger's S-criterion is checked. The output is boolean value.

B.18 gfan_krulldimension

Takes an ideal I and computes the Krull dimension of R/I where R is the polynomial ring. This is done by first computing a Gröbner basis.
Options:

-g Tell the program that the input is already a reduced Gröbner basis.

B.19 gfan_latticeideal

This program computes the lattice ideal of a lattice. The input is a list of generators for the lattice.
Options:

-t Compute the toric ideal of the matrix whose rows are given on the input instead.

--convert Does not do any computation, but just converts the vectors to binomials.

B.20 gfan_leadingterms

This program converts a list of polynomials to a list of their leading terms.
Options:

-m Do the same thing for a list of polynomial sets. That is, output the set of sets of leading terms.

B.21 gfan_list

This program lists all subcommands of the Gfan installation.
Options:

--hidden Show hidden commands which are not officially supported.

B.22 gfan_markpolynomialset

This program marks a set of polynomials with respect to the vector given at the end of the input, meaning that the largest terms are moved to the front. In case of a tie the lexicographic term order with $a > b > c$... is used to break it.

B.23 gfan_minkowskisum

This is a program for computing the normal fan of the Minkowski sum of the Newton polytopes of a list of polynomials.
Options:

--symmetry Tells the program to read in generators for a group of symmetries (subgroup of S_n) after having read in the ideal. The program checks that the ideal stays fixed when permuting the variables with respect to elements in the group. The program uses breadth first search to compute the set of reduced Gröbner bases up to symmetry with respect to the specified subgroup.

--disableSymmetryTest When using --symmetry this option will disable the check that the group read off from the input actually is a symmetry group with respect to the input ideal.

--nocones Tell the program to not list cones in the output.

B.24 gfan_minors

This program will generate the r*r minors of a d*n matrix of indeterminates.
Options:

-r value Specify r.

-d value Specify d.

-n value Specify n.

-M2 Use Macaulay2 conventions for order of variables.

--names Assign names to the minors.

--dressian Produce tropical defining the Dressian(3,n) instead. (The signs may not be correct, that is the equations may not be Pluecker relations.)

--pluckersymmetries Do nothing but produce symmetry generators for the Pluecker ideal.

--symmetry Produces a list of generators for the group of symmetries keeping the set of minors fixed. (Only without --names).

--parametrize Parametrize the set of d times n matrices of Barvinok rank less than or equal to r-1 by a list of tropical polynomials.

B.25 gfan_mixedvolume

This program computes the mixed volume of the Newton polytopes of a list of polynomials. The ring is specified on the input. After this follows the list of polynomials.

B.26 gfan_overintegers

This program is an experimental implementation of Gröbner bases for ideals in Z[x_1,...,x_n]. Several operations are supported by specifying the appropriate option: (1) computation of the reduced Gröbner basis with respect to a given vector (tiebroken lexicographically), (2) computation of an initial ideal, (3) computation of the Gröbner fan, (4) computation of a single Gröbner cone. Since Gfan only knows polynomial rings with coefficients being elements of a field, the ideal is specified by giving a set of polynomials in the polynomial ring Q[x_1,...,x_n]. That is, by using Q instead of Z when specifying the ring. The ideal MUST BE HOMOGENEOUS (in a positive grading) for computation of the Gröbner fan. Non-homogeneous ideals are allowed for the other computations if the specified weight vectors are positive. NOTE: This program is experimental and expected to change behaviour in future releases, so don't write your SAGE and M2 interfaces just yet.
Options:

--groebnerBasis Asks the program to compute a marked Gröbner basis with respect to a weight vector tie-broken lexicographically. The input order is: Ring ideal vector.

--initialIdeal Asks the program to compute an initial ideal with respect to a vector. The input order is: Ring ideal vector.

--groebnerFan Asks the program to compute the Gröbner fan. The input order is: Ring ideal.

--groebnerCone Asks the program to compute a single Gröbner cone containing the specified vector in its relative interior. The output is stored as a fan. The input order is: Ring ideal vector.

-m For the operations taking a vector as input, read in a list of vectors instead, and perform the operation for each vector in the list.

-g Tells the program that the input is already a Gröbner basis (with the initial term of each polynomial being the first ones listed). Use this option if the usual --groebnerFan is too slow.

B.27 gfan_padic

This program is an experimental implementation of p-adic Gröbner bases as proposed by Diane Maclagan. Several operations are supported by specifying the appropriate option: (1) computation of Gröbner basis with respect to a given vector (tiebroken lexicographically), (2) computation of the p-adic initial ideal, (3) computation of the p-adic Gröbner complex as defined by Maclagan and Sturmfels, (4) computation of a single polyhedron of the p-adic Gröbner complex. The input ideal should be an ideal of the polynomial ring with coefficient field Q. The valuation is specified with the option -p. The ideal MUST BE HOMOGENEOUS (in a positive grading). Since gfan can only handle fans and not polyhedral complexes in general, what is computed as the Gröbner complex is actually the "fan over" the complex - in other words, the first coordinate is supposed to be 1 in the output fan. Similarly, the weight vectors must be specified in an homogeneous way, for example by adding an additional 1 entry as first coordinate. (If fractions are needed, use the entry as a common denominator.) NOTE: This program is experimental and expected to change behaviour in future releases, so don't write your SAGE and M2 interfaces just yet. In particular this program uses the tropical minimum-convention!!
Options:

-p value Defines the prime used for the valuation.

--groebnerBasis Asks the program to compute a marked Gröbner basis with respect to a weight vector (tie-broken lexicographically). The input order is: Ring ideal vector.

--initialIdeal Asks the program to compute an initial ideal with respect to a vector. The input order is: Ring ideal vector.

--groebnerComplex Asks the program to compute the p-adic Gröbner complex. The input order is: Ring ideal.

--groebnerPolyhedron Asks the program to compute a single polyhedron of the Gröbner complex containing the specified vector in its relative interior. The output is stored as a fan. The input order is: Ring ideal vector.

-m For the operations taking a vector as input, read in a list of vectors instead, and perform the operation for each vector in the list.

B.28 gfan_polynomialsetunion

This program computes the union of a list of polynomial sets given as input. The polynomials must all belong to the same ring. The ring is specified on the input. After this follows the list of polynomial sets.
Options:

-s Sort output by degree.

B.29 gfan_render

This program renders a Gröbner fan as an xfig file. To be more precise, the input is the list of all reduced Gröbner bases of an ideal. The output is a drawing of the Gröbner fan intersected with a triangle. The corners of the triangle are (1,0,0) to the right, (0,1,0) to the left and (0,0,1) at the top. If there are more than three variables in the ring these coordinates are extended with zeros. It is possible to shift the 1 entry cyclic with the option --shiftVariables.
Options:

-L Make the triangle larger so that the shape of the Gröbner region appears.

--shiftVariables value Shift the positions of the variables in the drawing. For example with the value equal to 1 the corners will be right: (0,1,0,0,...), left: (0,0,1,0,...) and top: (0,0,0,1,...). The shifting is done modulo the number of variables in the polynomial ring. The default value is 0.

B.30 gfan_renderstaircase

This program renders a staircase diagram of a monomial initial ideal to an xfig file. The input is a Gröbner basis of a (not necessarily monomial) polynomial ideal. The initial ideal is given by the leading terms in the Gröbner basis. Using the -m option it is possible to render more than one staircase diagram. The program only works for ideals in a polynomial ring with three variables.
Options:

-m Read multiple ideals from the input. The ideals are given as a list of lists of polynomials. For each polynomial list in the list a staircase diagram is drawn.

-d value Specifies the number of boxes being shown along each axis. Be sure that this number is large enough to give a correct picture of the standard monomials. The default value is 8.

-w value Width. Specifies the number of staircase diagrams per row in the xfig file. The default value is 5.

B.31 gfan_saturation

This program computes the saturation of the input ideal with the product of the variables x_1,...,x_n. The ideal does not have to be homogeneous.
Options:

-h Tell the program that the input is a homogeneous ideal (with homogeneous generators).

--noideal Do not treat input as an ideal but just factor out common monomial factors of the input polynomials.

B.32 gfan_secondaryfan

This program computes the secondary fan of a vector configuration. The configuration is given as an ordered list of vectors. In order to compute the secondary fan of a point configuration an additional coordinate of ones must be added. For example {(1,0),(1,1),(1,2),(1,3)}.
Options:

--unimodular Use heuristics to search for unimodular triangulation rather than computing the complete secondary fan

--scale value Assuming that the first coordinate of each vector is 1, this option will take the polytope in the 1 plane and scale it. The point configuration will be all lattice points in that scaled polytope. The polytope must have maximal dimension. When this option is used the vector configuration must have full rank. This option may be removed in the future.

--restrictingfan value Specify the name of a file containing a polyhedral fan in Polymake format. The computation of the Secondary fan will be restricted to this fan. If the --symmetry option is used then this restricting fan must be invariant under the symmetry and the orbits in the file must be with respect to the specified group of symmetries. The orbits of maximal cones of the file are then read in rather than the maximal cones.

--symmetry Tells the program to read in generators for a group of symmetries (subgroup of S_n) after having read in the vector configuration. The program checks that the configuration stays fixed when permuting the variables with respect to elements in the group. The output is grouped according to the symmetry.

--nocones Tells the program not to output the CONES and MAXIMAL_CONES sections, but still output CONES_COMPRESSED and MAXIMAL_CONES_COMPRESSED if --symmetry is used.

B.33 gfan_stats

This program takes a list of reduced Gröbner bases for the same ideal and computes various statistics. The following information is listed: the number of bases in the input, the number of variables, the dimension of the homogeneity space, the maximal total degree of any polynomial in the input and the minimal total degree of any basis in the input, the maximal number of polynomials and terms in a basis in the input.

B.34 gfan_substitute

This program changes the variable names of a polynomial ring. The input is a polynomial ring, a polynomial set in the ring and a new polynomial ring with the same coefficient field but different variable names. The output is the polynomial set written with the variable names of the second polynomial ring. Example: Input: Q[a,b,c,d]{2a-3b,c+d}Q[b,a,c,x] Output: Q[b,a,c,x]{2*b-3*a,c+x}

B.35 gfan_symmetries

This program computes the symmetries of a polynomial ideal. The program is slow, so think before using it. Use --symmetry to give hints about which subgroup of the symmetry group could be useful. The program checks each element of the specified subgroup to see if it preserves the ideal.
Options:

--symmetry Specify subgroup to be searched for permutations keeping the ideal fixed.

--symsigns Specify for each generator of the group specified wiht --symmetry an element of $\{-1, +1\}^n$ which by its multiplication on the variables together with the permutation is expected to keep the ideal fixed.

B.36 gfan_tolatex

This program converts ASCII math to TeX math. The data-type is specified by the options.
Options:

-h Add a header to the output. Using this option the output will be LaTeXable right away.

--polynomialset_ The data to be converted is a list of polynomials.

--polynomialsetlist_ The data to be converted is a list of lists of polynomials.

B.37 gfan_topolyhedralfan

This program takes a list of reduced Gröbner bases and produces the fan of all faces of these. In this way by giving the complete list of reduced Gröbner bases, the Gröbner fan can be computed as a polyhedral complex. The option --restrict lets the user choose between computing the Gröbner fan or the restricted Gröbner fan.

Options:

--restrict Add an inequality for each coordinate, so that the the cones are restricted to the non-negative orthant.

--symmetry Tell the program to read in generators for a group of symmetries (subgroup of S_n) after having read in the ring. The output is grouped according to these symmetries. Only one representative for each orbit is needed on the input.

B.38 gfan_tropicalbasis

This program computes a tropical basis for an ideal defining a tropical curve. Defining a tropical curve means that the Krull dimension of R/I is at most 1 + the dimension of the homogeneity space of I where R is the polynomial ring. The input is a generating set for the ideal. If the input is not homogeneous option -h must be used.

Options:

-h Homogenise the input before computing a tropical basis and dehomogenise the output. This is needed if the input generators are not already homogeneous.

B.39 gfan_tropicalbruteforce

This program takes a marked reduced Gröbner basis for a homogeneous ideal and computes the tropical variety of the ideal as a subfan of the Gröbner fan. The program is slow but works for any homogeneous ideal. If you know that your ideal is prime over the complex numbers or you simply know that its tropical variety is pure and connected in codimension one then use gfan_tropicalstartingcone and gfan_tropicaltraverse instead.

B.40 gfan_tropicalevaluation

This program evaluates a tropical polynomial function in a given set of points.

B.41 gfan_tropicalfunction

This program takes a polynomial and tropicalizes it. The output is piecewise linear function represented by a fan whose cones are the linear regions. Each ray of the fan gets the value of the tropical function assigned to it. In other words this program computes the normal fan of the Newton polytope of the input polynomial with additional information.

B.42 gfan_tropicalhypersurface

This program computes the tropical hypersurface defined by a principal ideal. The input is the polynomial ring followed by a set containing just a generator of the ideal.

B.43 gfan_tropicalintersection

This program computes the set theoretical intersection of a set of tropical hypersurfaces (or to be precise, their common refinement as a fan). The input is a list of polynomials with each polynomial defining a hypersurface. Considering tropical hypersurfaces as fans, the intersection can be computed as the common refinement of these. Thus the output is a fan whose support is the intersection of the tropical hypersurfaces.
Options:

-t Note that the input polynomials generate an ideal. This option will make the program choose a relative interior point for each listed output cone and check if its initial ideal contains a monomial. The actual check is done on a homogenization of the input ideal, but this does not affect the result.

--tplane This option intersects the resulting fan with the plane x_0=-1, where x_0 is the first variable. To simplify the implementation the output is actually the common refinement with the non-negative half space. This means that "stuff at infinity" (where x_0=0) is not removed.

--symmetryPrinting Parse a group of symmetries after the input has been read. Used when printing with --incidence.

--symmetryExploit Restrict computation to the closed lexicographic fundamental domain of the specified symmetry group. This overwrites --restrict.

--nocones Tells the program not to output the CONES and MAXIMAL_CONES sections, but still output CONES_COMPRESSED and MAXIMAL_CONES_COMPRESSED if --symmetry is used.

--restrict Restrict the computation to a full-dimensional cone given by a list of marked polynomials. The cone is the closure of all weight vectors choosing these marked terms.

--stable Find the stable intersection of the input polynomials using tropical intersection theory. This can be slow. Most other options are ignored.

B.44 gfan_tropicallifting

This program is part of the Puiseux lifting algorithm implemented in Gfan and Singular. The Singular part of the implementation can be found in:

Anders Nedergaard Jensen, Hannah Markwig, Thomas Markwig: tropical.lib. A SINGULAR 3.0 library for computations in tropical geometry, 2007

See also

http://www.mathematik.uni-kl.de/~keilen/de/tropical.html

and the paper

Jensen, Markwig, Markwig: "An algorithm for lifting points in a tropical variety".

Example:

Run Singular from the directory where tropical.lib is located. Give the following sequence of commands to Singular:

LIB "tropical.lib"; ring R=0,(t,x,y,z),dp; ideal i=-y2t4+x2,yt3+xz+y; intvec w=1,-2,0,2; list L=tropicallifting(i,w,3); displaytropicallifting(L,"subst"); This produces a Puiseux series solution to i with valuation (2,0,-2)

Options:

--noMult Disable the multiplicity computation.

-n value Number of variables that should have negative weight.

-c Only output a list of vectors being the possible choices.

B.45 gfan_tropicallinearspace

This program generates tropical equations for a tropical linear space in the Speyer sense given the tropical Pluecker coordinates as input.

Options:

-d value Specify d.

-n value Specify n.

--trees list the boundary trees (assumes d=3)

B.46 gfan_tropicalmultiplicity

This program computes the multiplicity of a tropical cone given a marked reduced Gröbner basis for its initial ideal.

B.47 gfan_tropicalrank

This program will compute the tropical rank of matrix given as input. Tropical addition is MAXIMUM.
Options:

--kapranov Compute Kapranov rank instead of tropical rank.

--determinant Compute the tropical determinant instead.

B.48 gfan_tropicalstartingcone

This program attempts to compute a starting pair of marked reduced Gröbner bases to be used as input for gfan_tropicaltraverse. If unsuccessful the program will say so. The input is a homogeneous ideal whose tropical variety is a pure d-dimensional polyhedral complex.
Options:

-g Tell the program that the input is already a reduced Gröbner basis.

-d Output dimension information to standard error.

--stable Find starting cone in the stable intersection or, equivalently, pretend that the coefficients are genereric.

B.49 gfan_tropicaltraverse

This program computes a polyhedral fan representation of the tropical variety of a homogeneous prime ideal I. Let d be the Krull dimension of I and let ω be a relative interior point of d-dimensional Gröbner cone contained in the tropical variety. The input for this program is a pair of marked reduced Gröbner bases with respect to the term order represented by ω, tie-broken in some way. The first one is for the initial ideal $in\_\omega(I)$ the second one for I itself. The pair is the starting point for a traversal of the d-dimensional Gröbner cones contained in the tropical variety. If the ideal is not prime but with the tropical variety still being pure d-dimensional the program will only compute a codimension 1 connected component of the tropical variety.
Options:

--symmetry Do computations up to symmetry and group the output accordingly. If this option is used the program will read in a list of generators for a symmetry group after the pair of Gröbner bases have been read. Two advantages of using this option is that the output is nicely grouped and that the computation can be done faster.

--symsigns Specify for each generator of the symmetry group an element of $\{-1, +1\}^n$ which by its multiplication on the variables together with the permutation will keep the ideal fixed. The vectors are given as the rows of a matrix.

--nocones Tells the program not to output the CONES and MAXIMAL_CONES sections, but still output CONES_ORBITS and MAXIMAL_CONES_ORBITS if --symmetry is used.

--disableSymmetryTest When using --symmetry this option will disable the check that the group read off from the input actually is a symmetry group with respect to the input ideal.

--stable Traverse the stable intersection or, equivalently, pretend that the coefficients are genereric.

B.50 gfan_tropicalweildivisor

This program computes the tropical Weil divisor of piecewise linear (or tropical rational) function on a tropical k-cycle. See the Gfan manual for more information.
Options:

-i1 value Specify the name of the Polymake input file containing the k-cycle.

-i2 value Specify the name of the Polymake input file containing the piecewise linear function.

B.51 gfan_version

This program writes out version information of the Gfan installation.

References

[1] 4ti2 team. 4ti2—a software package for algebraic, geometric and combinatorial problems on linear spaces. Available at www.4ti2.de.

[2] Lars Allermann and Johannes Rau. First steps in tropical intersection theory. *Mathematische Zeitschrift*, 2009.

[3] David Avis and Komei Fukuda. A basis enumeration algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Computational Geometry*, 8:295–313, 1992.

[4] Robert Bieri and J. R. J. Groves. The geometry of the set of characters induced by valuations. *J. Reine Angew. Math.*, 347:168–195, 1984.

[5] T. Bogart, A. N. Jensen, D. Speyer, B. Sturmfels, and R. R. Thomas. Computing tropical varieties. *J. Symbolic Comput.*, 42(1-2):54–73, 2007, math.AG/0507563.

[6] Bruno Buchberger. An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal, 1965. PhD-thesis (German).

[7] S. Collart, M. Kalkbrener, and D. Mall. Converting bases with the Gröbner walk. *J. Symbolic Comput.*, 24(3-4):465–469, 1997. Computational algebra and number theory (London, 1993).

[8] Komei Fukuda. *cddlib reference manual, cddlib Version 094b.* Swiss Federal Institute of Technology, Lausanne and Zürich, Switzerland, 2005. `http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html`.

[9] Komei Fukuda, Anders Jensen, and Rekha Thomas. Computing Gröbner fans. *Mathematics of Computation, to appear*, 2007, math.AC/0509544.

[10] Ewgenij Gawrilow and Michael Joswig. polymake: a framework for analyzing convex polytopes. In Gil Kalai and Günter M. Ziegler, editors, *Polytopes — Combinatorics and Computation*, pages 43–74. Birkhäuser, 2000.

[11] Torbjörn Granlund et al. GNU multiple precision arithmetic library 4.1.2, December 2002. `http://swox.com/gmp/`.

[12] Birkett Huber and Rekha R. Thomas. Computing Gröbner fans of toric ideals. *Experimental Mathematics*, 9(3/4):321–331, 2000.

[13] Anders Jensen. CaTS, a software system for toric state polytopes. Available at http://www.soopadoopa.dk/anders/cats/cats.html.

[14] Anders N. Jensen. Traversing symmetric polyhedral fans. to appear in "Mathematical Software - ICMS 2010", 2010.

[15] Anders Nedergaard Jensen. *Algorithmic aspects of Gröbner fans and tropical varieties*. PhD thesis, University of Aarhus, 2007. `http://www.imf.au.dk/publs?id=655`.

[16] Anders Nedergaard Jensen. A non-regular Gröbner fan. *Discrete Comput. Geom.*, 37(3):443–453, 2007, math.CO/0501352.

[17] Hannah Markwig, Thomas Markwig, and Anders Jensen. An algorithm for lifting points in a tropical variety. 2007, math.AG/0705.2441.

[18] Teo Mora and Lorenzo Robbiano. The Gröbner fan of an ideal. *J. Symbolic Comput.*, 6(2-3):183–208, 1988. Computational aspects of commutative algebra.

[19] Jörg Rambau. TOPCOM: Triangulations of point configurations and oriented matroids. *ZIB report*, 02-17, 2002.

[20] David Speyer and Bernd Sturmfels. The tropical Grassmannian. *Adv. Geom.*, 4(3):389–411, 2004, math.AG/0304218.

[21] Bernd Sturmfels. *Gröbner bases and convex polytopes*, volume 8 of *University Lecture Series*. American Mathematical Society, Providence, RI, 1996.

[22] Bernd Sturmfels. *Solving systems of polynomial equations*, volume 97 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC, 2002.

[23] Roland Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus*. PhD thesis, Technische Universität Berlin, 1996. `http://www.zib.de/Publications/abstracts/TR-96-09/`.