CaTS version 2.2: A User's Manual

Anders Nedergaard Jensen

28th November 2003

Abstract

CaTS is a software package whose main functions enumerate (1) all reduced Gröbner bases of a lattice ideal, and (2) all monomial A-graded ideals that are flip-connected to a given one. Several variants of these enumeration algorithms are supported such as restricting the above enumerations to all initial ideals or monomial A-graded ideals with a fixed radical. CaTS supports several additional commands among which the highlights are:
• `cats_interactive` which allows interactive walks on the edge graph of the state polytope of a toric ideal,
• `cats_fiber` which enumerates all monomials of a fixed A-degree, and
• `cats_monomialideal2standardpairs` which computes the standard pair decomposition of a monomial ideal.
The full list of commands can be found in appendix A.

CaTS began as a re-implementation of TiGERS, a software package to compute state polytopes of toric ideals, written by Birkett Huber based on algorithms in [Huber & Thomas]. The first version was developed for Jensen's Masters thesis at the University of Aarhus in Denmark. The program has been expanded since then. CaTS is faster than TiGERS (by a factor of 30 on some examples [Jensen]) and has several additional functionalities. To access the full range of commands, CaTS needs to be linked to the packages `TOPCOM` [Rambau], `4ti2` [Hemmecke & Hemmecke], `Normaliz` [Bruns & Koch], `Macaulay 2` [Grayson & Stillman], `SoPlex` [Wunderling] and `cdd` [Fukuda]. The last two packages are used to solve linear programs and have considerably increased the numerical stability of CaTS. The basic installation only requires linking to the LP solvers.

Contents

1 Introduction

CaTS is a computer program for enumerating all reduced Gröbner bases, or equivalently, all monomial initial ideals of a *lattice ideal* in $\mathbf{k}[x_1,\ldots,x_n] =: \mathbf{k}[\mathbf{x}]$ where $\mathbf{k}$ is a field with $char(\mathbf{k}) \neq 2$. The lattice ideal of a sublattice $\mathcal{L} \subseteq \mathbb{Z}^n$, is

$$I_{\mathcal{L}} = \langle \mathbf{x}^{\mathbf{u}^+} - \mathbf{x}^{\mathbf{u}^-} \, : \, \mathbf{u} = \mathbf{u}^+ - \mathbf{u}^- \in \mathcal{L}, \, \mathbf{u}^+, \mathbf{u}^- \in \mathbb{N}^n \rangle \subseteq \mathbf{k}[\mathbf{x}].$$

The ideal $I_{\mathcal{L}}$ is *toric* when $\mathcal{L}$ is a saturated lattice. See [Sturmfels] for the theory and applications of toric ideals.

The reduced Gröbner bases of a polynomial ideal I are in bijection with the vertices of the *state polyhedron* of I. When I is homogeneous, this polyhedron is bounded and known as the *state polytope* of I [Bayer & Morrison]. The *normal fan* of the state polyhedron is the *Gröbner fan* of I [Mora & Robbiano]. The main algorithm in CaTS traverses the edge graph of the state polyhedron of a lattice ideal and outputs the reduced Gröbner bases corresponding to the vertices of the graph. See Chapters 1-3 in [Sturmfels] for the theory and computation of state polytopes and Gröbner fans of general polynomial ideals. When the ideals have more structure as in the case of lattice ideals, the general algorithms can be specialized. CaTS is based on such specialized algorithms that were developed for toric ideals in [Huber & Thomas] and first implemented in the software package TiGERS [Huber] written by Birkett Huber. With minor modifications the same algorithms work for all lattice ideals. The basic strategy of these algorithms is to enumerate all reduced Gröbner bases of the ideal by walking on the edge graph of the state polyhedron and visiting every vertex during the walk.

Walking from one vertex of the state polyhedron to a neighboring vertex along an edge of the polyhedron is the same as walking from one maximal Gröbner cone through a facet into the neighboring maximal Gröbner cone. Thus at any given vertex of the state polytope we need algorithms for the following local computations:

1. A method for finding the facets of the Gröbner cone of a given reduced Gröbner basis.

2. A method for computing a neighbouring reduced Gröbner basis given one Gröbner basis and a facet of its Gröbner cone.

Globally, we use a graph search algorithm to enumerate all the vertices of the (edge graph of the) state polyhedron. The main features of the specialized algorithms for 1. and 2. are:

- All polynomials in the input, output and intermediate computations are binomials or monomials with ± 1 or 0 as coefficients.

- Monomials and binomials can be represented by vectors.

- The local change procedure to convert one Gröbner basis to another is combinatorial and does not need to know weight vectors that induce either Gröbner bases. This avoids numerical issues that arise with computing and storing weight vectors.

- The facets of a Gröbner cone are computed from the elements of the Gröbner basis via linear programming. CaTS can be linked to the linear programming solvers `SoPlex` [Wunderling], `cddlib` [Fukuda] or Huber's implementation of the simplex algorithm from TiGERS [Huber] (included in this package). With cddlib exact arithmetic is possible.

Example 1.1 Let $A = [1\ 2\ 3\ 6] \in \mathbb{Z}^{1\times 4}$ and let $\prec$ be the graded reverse lexicographic order with the grading $(1,2,3,6)^T$ and with $a^6 \succ b^3 \succ c^2 \succ d$. There are 12 reduced Gröbner bases for $I_A \subset \mathbf{k}[a,b,c,d]$, numbered $\mathcal{G}_0$ to $\mathcal{G}_{11}$ and listed below. $\mathcal{G}_1$ is the reduced Gröbner basis corresponding to $\prec$.

$$\mathcal{G}_0 = \{ \boxed{\boxed{b - \underline{a^2}}}_{\,9},\ \boxed{c - \underline{a^3}}_{\,10},\ \boxed{d - \underline{a^6}}_{\,11} \}$$

$$\mathcal{G}_1 = \{ \boxed{\underline{c^2} - d}_{\,7},\ \underline{ab} - c,\ \boxed{\underline{b^2} - ac}_{\,4},\ \boxed{\underline{a^2} - b}\mathrm{b}_2 \}$$

$$\mathcal{G}_2 = \{ \boxed{\underline{c^2} - d}_{\,10},\ \boxed{\underline{a^3} - c}_{\,11},\ \boxed{\boxed{b - \underline{a^2}}}_{\,1} \}$$

$$\mathcal{G}_3 = \{ \boxed{\underline{b^3} - c^2}_{\,8},\ \underline{ab} - c,\ \underline{a^2} - b,\ \boxed{\boxed{ac - \underline{b^2}}}_{\,7},\ \boxed{d - \underline{c^2}}_{\,5}, \}$$

$$\mathcal{G}_4 = \{ \underline{b^3} - d,\ \boxed{\underline{b^2c} - ad}_{\,5},\ \underline{c^2} - d,\ \boxed{\underline{ab} - c}_{\,6},\ \underline{a^2} - b,\ \boxed{\boxed{ac - \underline{b^2}}}_{\,1} \}$$

$$\mathcal{G}_5 = \{ \boxed{\underline{b^3} - d}_{\,8},\ \boxed{\underline{c^2} - d}_{\,3},\ \underline{ab} - c,\ \underline{a^2} - b,\ ac - \underline{b^2},\ \boxed{\boxed{ad - \underline{b^2c}}}_{\,4}, \}$$

$$\mathcal{G}_6 = \{ \boxed{\underline{b^3} - d}_{\,9},\ \boxed{\underline{a^2} - b}_{\,11},\ \boxed{\boxed{c - \underline{ab}}}_{\,4} \}$$

$$\mathcal{G}_7 = \{ \underline{ab} - c,\ \boxed{\underline{b^2} - ac}_{\,3},\ \boxed{\underline{a^2} - b}_{\,10},\ \boxed{\boxed{d - \underline{c^2}}}_{\,1} \}$$

$$\mathcal{G}_8 = \{ \boxed{\underline{ab} - c}_{\,9},\ \underline{a^2} - b,\ ac - \underline{b^2},\ \boxed{\boxed{c^2 - \underline{b^3}}}_{\,3},\ \boxed{d - \underline{b^3}}_{\,5} \}$$

$$\mathcal{G}_9 = \{ \boxed{\underline{a^2} - b}_{\,0},\ \boxed{\boxed{c - \underline{ab}}}_{\,8},\ \boxed{d - \underline{b^3}}_{\,6} \}$$

$$\mathcal{G}_{10} = \{ \boxed{\underline{a^3} - c}_{\,0},\ \boxed{\boxed{b - \underline{a^2}}}_{\,7},\ \boxed{d - \underline{c^2}}_{\,2} \}$$

$$\mathcal{G}_{11} = \{ \boxed{\underline{a^6} - d}_{\,0},\ \boxed{\boxed{b - \underline{a^2}}}_{\,6},\ \boxed{c - \underline{a^3}}_{\,2} \}$$

This ideal has a three dimensional state polytope and hence its edge graph is planar (see Figure 1.1). The boxed binomials in a reduced Gröbner basis, give the facets of the corresponding Gröbner cone. The number that appears to the bottom right of a facet binomial (outside the box) indexes the Gröbner cone that shares this facet with the current Gröbner cone. The double boxes will be explained later.

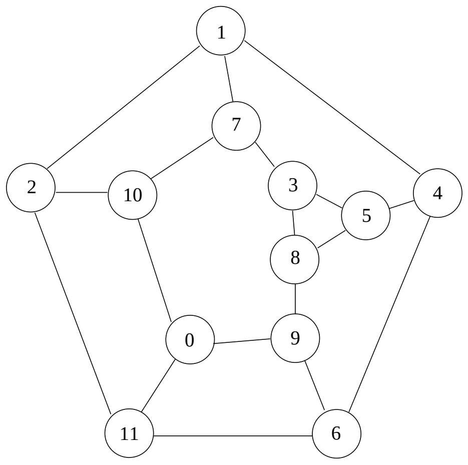

Figure 1: The edge graph of the state polytope from Example 1.1

1.1 Algorithms

The following is a brief description of some of the algorithms in CaTS. Details can be found in [Huber & Thomas],[Jensen] and [Sturmfels, Chapter 3]. We list the option that needs to be added to the basic command in CaTS to invoke the stand alone algorithms in this list.

1.1.1 Finding the facets of a Gröbner cone

Given a reduced Gröbner basis $\mathcal{G}_\prec(I)$ of I with respect to a term order $\prec$ we need to find the facets of its Gröbner cone which is the closure of the set $\{\omega \in \mathbb{R}^n : \mathcal{G}_\prec(I) = \mathcal{G}_\omega(I)\} = \{\omega \in \mathbb{R}^n : \alpha \cdot \omega \geq \beta \cdot \omega$ for all $\mathbf{x}^\alpha - \mathbf{x}^\beta \in \mathcal{G}_\prec(I)$ with leading term $\mathbf{x}^\alpha\}$. This follows from Proposition 2.3 [Sturmfels] which states that:

$$in_\omega(I) = in_\prec(I) \Leftrightarrow \text{ for all } \mathbf{x}^\alpha - \mathbf{x}^\beta \in \mathcal{G}_\prec(I) : \alpha \cdot \omega > \beta \cdot \omega.$$

Thus each binomial in $\mathcal{G}_\prec(I)$ gives rise to one valid inequality of the cone. Some of these equalities are essential (i.e., correspond to facets of the Gröbner cone). If an inequality is essential we call its binomial a *facet binomial.* By the Farkas lemma of linear programming, the facet normals of the Gröbner cone are exactly the primitive generators of the extreme rays of the cone polar to the Gröbner cone. This polar cone is generated by $\{\alpha - \beta : \mathbf{x}^\alpha - \mathbf{x}^\beta \in \mathcal{G}_\prec(I)\}$. CaTS uses linear programming to find these generators by using Huber's implementation of the simplex method in TiGERS or `cddlib` [Fukuda] or SoPlex [Wunderling]. In Example 1.1, $\mathcal{G}_1$ has three facet binomials: $c^2 - d, b^2 - ac, a^2 - b$; $ab - c$ is not a facet binomial.

To improve speed, two extra tests called the "superset test" and the "subset test" are used when finding facets [Huber & Thomas]. The subset test simply tests if the ith coordinate is positive only for a single vector in the list $\{\alpha - \beta : \mathbf{x}^\alpha - \mathbf{x}^\beta \in \mathcal{G}_\prec(I)\}$. If so, this vector must be an extreme ray of the polar cone.

1.1.2 Flip - the local procedure to change Gröbner bases

Suppose that a reduced Gröbner basis $\mathcal{G}_\prec(I)$ has the facet binomial $\mathbf{x}^\alpha - \mathbf{x}^\beta$. If the Gröbner fan is complete there is another Gröbner basis $\mathcal{G}_{\prec'}(I)$ whose Gröbner cone shares this facet. We wish to compute $\mathcal{G}_{\prec'}(I)$.

Let ω be a vector in the relative interior of the common facet and $\prec_\omega$ be the term order that refines the order induced by ω by $\prec$. Then $in_{\prec_\omega}(I) = in_\prec(in_\omega(I)) = in_\prec(I)$ is the initial ideal of $in_\omega(I)$ with respect to $\prec$. The ideal $in_\omega(I)$ has precisely two initial ideals: $in_\prec(I)$ and $in_{\prec'}(I)$. The flip algorithm in [Huber & Thomas] allows us to compute the ideals $in_\omega(I)$ and $in_{\prec'}(I)$ given $\mathcal{G}_\prec(I)$ without actually computing ω or knowing $\prec$ and $\prec'$. It is a special case of Subroutine 3.7 in [Sturmfels]. The procedure is purely combinatorial. Finally, the algorithm "lifts" $in_{\prec'}(I)$ to the Gröbner basis $\mathcal{G}_{\prec'}(I)$. In Example 1.1 flipping across the facet binomial $c^2 - d$ in $\mathcal{G}_1$ we get $\mathcal{G}_7$.

1.1.3 Graph search algorithms: Exhaustive search (option `-t1, -t3`)

Enumerating the graph is easy now that we know how to find edges and compute neighbouring vertices. We simply apply an exhaustive enumeration algorithm for graphs. One improvement is directing the edge graph with respect to a generic vector $\omega \in \mathbb{R}^n$ or a term order. Since the graph is the edge graph of a polytope we get a unique sink in this new directed graph and no cycles. Starting at the sink and always moving in the opposite direction on the edges we can enumerate the graph. Having ordered the graph in this way we only need to do half as many "flips" as before — usually we would have flipped twice for each edge in the graph, but now we flip only once. See the figure in Example 1.2; $\mathcal{G}_1$ is the unique sink here.

1.1.4 Graph search algorithms: Reverse search (option `-t2`)

When using exhaustive search, we need to store all computed vertices in memory and each time we compute a new vertex we must check if it has already been computed. We want to avoid storing the set of computed vertices as the output size can be enormous. This can be done via the memory-free reverse search technique in [Avis & Fukuda]. We remove edges from the oriented graph in the previous section until we are left with a tree. See Example 1.2. For each vertex we need exactly one outgoing edge except for the sink. To remove edges we choose a rule. For example we could choose an outgoing edge in the oriented graph to be kept exactly if the initial term of the corresponding binomial is maximal with respect to the lexicographic order among all outgoing edges from that vertex. Checking if an edge in the oriented graph is a part of the tree can be done locally.

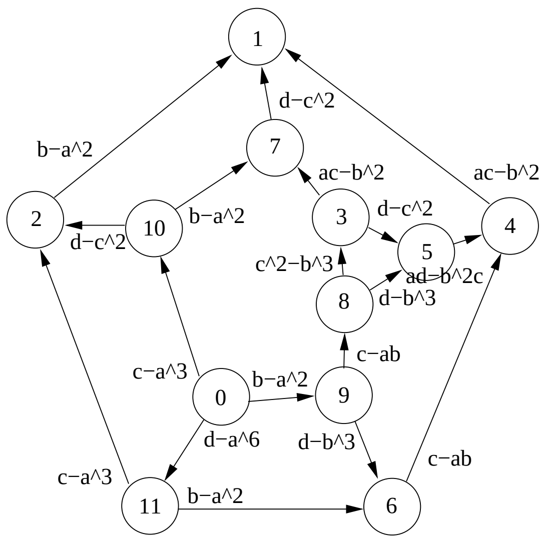

Figure 2: Oriented graph

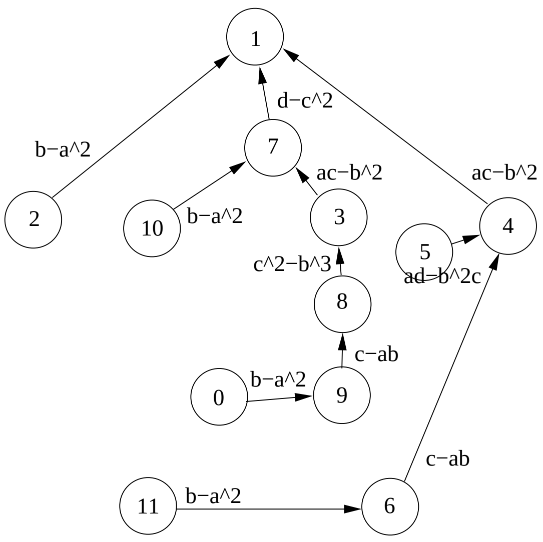

Figure 3: Oriented tree

We now apply a tree traversal algorithm and we no longer have to store the set of computed vertices. One way to think of the tree induced by ω, is as the union of the paths we walk along when when computing $\mathcal{G}_\omega(I)$ via the *Gröbner walk* procedure [Sturmfels] starting at any reduced Gröbner basis.

Example 1.2 (Example 1.1 continued.) Recall that in the list of reduced Gröbner bases, the facet binomials are boxed and subscripted by the index of the neighboring Gröbner basis that shares this facet.

Each binomial has an underlined term. This term is initial with respect to our graded reverse lexicographic term order $\prec$. Each edge in the graph is labeled by the facet binomial corresponding to that edge. The edges are oriented with respect to $\prec$ and $\mathcal{G}_\prec(I)$ becomes the unique sink.

We now delete edges to get the tree. The remaining edges correspond to the binomials in Example 1.1 marked with a double box. For each reduced Gröbner basis this is the facet binomial (among facet binomials corresponding to outgoing edges) with the largest initial term with respect to the lexicographic term order.

1.1.5 Monomial initial ideals with a fixed radical (option `-t5`)

Usually CaTS computes all monomial initial ideals of a lattice ideal. However, CaTS could compute just those monomial initial ideals with a prespecified radical if needed. The option `-t5` in CaTS allows this computation. A reduced Gröbner basis whose initial ideal has the specified radical is needed as input. The set of monomial initial ideals of I with this radical is connected in the edge graph of the state polytope. Hence all CaTS needs to do is to exhaustively search this part of the graph, and for each edge, check if it leads to an initial ideal with the same or a different radical. Fixing the radical is extremely important when combining CaTS with TOPCOM. See section 4.4.

It is possible to improve this algorithm using reverse search on the subgraph. However, this requires computing the term order for one of the monomial initial ideals with this radical. CaTS does not have this option at the moment.

1.1.6 Computing monomial A-graded ideals (options `-t7`,`-t8`)

An A-graded ideal is an ideal with the same A-graded Hilbert function as I_A, the toric ideal defined by A [Sturmfels, Chapter 10]. All initial ideals of I_A are A-graded, but in general there may be many more monomial A-graded ideals of I_A. One way of computing more monomial A-graded ideals is by allowing flips through binomials that are not facet binomials. There are restrictions on which binomials are flippable in this new sense — only those that pass the superset test are eligible (see section 1.1.1 and [Maclagan & Thomas]). The flip algorithm behaves nicely on these flips except that the "lifting" procedure from Section 1.1.2 may fail when we allow these generalized flips. The lifting algorithm was used before to lift an initial ideal to its Gröbner basis which was needed to continue the graph search. In this new setup, we need to lift a monomial A-graded ideal to a "fake Gröbner basis" which is a list of binomials such that for each binomial, the positive term is a minimal generator of the monomial A-graded ideal and negative term is the unique standard monomial of the same A-degree. CaTS has two ways of solving this. The first is to compute the unique standard monomial of a given A-degree by searching the set of all monomials of that A-degree. This amounts to searching the lattice points in a polytope. CaTS can compute lattice points in polytopes using the command `cats_fiber`. See Section 5 for details. The second method is to use `4ti2` [Hemmecke & Hemmecke] to compute the *Graver basis* of I_A and then pick the right binomial to include in the fake Gröbner basis from the Graver basis. CaTS can compute all monomial A-graded ideals that are connected by flips to the monomial initial ideals of I_A, using exhaustive search. There are examples for which all monomial A-graded ideals are not connected by flips [Santos]. Hence CaTS may not find all monomial A-graded ideals.

1.1.7 Other algorithms

A few other algorithms are also implemented in CaTS.

- Observing that reduced Gröbner bases of lattice ideals are generated by pure binomials we represent the binomials as vectors. Lattice ideals allow cancellation: $\mathbf{x}^\alpha\mathbf{x}^v - \mathbf{x}^\beta\mathbf{x}^v \in I \Rightarrow \mathbf{x}^\alpha - \mathbf{x}^\beta \in I$. These two observations lead us to the saturating Buchberger and the saturating division algorithms implemented in CaTS. They work on vectors alone and cancel out common terms when possible. We expect these algorithms to be faster than the original ones. See [Jensen] for details on these algorithms and why they can be used.

- Algorithm 12.3 in [Sturmfels] for computing a first Gröbner basis of a toric ideal I_A given the matrix A, is implemented in CaTS.
- The Lenstra-Lenstra-Lovász algorithm for computing reduced lattice bases is needed in [Sturmfels, Algorithms 12.3] and is also implemented in CaTS.
- An algorithm for computing the standard pairs [Sturmfels, Trung & Vogel] of a monomial ideal is implemented. This algorithm differs from published algorithms for this computation. The monomial ideal $\{0\} \subset \mathbf{k}[x_1, \ldots, x_n]$ has only a single standard pair. The algorithm works by successively adding a monomial generator to the ideal and updating the set of standard pairs accordingly.

2 Installation

This section describes how to install CaTS on a Linux / Unix system with a modern version of gcc. CaTS has been compiled successfully with gcc version 3.2.

Download the file `cats2.2.tar.gz` from the CaTS homepage located at:

`http://www.soopadoopa.dk/anders/cats/cats.html.`

2.1 Basic installation

Decompress `cats2.2.tar.gz` by typing

```
gzip -d cats2.2.tar.gz
```

in the shell. This produces a `.tar` file which is extracted using

```
tar -x<cats2.2.tar
```

This creates a new directory named `cats2.2.tar` . Go to this directory by typing

```
cd cats2.2
```

You can now compile CaTS by typing

```
make
```

This produces the executable file `cats`. Type `./cats -h` in the shell to test it. It is recommended that you continue reading — the following section will give you more information on how to install the program including instructions on how to add a reliable LP-solver.

2.2 Installation to invoke advanced features

Having compiled the program using the instructions above, you can now install cats in `/usr/local/bin` by typing

```
make install
```

This will make cats accessible from any directory. You must be root to run `make install`.

The version compiled in the description above uses the simplex implementation in TiGERS. Due to floating point precision problems **it is highly recommended that you download and install SoPlex or cddlib** see [Wunderling] and [Fukuda] which contain standard LP solvers. There are examples for which TiGERS runs into numerical problems arising from its in-built LP solver.

2.2.1 SoPlex

To compile CaTS with SoPlex you should install SoPlex version 1.2.1 downloadable at `http://www.zib.de/Optimization/Software/Soplex/soplex.php`. Follow the instructions given. Or extract the SoPlex `.tar` file into some directory, go to the newly created directory and type

```
make COMP=gnu OPT=opt
```

This will work in most cases. If not follow the instructions given with SoPlex.

When SoPlex is installed go to the directory `cats2.2` where you extracted CaTS. In this directory make a symbolic link to the directory where you installed SoPlex by typing

```
ln -s /this/is/the/path/to/soplex-1.2.1 soplex-1.2.1
```

You are now ready to compile CaTS with SoPlex. Do this by typing
`make soplex=enabled`
If you want the new re-compiled CaTS to be installed in `/usr/local/bin` you should type
`make install`
again. To test, type `./cats -h`. CaTS should show its help file and a list of LP solvers linked to the program.

2.2.2 Cddlib

Cddlib has the advantage that it can do exact arithmetics. For CaTS to use it you must be root when installing cddlib on your system (when running `make install`). Download the file `cddlib-093b.tar.gz` from

```
http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html
```

into a directory. Decompress the file using `gzip -d cddlib-093b.tar.gz` and extract the tar archive using `tar -xvf cddlib-093b.tar`. Change directory to the newly created directory `cddlib-093b` and run `./configure` , `make` and `make install`. You must be root to execute the last command. Return to the directory CaTS was extracted to and run `make cdd=enabled`. This option can also be combined with `soplex=enabled`.

2.3 Additional applications

The CaTS package contains several specialized applications. They are compiled by moving to the sub directory `apps` and running `make`:

```
cd apps
make
```

The options from the previous sections (`soplex=enabled` and `cdd=enabled`) can also be used here. To install the applications in `/usr/local/bin` type:

```
make install
```

You must be root to do this. The names of all the CaTS applications start with `cats_`. If you installed the files in `/usr/local/bin`, then type `cats<TAB>` in your shell to list the installed CaTS applications.

2.4 Invoked programs

If the enumeration of monomial A-graded ideals is done using option `-g1` which needs the Graver basis, CaTS requires `4ti2` [Hemmecke & Hemmecke] to be installed on the system. The command `graver` from `4ti2` must be in the path. The same is true for option `-L`. It is **very important** to use 4ti2 version 1.1 and not version 1.0.

Some of the additional application programs also depend on other programs. The programs needed are `Normaliz` [Bruns & Koch], `TOPCOM` [Rambau] and `Macaulay 2` [Grayson & Stillman]. The executable files `normaliz`, `points2triangs` and `M2` must be in the path for these applications to work correctly.

When CaTS communicates with these programs it will use files in the current directory. They are called something like `normalizdata.in`. Hence it is important not to start simultaneous copies of CaTS invoked from the same directory. This is only a problem when using CaTS in a way that makes use of these programs.

The complete list of functionalities available in CaTS and the programs they need are listed at the end of the manual.

3 The main functions

The main CaTS program called `cats` computes the Gröbner fan of a lattice ideal and writes the output to a file. Example:

```
./cats -p1+
```

followed by

```
{(1)(2)(3)}
```

writes all reduced Gröbner basis of the toric ideal I_A to the file `output_list` where $A = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$.

3.1 Options

The following is the complete list of options in CaTS (* is the default setting):
Traversal algorithm:

```
-t1  Exhaustive Search
-t2 *Reverse Search
-t3  Oriented Exhaustive Search
-t5  Exhaustive Search, for initial ideals with a fixed radical
-t7  Exhaustive Search to find A-graded monomial ideals
-t8  A-Graded fixed radical exhaustive search
```

Linear programming library:

```
-fsoplex  SoPlex
-fcdd     cddlib
-fcddgmp  cddlib with gmp for exact arithmetics
-fhuber   Huber's implementation of the simplex method
```

Vertex printing:

```
-p0  No vertex printing
-p1  Print facet binomials and reduced Groebner bases when found
-p1+ Print reduced Groebner bases when found (minimalistic version)
-p2  Print facet binomials and initial ideals when found
-p2+ Print initial ideals when found (minimalistic version)
-qs  Split printing, print radical by radical, together with -p2+ and -w only
```

Output file name options:

```
-or FILENAME  file for the results (time, size of graph,
              floating point status ....) Default: output_results
-ol FILENAME  file for listing the graph during traversal
              (see options -p1, -p2 and -p3) Default: output_list
```

Input file name options:

```
-i FILENAME   input file
-w FILENAME   weight file, containing weight vectors for each radical
              (regular triangulation)
```

Lattice ideal options:

```
-L  The ideal considered should be the lattice ideal of the lattice
    generated by the rows of the input matrix rather than the toric
    ideal of the input matrix
```

Performance options:

```
-T  Print internal timers when done
```

Help:

```
-h Print the help file
```

Options `-t1`, `-t2`, `-t3`, `-t5` `-t7` and `-t8` are mentioned in Section 1.1. The options `-t7` and `-t8` are further explained in section 4.2. Option `-w` is explained in section 4.4. The examples in Section 4 illustrate the use of many of these options.

3.2 Input formats

When the program is started with `./cats -p1+` it will ask for input from the keyboard. The input specifies a lattice ideal for which the Gröbner fan should be computed. The input can either be a matrix or a reduced Gröbner basis. If the input is a matrix, the Gröbner fan of the associated toric ideal is computed. If the input is a reduced Gröbner basis, it is used to initiate the computation of the Gröbner fan of the ideal it generates.

For compatibility, three input formats are supported. The toric ideal associated to the 3x6 matrix:

$$A = \begin{pmatrix} 2 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 2 \end{pmatrix}$$

can be given to the program in the following three ways:

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

(The brackets may be of any type and columns may be separated by ',') or

```
M:
{3 6 :
2 1 0 1 0 0
0 1 2 0 1 0
0 0 0 1 1 2
}
```

or

```
R:6
G:{e^2-cf, de-b*f, c*d^2-b^2*f, b*e-c*d, a*f-d^2, a*e-b*d, a*c-b^2}
```

Here `R` stands for `ring` and `G` for `Gröbner basis`. The number after `R` is the number of variables in the polynomial ring. This version of CaTS cannot handle more than 52 variables. The variables are named `a, b, ..., z, A, B, ...,Z`.

If you store your matrix / Gröbner basis in a file called `example.dat` you can make CaTS read it directly by typing:

```
./cats -p1+ <example.dat
```

The output is written to `output_list`.

3.3 Printing formats

CaTS supports several output formats. Which output format to use is selected with the `-p` option. In the following the options are demonstrated on the example $A = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$:

- `-p1` (facet binomials and reduced Gröbner bases)

```
Vtx: 0 (2 facets/3 binomials/degree 2)
{# b^2-a*c, # a^2-b}
{b^2-a*c, a*b-c, a^2-b}
Vtx: 1 (2 facets/4 binomials/degree 3)
{# b^3-c^2, # a*c-b^2}
{a^2-b, a*b-c, b^3-c^2, a*c-b^2}
Vtx: 2 (2 facets/4 binomials/degree 2)
{# a*b-c, # c^2-b^3}
{a^2-b, a*b-c, a*c-b^2, c^2-b^3}
Vtx: 3 (2 facets/2 binomials/degree 2)
{# a^2-b, # c-a*b}
{a^2-b, c-a*b}
Vtx: 4 (2 facets/2 binomials/degree 1)
{# b-a^2, # c-a^3}
{b-a^2, c-a^3}
Vtx: 5 (2 facets/2 binomials/degree 3)
{# a^3-c, # b-a^2}
{a^3-c, b-a^2}
```

- `-p1+` (the list of reduced Gröbner bases)

```
{{b^2-a*c, a*b-c, a^2-b},
{a^2-b, a*b-c, b^3-c^2, a*c-b^2},
{a^2-b, a*b-c, a*c-b^2, c^2-b^3},
{a^2-b, c-a*b},
{b-a^2, c-a^3},
{a^3-c, b-a^2}}
```

- `-p2` (facet binomials and initial ideals)

```
Vtx: 0 (2 facets/3 binomials/degree 2)
    Initial ideal:{b^2, a*b, a^2}
  Facet Binomials:{# b^2-a*c,# a^2-b}
Vtx: 1 (2 facets/4 binomials/degree 3)
    Initial ideal:{a^2, a*b, b^3, a*c}
  Facet Binomials:{# b^3-c^2,# a*c-b^2}
Vtx: 2 (2 facets/4 binomials/degree 2)
    Initial ideal:{a^2, a*b, a*c, c^2}
  Facet Binomials:{# a*b-c,# c^2-b^3}
Vtx: 3 (2 facets/2 binomials/degree 2)
    Initial ideal:{a^2, c}
  Facet Binomials:{# a^2-b,# c-a*b}
Vtx: 4 (2 facets/2 binomials/degree 1)
    Initial ideal:{b, c}
  Facet Binomials:{# b-a^2,# c-a^3}
Vtx: 5 (2 facets/2 binomials/degree 3)
    Initial ideal:{a^3, b}
  Facet Binomials:{# a^3-c,# b-a^2}
```

- `-p2+` (list of initial ideals)

```
{{b^2, a*b, a^2},
{a^2, a*b, b^3, a*c},
{a^2, a*b, a*c, c^2},
{a^2, c},
{b, c},
{a^3, b}}
```

3.4 Files

It is possible to change the output files in CaTS using the options `-or` and `-ol`. If these options are not used, the default output files are `output_results` and `output_list`. CaTS lists these filenames when starting up.

By default, CaTS gets its input from the standard input (keyboard). It is possible to read the input from a file using the < of the shell like this `./cats < inputfile`. However, it is sometimes more useful to use the -i option. If in addition you add the option `-e`, then CaTS will use the name of the input file as the prefix of all output files. This is useful if you have a large collection of examples. For example, if the input file is named `V_2_4.dat` and the example is input via the command:

```
./cats -p1 -e -i V_2_4.dat
```

then the files get named `V_2_4.lists`, `V_2_4.results`, and `V_2_4.heights`. The `.heights` file is explained in Section 4.4.

4 Examples

4.1 Simple examples

We will explain how to use CaTS for computations with toric ideals that are homogeneous with respect to a positive vector. Let

$$A = \begin{pmatrix} 2 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 2 \end{pmatrix}.$$

1. To count the number of monomial initial ideals of the toric ideal I_A type

   ```
   ./cats
   ```

   When asked for input type

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

   The program will write the number of vertices and edges of the state polytope to the file `output_results`.

2. If we want to use exhaustive search for the above computation we should have started the program with:

   ```
   ./cats -t1
   ```

3. If we want to produce a list of the initial ideals we should have started the program with:

   ```
   ./cats -p2+
   ```

   The list will be written to the file `output_list`.

4. If we get tired of retyping the matrix every time we run the program, we can write it to the file `example.dat` and start CaTS using

   ```
   ./cats < example.dat
   ```

   or

   ```
   ./cats -i example.dat
   ```

5. It is possible to give CaTS a reduced Gröbner basis of the toric ideal instead of the matrix defining the ideal: type

   ```
   ./cats
   ```

   followed by

   ```
   R:6
   G:{e^2-c*f, d*e-b*f, d^2-a*f, c*d-b*e, b*d-a*e, b^2-a*c}
   ```

   or write the above lines to a file and use the `-i` option.

6. To compute all monomial initial ideals with the same radical as the initial ideal corresponding to the input Gröbner basis type

   ```
   ./cats -t5 -p2+
   ```

   followed by

   ```
   R:6
   G:{e^2-c*f, d*e-b*f, d^2-a*f, c*d-b*e, b*d-a*e, b^2-a*c}
   ```

4.2 Monomial A-graded ideals

Let $A = \begin{pmatrix} 1 & 2 & 3 & 7 & 8 & 9 \end{pmatrix}$ in the all the following examples.

1. To compute all monomial A-graded ideals flip-connected to the initial ideals of I_A type:

   ```
       ./cats -t7
   ```

   Type `{(1),(2),(3),(7),(8),(9)}` when asked for input. If you wanted the ideals written to the file `output_list` you should have used:

   ```
       ./cats -t7 -p1
   ```

2. To compute all monomial A-graded ideals flip-connected to a given monomial A-graded ideal using *fiber search* for lifting to "fake Gröbner bases" type:

   ```
       ./cats -t7 -A
   ```

   Type `{(1),(2),(3),(7),(8),(9)}` when asked for input. When asked for more input type

   ```
       {e^8-a*f^7, d*e^6-a*f^6, d^2*e^4-a*f^5, d^4-a*f^3,
       d^3*e-b*f^3, c^3-f, b*d-f, c*d-a*f, a*e-f, b*e-a*f,
       c*e-b*f, b*c^2-e, a*d-e, a*c^2-d, a*b-c, b^2-a*c,
       a^2-b, d*f-e^2, c^2*f-d*e, b*c*f-d^2, c*f^2-d^3,
       b*f^4-d^2*e^3}
   ```

   This last input is a "fake Gröbner basis". The positive terms generate a monomial A-graded ideal and for each binomial, the second term is exactly the standard monomial of the ideal with the same multi-degree as the positive term. This is not necessarily a Gröbner basis.

3. The same as above but using the Graver basis from `4ti2` for lifting

   ```
   ./cats -t7 -A -g1
   ```

   If you are having trouble using the `-g1` option this could be because you don't have the right version of `4ti2` installed on your system. Try to do the computation without `-g1` which is very slow or install `4ti2` version 1.1.

4. Same as 2. but reading the matrix and the Gröbner basis from a file named `inputMatrix.dat`:

   ```
   ./cats -t7 -A <inputMatrix.dat
   ```

   or

   ```
   ./cats -t7 -A -i inputMatrix.dat
   ```

   The file `inputMatrix.dat` should look like this:

   ```
   {(1),(2),(3),(7),(8),(9)}
   {e^8-a*f^7, d*e^6-a*f^6, d^2*e^4-a*f^5, d^4-a*f^3,
   d^3*e-b*f^3, c^3-f, b*d-f, c*d-a*f, a*e-f, b*e-a*f,
   c*e-b*f, b*c^2-e, a*d-e, a*c^2-d, a*b-c, b^2-a*c,
   a^2-b, d*f-e^2, c^2*f-d*e, b*c*f-d^2, c*f^2-d^3,
   b*f^4-d^2*e^3}
   ```

5. **Fixed radical:** Compute all monomial A-graded ideals flip-connected to a given A-graded monomial ideal with the property that their radical is the same as the radical of the given A-graded ideal. Type:

   ```
   ./cats -t8 -A
   ```

   Type `{(1),(2),(3),(7),(8),(9)}` when asked for input. When asked for more input type

   ```
   (e^6-c*f^5, d*f-e^2, d*e^4-c*f^4, d^2*e^2-c*f^3,
   d^3-c*f^2, c*e^3-f^3, c*d-b*e, c^2-b^3, b*f-c*e, b*e^2-f^2,
   b*d-f, b^2*e-c*f, b^2*c-d, b^4-e, a*f-b*e, a*e-f, a*d-e,
   a*c-b^2, a*b-c, a^2-b)
   ```

   The positive terms generate an A-graded ideal and for each binomial the second term is exactly the standard monomial of the ideal with the same multi-degree as the initial term. This is not necessarily a Gröbner basis. All monomial A-graded ideals that are flip-connected to this monomial A-graded ideal and has the same radical will be enumerated. The computed monomial initial ideals are written to `output_list`.

4.3 Lattice ideal examples

The lattice ideals seen so far are all toric ideals homogeneous with respect to a positive grading. CaTS can also do computations on general lattice ideals. We look at the lattice ideal $I = \langle a^2 - b, c^3 - b^2, b^2c - a, abc - 1, b^4 - ac^2, ab^3 - c^2 \rangle$ associated to the lattice generated by $\{(2,-1,0)^T, (0,-2,3)^T, (1,-2,-1)^T\}$.

1. To compute all monomial initial ideals of I type

   ```
   ./cats -p2+
   ```

   followed by

   ```
   R:3
   G:{a^2-b,c^3-b^2,b^2c-a,abc-1,b^4-ac^2,ab^3-c^2}
   ```

   or

   ```
   ./cats -p2+ -L
   ```

   followed by

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

   The rows of the input matrix are the lattice generators. Since CaTS reads a matrix column by column we must "transpose" the lattice genrators by hand before giving them to CaTS. The ideal has 13 monomial initial ideals.

   If you are having trouble using the `-L` option this could be because you don't have the right version of `4ti2` installed on your system. It is recommended that you either install `4ti2` version 1.1 or give the Gröbner basis as input without the `-L` option.

4.4 Fixed radical computations and TOPCOM

The package TOPCOM [Rambau] can be used for computing the regular triangulations of a point configuration. For a matrix A the radicals of the monomial initial ideals of I_A are in bijection with the regular triangulations of the columns of A. CaTS can compute all initial ideals of I_A with a fixed radical with option `-t5` given just one reduced Gröbner basis whose initial ideal has this radical, as input.

Using the output from TOPCOM, we could use CaTS to find initial ideals radical by radical. TOPCOM can output regular triangulations upto symmetry. If this is used as input to CaTS, we get initial ideals under these symmetry classes of triangulations. This can tremendously cut down the amount of computations needed. CaTS can do these computation automatically using the output from TOPCOM. Use TOPCOM to compute the triangulations of A with the command "points2triangs –heights". For each symmetry class of triangulations, CaTS needs:

- a weight vector inducing the regular triangulation
- the number of triangulations in the class

The weight vector is needed to produce a Gröbner basis whose initial ideal has radical corresponding to the regular triangulation. The number of triangulations is just used for counting the total number of monomial initial ideals. Example:

```
 points2triangs --heights <examples/V_2_4.dat >examples/V_2_4.heights
 ./cats -w -i examples/V_2_4.dat -p2+ -qs -e
```

This produces the file `examples/V_2_4.list` containing the initial ideals radical by radical. Only one radical from each symmetry class is used. The last part of the example file `V_2_4.dat` contains symmetries. CaTS will ignore this part. See [Rambau] for details on how to specify symmetries. Note that CaTS often computes initial ideals under the same radical that are symmetric. So the output from CaTS is not exactly an "enumeration of initial ideals up to symmetry", but the point is to restrict to one representative from each symmetry class at the level of radicals or regular triangulations.

5 Additional functions

This section briefly describes the additional applications included in the CaTS software package. Some of these applications contains a help file. To view this you should use option `-h`. Example:

```
./cats_interactive -h
```

5.1 The interactive mode

The program `cats_interactive` makes it possible to walk in the Gröbner fan interactively. You start the program with

```
./cats_interactive
```

Now type in a matrix/point configuration. Example: `((1)(2)(3)(4))`. The program computes a reduced Gröbner basis with respect to reverse lexicographic order with $a > b > c > d > \ldots$ where a, b, c, d are the variables of the toric ideal.

The program will print this reduced Gröbner basis, the radical of its initial ideal, the standard pairs of its initial ideal and the facets of the Gröbner cone. You now have a few options:

- Type a number <enter> to flip the facet binomial indexed by that number.
- Type 'c' <enter> to test for Cohen Macaulayness of the initial ideal.
- Type 'b' <enter> to take a step back in the path.

The program is useful when looking for initial ideals without embedded primes — we want to flip a facet binomial leading to an initial ideal with fewer standard pairs as we expect this initial ideal to be closer to an initial ideal without embedded primes (there is no theorem to this effect, just an empirical observation). To easily find such a facet, the facets have been marked with the number of standard pairs for the ideal they flip to. Also the number of standard pairs for the current initial ideal, the facet we came from and whether a facet binomial flips to a different radical/triangulation is shown.

5.2 Enumerating lattice points in polytopes

The program `cats_fiber` can enumerate the fiber of the map

$$\phi : \mathbb{N}^n \to \mathbb{Z}^d \ : \ u \mapsto Au$$

for reasonably small examples. The input is a matrix of the form described in section 3.2 followed by a vector in the fiber. Example:

```
./cats_fiber
```

with input:

```
{(3)(4)(5)}
(0,0,3)
```

gives output:

```
{
(0,0,3),
(2,1,1),
(1,3,0),
(5,0,0)
}
```

Thus $\phi^{-1}(15) = \{(0,0,3),(2,1,1),(1,3,0),(5,0,0)\}$ when $A = (3\,4\,5)$ — in other words, there are four ways of writing 15 as a sum of numbers from the set $\{3,4,5\}$: $15 = 5+5+5 = 3+3+4+5 = 3+4+4+4 = 3+3+3+3+3$.

5.3 VectorList2MonomialList

The program `cats_vectorlist2monomiallist` takes a list of vectors and transforms it into a list of monomials. It is usually used in combination with `cats_fiber`. Example:

```
./cats_fiber | ./cats_vectorlist2monomiallist
```

with input:

```
{(3)(4)(5)}
(0,0,3)
```

gives output:

```
(c^3,
a^2*b*c,
a*b^3,
a^5
)
```

5.4 IsDeltaNormal

The program `cats_isdeltanormal` can check if a point configuration has a regular triangulation Δ for which it is Δ-normal with respect to the lattice generated by the points [Hoşten & Thomas 2003]. The program does the following

- computes a lattice basis of the generated lattice using the LLL-algorithm [Lenstra et al.]
- rewrites each of the input vectors as an integer combination of the lattice basis elements.
- computes all regular triangulations of the new point configuration using TOPCOM [Rambau]
- for each triangulation Δ it tests if the point configuration is Δ-normal. This is done using `normaliz` [Bruns & Koch] on each triangle in Δ.

Rewriting the points in another basis allows us to check normality of each triangle with respect to $\mathbb{Z}^n$ instead of the lattice generated by the configuration (which `normaliz` is unable to do). The computations done with `normaliz` are cached to improve speed.

The input is a point configuration in the usual format and the output is "true" or "false".

5.5 Graver

The program `cats_graver` can compute the graver basis of a toric ideal. The input should be a matrix (see 3.2). The program simply calls 4ti2's `graver` command. The reason for using the CaTS version of this program is that the input and output are written in formats that CaTS supports. The Graver basis is part of the input to the computation of monomial A-graded ideals.

5.6 BinomialList2Degree

The program `cats_binomiallist2degree` computes the maximal degree of any binomial in a list of binomials. The program is usually used in combination with `cats_graver`.

5.7 MonomialIdeal2StandardPairs

The program `cats_monomialideal2standardpairs` computes the standard pairs of a monomial ideal. Example:

```
./cats_monomialideal2standardpairs
```

followed by

```
{a^2*b,b*b}
```

produces the list

```
{( 1 , (a) ),
( a*b , () ),
( b , () )}
```

5.8 MaximalMinors

The program `cats_maximalminors` computes the set of determinants of all maximal minors of the input matrix. The input matrix must have more columns than rows. Example:

```
./cats_maximalminors
```

. followed by

```
{(0,1)(2,3)(4,5)}
```

produces the list

```
{-4, -2}
```

So the given point configuration is not unimodular.

6 Reliability

The first version of CaTS used the implementation of the simplex algorithm from TiGERS to find facets of Gröbner cones. This was not numerically stable. Permuting the points in large point configurations would sometimes change the number of Gröbner bases computed. To avoid this CaTS was linked to SoPlex [Wunderling] and no such problems have been discovered since. Unfortunately SoPlex relies on floating-point arithmetics so there is no guarantee that the computations are correct. Recently this has been solved by linking CaTS to cddlib [Fukuda] which is capable of handling exact arithmetics using gmp [gmp]. cddlib can also do floating point arithmetics. It is recommended to verify ones computation using cddlib. See Section 3.1 for how to specify which LP solver is being used. Cddlib is slightly faster than SoPlex when not using gmp. With gmp cddlib is many times (>15) times slower. A reason for using SoPlex is that you must be root to install cddlib in a way that works with CaTS without changing the source files. Use SoPlex if cddlib is not installed on your system and you do not have root access to install it.

Another issue regarding reliability is the initial computation of a Gröbner basis for a toric ideal of a point configuration. CaTS uses floating point computations for this, so it might be a good idea to verify the result with another program or simply give CaTS the a reduced Gröbner basis as input.

7 Acknowledgements

Many of the additional features in this version of CaTS were implemented during my visit to the University of Washington in Seattle in Summer 2003. This visit was supported partially by the Milliman Fund at the University of Washington and Rekha Thomas' NSF grant DMS-0100141.

I would like to thank Komei Fukuda for helping with cddlib and Jörg Rambau for helping with TOPCOM and for making it produce output suited for CaTS. Thanks to Edwin O'Shea and Luis David Garcia for suggestions and for testing this software. Rekha Thomas and my thesis adviser Niels Lauritzen have been sharing ideas and comments. In particular I am grateful for Rekha Thomas' many suggestions and her contributions to this manual.

A Commands

The following is a list of command in CaTS with references to examples or explanation. The files listed in "Invoked" are programs run by CaTS when using the specified program/option, see section 2.4.

Program	Description	Invoked	Section
`cats -t2`	Reverse search		4.1
`cats -t3` , `-t1`	Exhaustive search		4.1
`cats -L`	Input as a lattice	4ti2	4.3
`cats -t5`	Fixed radical		4.1
`cats -t7` (, `-A`, `-g`)	A-graded	4ti2 if `-g`	4.2
`cats -t8` (, `-A`, `-g`)	A-graded fixed radical	4ti2 if `-g`	4.2
`cats -w`	Radical by radical using TOPCOM output		4.4
`cats_binomiallist2degree`	Largest degree in a list of binomials		5.6
`cats_fiber`	A fiber of a matrix		5.2
`cats_graver`	The Graver basis	4ti2	5.5
`cats_interactive`	Interactive mode	`M2`	5.1
`cats_isdeltanormal`	Tests if a point configuration is Δ-normal	`normaliz`, `TOPCOM`	5.4
`cats_maximalminors`	The determinants of the maximal minors		5.8
`cats_monomialideal2standardpairs`	The standard pairs of a monomial ideal		5.7
`cats_vectorlist2monomiallist`	Converter		5.3

References

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

[Bayer & Morrison] D. Bayer and I. Morrison,. "Gröbner bases and geometric invariant theory I", *Journal of Symbolic Computation* 1998 6:209–217.

[Bigitta et al.] Bigatti A. M., Scala R. and Robbiano L. , "Computing Toric Ideals", *J. Symbolic Computation,* 1999 27:351–365.

[Bruns & Koch] Bruns W. and Koch R., "NORMALIZ - Computing normalizations of affine semigroups", Available at `ftp://ftp.mathematik.uni-osnabrueck.de/pub/osm/kommalg/software/`, jun 2001

[Fukuda] Fukuda K., "cddlib version 0.93b", Available at `http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html`, oct 2003.

[gmp] "GMP, the GNU MP library version 4.1.2", Available at `http://www.swox.com/gmp/` , oct 2003

[Grayson & Stillman] Grayson, D. R. and Stillman , M. E.,"Macaulay 2, a software system for research in algebraic geometry", Available at `http://www.math.uiuc.edu/Macaulay2/`

[Hemmecke & Hemmecke] Hemmecke R. and Hemmecke R., "4ti2 Version 1.1—Computation of Hilbert bases, Graver bases, toric Gröbner bases, and more", Available at `http://www.4ti2.de`, sep 2003.

[Hoşten & Sturmfels] Hosten S. and Sturmfels B., "GRIN: An implementation of Gröbner Bases for Integer Programming", *Integer Programming and Combinatorial Optimization*, (Springer Lecture Notes in Computer Science, 1995), 920:267–276.

[Hoşten & Thomas] Hosten S. and Thomas R. R., "The associated primes of initial ideals of lattice ideals", *Mathematical Research Letters, 6 (1999)*: 83–97.

[Hoşten & Thomas 2003] Hoşten S. and Thomas, R.R, Gomory Integer Programs, *Math. Programming, Series B* 96 2003 271–292.

[Huber] Huber, B., "TiGERS: Toric Gröbner bases Enumeration by Reverse Search", available at `http://www.math.washington.edu/`$\sim$`thomas`.

[Huber & Thomas] Huber B. and Thomas R. R., "Computing Gröbner Fans of Toric Ideals", *Experimental Mathematics, Vol. 9 (2000), No. 3*: 321–331.

[Jensen] Jensen A., "Computing Gröbner Fans of Toric Ideals", Master's thesis, University of Aarhus 2002, Available at `http://www.soopadoopa.dk/anders/speciale.html`.

[Lenstra et al.] A. K. Lenstra, H. W. Lenstra, L. Lovász, "Factoring Polynomials with Rational Coefficients", *Mathematische Annalen* (Springer Verlag, 1982): 515–534.

[Maclagan & Thomas] Maclagan D. and Thomas R. R., "Combinatorics of the toric Hilbert scheme", *Discrete and Computational Geometry* 27, 2002, no. 2, 249-272.

[Mora & Robbiano] T. Mora and L. Robbiano, "The Gröbner fan of an ideal", *Journal of Symbolic Computation* 6, 1998 183–208.

[Rambau] Rambau J., "TOPCOM: Triangulations of Point Configurations and Oriented Matroids", ZIB report 02-17, Berlin 2002

[Santos] Santos F., Non-connected toric Hilbert schemes, preprint March 2002, 19 pages.

[Sturmfels] Sturmfels B., "Gröbner Bases and Convex Polytopes," American Mathematical Society, University Lecture Series, 1996.

[Sturmfels, Trung & Vogel] Sturmfels B. and Trung N. and Vogel W., Bounds on projective schemes, *Mathematische Annalen* 302 1995 417–432.

[Sturmfels et al.] Sturmfels B. and Weismantel R. and Ziegler G., "Gröbner bases of lattices, corner polyhedra and integer programming", *Beiträge zur Algebra und Geometrie* 36 1995 281–298.

[Wunderling] Wunderling R., "Paralleler und Objekt-orientierter Simplex-Algorithmus", ZIB technical report TR 96-09, Berlin 1996