Normaliz 2.8

Winfried Bruns, Bogdan Ichim and Christof Söger

winfried@math.uos.de
bogdan.ichim@imar.ro
csoeger@math.uos.de

Contents

1 Introduction

1.1 The objectives of Normaliz

The program Normaliz, version 2.8, is a tool for computing the Hilbert bases and enumerative data of rational cones. A rational cone can be given by

(1) a system of generators $\mathscr{G}$ in a lattice $\mathbb{Z}^n$;
(2) constraints: a homogeneous linear system of equations and inequalities;
(3) generators and relations.

The Hilbert basis of a rational pointed cone C in $\mathbb{R}^n$ is defined with respect to a lattice $L \subset \mathbb{Z}^n$: it is the unique minimal system of generators of the monoid $C \cap L$. The standard choice for L is $\mathbb{Z}^n$ itself, but for Normaliz this choice can be modified in two ways:

(1) L can be chosen to be the sublattice of $\mathbb{Z}^n$ generated by $\mathscr{G}$;
(2) L can be chosen to be the lattice of solutions of a homogeneous system of congruences if the cone is specified by equations and inequalities.

In particular, Normaliz solves combined systems of homogeneous diophantine linear equations, inequalities and congruences. (An extension to nonhomogeneous systems is envisaged.) Conversely, Normaliz computes a system of constraints defining the cone and the lattice for which the Hilbert basis has been computed.

Normaliz has special input types for lattice polytopes (represented by their vertices) and monomial ideals (represented by the exponent vectors of their generators). Via the specification of a grading, one can easily apply Normaliz also to rational polytopes.

The enumerative data computed by Normaliz depend on a grading of the monoid under consideration (see Section 3.4): if asked to do so, Normaliz computes the Hilbert series and the Hilbert quasipolynomial of the monoid (or its associated algebra). In polytopal terminology: Normaliz computes Ehrhart series and quasipolynomials of rational polytopes.

The computations can be restricted, for example to the support hyperplanes of the cone or the lattice points of a rational polytope.

For the mathematical background we refer the reader to [2] and [4]. The terminology follows [2]. For algorithms of Normaliz see [5], [3], [6] and [7].

The input syntax of Normaliz has always been kept backward compatible so that input files for older versions can still be used.

1.2 Access from other systems

We provide a SINGULAR library `normaliz.lib` and the package `Normaliz.m2` for MACAULAY2 that make Normaliz accessible from these systems. Thus SINGULAR or MACAULAY2 can be used as a comfortable environment for the work with Normaliz, and, moreover, Normaliz can be applied directly to objects belonging to the classes of toric rings and monomial ideals.

Normaliz has been made accessible from POLYMAKE (thanks to the POLYMAKE team). Normaliz is used by the B. Burton's system REGINA.

1.3 Major changes relative to version 2.7

(1) The main extension of the functionality is the addition of arbitrary $\mathbb{Z}$-gradings. This allows the computation of Hilbert (or Ehrhart series) for such gradings.
(2) The implementation of the algorithms has been improved substantially. In particular, the parallelization is considerably better than in 2.7. See [6] for the details.
(3) The introduction of general $\mathbb{Z}$-gradings forced us to change the wording of the main output file at several places.
(4) Moreover, it was necessary to change the treatment of triangulations in the output.

1.4 Future extensions

(1) Inhomogeneous systems of equations, inequalities and congruences,
(2) a programming interface (using the already existing library),
(3) exploitation of symmetries,
(4) access from further systems,
(5) evaluation of additive functions.

2 Getting started

Download

- the zip file with the Normaliz source, documentation, examples and further platform independent components, and
- zip file made with the executable for your system

from the Normaliz website

`http://www.math.uos.de/normaliz`

and unzip both in the same directory of your choice. In it, a directory `Normaliz2.8` (called Normaliz directory in the following) is created with several subdirectories. (Some versions of the Windows executables may need the installation of a runtime library; see website.)

In the Normaliz directory open jNormaliz by clicking `jNormaliz.jar` in the appropriate way. (We assume that Java is installed on your machine.) In the jNormaliz file dialogue choose one of the input files in the subdirectory `example`, say `small.in`, and press `Run`. In the console window you can watch Normaliz at work. Finally inspect the output window for the results.

The menus and dialogues of jNormaliz are self explanatory, but you can also consult the documentation [1] via the help menu.

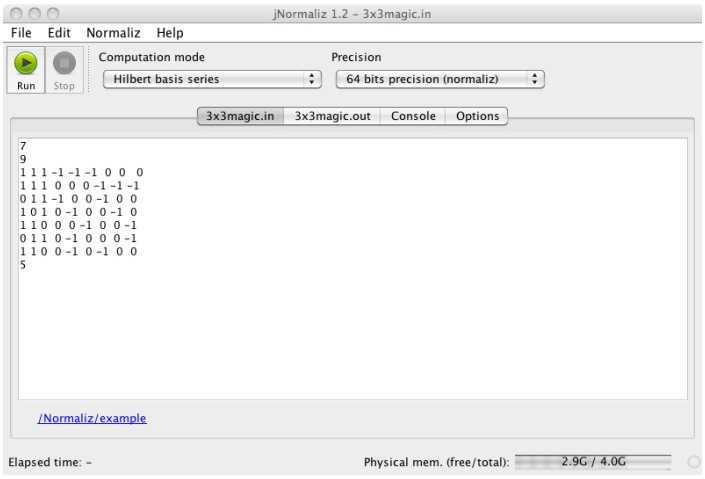

Figure 1: jNormaliz

If the executables prepared cannot be run on your system, then you can compile Normaliz yourself (see Section 11).

Moreover, one can, and often will, run Normaliz from the command line. This is explained in Section 4.

If 64 bit integer precision is not sufficient, then one can switch jNormaliz to infinite precision (or use the option `-B` from the command line). Then Normaliz has no restrictions on the integer precision. See Section 4.5. (The integer precision has nothing to do with the address width (32 bit or 64 bit) of your operating system.)

3 The input file

The input file `<projectname>.in` consists of one or several matrices. Each matrix is built as follows:

(1) The first line contains the number of rows m.
(2) The second contains the number of columns n.
(3) The next m lines of n integers each contain the rows.
(4) The last line contains a single number or word specifying the type of input the matrix presents.

At the moment there are three major types of input matrices, namely *generators*, *constraints*, and *relations*. An additional type is *grading*.

For each input type we specify two lattices: the *ambient lattice* $\mathbb{A}$ to which the input data refer and the *essential lattice* $\mathbb{E} \subset \mathbb{A}$ with respect to which all data are computed.

In this section we assume that Normaliz is run in a computation mode in which the Hilbert basis is actually computed. (See Section 4 for computation modes.)

3.1 Generators

The generator types are 0, 1, 2 and 3. If a matrix of one of these types is in the input file, then it must be the only matrix in the file, unless a grading has been added.

3.1.1 Type 0, `integral_closure`

The rows of an $m \times n$ matrix of type 0 represent m vectors in the ambient lattice $\mathbb{A} = \mathbb{Z}^n$. The essential lattice $\mathbb{E}$ is the smallest direct summand of $\mathbb{Z}^n$ that contains the vectors in the matrix.

The vectors are considered as a system of generators $\mathscr{G}$ of a cone C, and Normaliz computes the Hilbert basis of C with respect to $\mathbb{E}$ (or, equivalently, $\mathbb{Z}^n$).

The nomenclature `integral_closure` is explained by the fact that the Hilbert basis generates the integral closure of the monoid $\mathbb{Z}_+\mathscr{G}$ in $\mathbb{Z}^n$.

A simple example:

```
Input            Hilbert basis
3                1 0
2                0 1
2 0
1 1
0 2
integral_closure
```

In this example, the three input vectors clearly generate the positive orthant $\mathbb{R}^2_+$ in $\mathbb{R}^2$, and the two unit vectors clearly are the Hilbert basis of $\mathbb{R}^2_+ \cap \mathbb{Z}^2$.

Example input files: `rproj2.in`, `small.in`.

3.1.2 Type 1, `normalization`

The matrix is interpreted as in type 0, however $\mathbb{E}$ is chosen as the sublattice of $\mathbb{Z}^n$ generated by $\mathscr{G}$.

The choice of the name `normalization` indicates that Normaliz computes the normalization of the monoid $\mathbb{Z}_+\mathscr{G}$. (The computation of such normalizations was the original goal of Normaliz, hence the name.)

We choose the same input vectors as above, but change the type to `normalization`:

```
Input            Hilbert basis
3                2 0
2                1 1
2 0              0 2
1 1
0 2
normalization
```

The cone has not changed, but the lattice has: $\mathbb{E}$ is now the sublattice of $\mathbb{Z}^2$ of all (z_1, z_2) with

$z_1 + z_2 \equiv 0 \mod 2$.

Example input files: `rafa2416.in`, `A443.in`,

3.1.3 Type 2, `polytope`

The rows of the matrix are interpreted as integral points of a lattice polytope in $\mathbb{R}^n$, which is their convex hull.

The cone C is the cone over the polytope, i.e. the cone with apex 0 in $\mathbb{R}^{n+1}$ generated by the vectors $(x,1)$ where x represents a row of the input matrix. We want to compute the *Ehrhart monoid* $C \cap \mathbb{Z}^{n+1}$.

The lattice $\mathbb{A}$ is $\mathbb{Z}^{n+1}$, and $\mathbb{E}$ is the smallest direct summand of $\mathbb{A}$ containing the generators of C.

Type 2 is only a variant of type 0. One obtains the same results as in type 0 with the extended vectors $(x,1)$ as input.

Note: In previous versions, the text in the output file was adapted to the polytopal situation. In 2.8 `polytope` is only an input variant of type 0.

Example input files: `polytop.in`, `FortuneCookie.in`, `lo6.in`.

3.1.4 Rational polytopes

Normaliz has no special input type for rational polytopes. In order to process them one uses type 0 together with a grading. Suppose the polytope is given by vertices

$$v_i = (r_{i1}, \ldots, r_{in}), \qquad i = 1, \ldots, m,\ r_{ij} \in \mathbb{Q}.$$

Then we write v_i with a common denominator:

$$v_i = \left(\frac{p_{i1}}{q_i}, \ldots, \frac{p_{in}}{q_i} \right), \quad p_{ij}, q_i \in \mathbb{Z},\ q_i > 0.$$

The generator matrix is given by the rows

$$\widetilde{v}_i = (p_{i1}, \ldots, p_{in}, q_i), \quad i = 1, \ldots, m.$$

We must add a grading since Normaliz cannot recognize it without help (unless all the q_i are equal). The grading vector has coordinates $(0, \ldots, 0, 1)$. See 3.4 below for general information on gradings.

Let us look at a concrete example (contained in `rational.in`), the triangle P with vertices

$$(1/2, 1/2),\ (-1/3, -1/3),\ (1/4, -1/2).$$

In order to apply Normaliz to it one uses the following input:

```
3
3
1 1 2
-1 -1 3
1 -2 4
integral_closure
1
3
0 0 1
grading
```

The output will be discussed in 6.1.3.

3.1.5 Type 3, `rees_algebra`

In this type the input vectors are considered as exponent vectors of the generators of a monomial ideal I in the polynomial ring $K[X_1,\dots,X_n]$. Normaliz computes the normalization of the Rees algebra of the ideal I (see [4] for the notion of Rees algebra.) This is a monomial subalgebra of the extended polynomial ring $K[X_1,\dots,X_n,T]$ with an auxiliary variable T. Normaliz computes the exponent vectors in $\mathbb{Z}^{n+1}$ of the system of generators. For an example, see Section 6.

In type 3 one has $\mathbb{A} = \mathbb{E} = \mathbb{Z}^{n+1}$.

Example input file: `rees.in`.

3.1.6 Preparation of the generators

After the coordinate transformation to the lattice $\mathbb{E}$, Normaliz divides each generator by the greatest common divisor of its components. For example, the extreme rays listed will always be such $\mathbb{E}$-primitive vectors (re-transformed to $\mathbb{A}$ where they may not be primitive).

If a grading is present, the generators will be sorted by degree in ascending order. Those of the same degree will remain sorted as in the input file (or the result of a previous computation).

3.2 Constraints

Inequalities, equations, and congruences defining the cone and the lattice are called *constraints*. Matrices representing them are of types 4, 5 and 6. All three types can be present in the input file, and there can be several matrices of each type. The order does not matter. Matrices of the same type will be concatenated. The numbers of columns must of course match: for the ambient lattice $\mathbb{Z}^n$ the matrices of types 4 and 5 must have n columns, and those of type 6 must have $n+1$ columns.

If there is no matrix of type 4, then it is assumed that the user wants to compute the nonnegative solutions of the system represented by the matrices of type 5 and/or 6. The input file is therefore compatible with the types 4 and 5 of previous versions of Normaliz.

3.2.1 Type 4, `hyperplanes`

A row $(\xi_1,\dots,\xi_n)$ of the input matrix of type 4 represents an inequality

$$\xi_1 x_1 + \cdots + \xi_n x_n \geq 0$$

for the vectors $(x_1,\dots,x_n)$ of $\mathbb{R}^n$.

Example:

```
Input            Hilbert basis
2                0 -1
2                1  1
1 0
1 -1
hyperplanes
```

Normaliz has computed the Hilbert basis of the cone defined by the inequalities $x_1 \geq 0$ and $x_1 - x_2 \geq 0$ with respect to the lattice $\mathbb{Z}^2$.

Example input file: `Condorcet.in`

3.2.2 Polytopes by inequalities

Normaliz has no special input type for polytopes defined by inequalities since they can easily be specified via type 4. Suppose the polytope is given by inequalities

$$\alpha_{i_1} x_1 + \cdots + \alpha_{in} x_n \geq \beta_i, \quad i = 1,\dots,m,\ \alpha_{ij},\beta_i \in \mathbb{Z}.$$

Then we homogenize the inequalities in the form

$$\alpha_{i_1} x_1 + \cdots + \alpha_{in} x_n - \beta_i x_{n+1} \geq 0,$$

and use type 4 for them in connection with the grading vector $(0,\dots,0,1)$.

The file `poly_ineq.in` contains

```
3
3
2  7 3
-8 2 3
1 -1 0
hyperplanes
1
3
0 0 1
grading
```

It reproduces the triangle that we have discussed in 3.1.4.

3.2.3 Type 5, `equations`

A row $(\xi_1,\dots,\xi_n)$ of the input matrix of type 5 represents an equation

$$\xi_1 x_1 + \cdots + \xi_n x_n = 0$$

for the vectors $(x_1,\dots,x_n)$ of $\mathbb{R}^n$.

Example:

```
Input            Hilbert basis
1                2 0 1
3                0 2 1
1 1 -2           1 1 1
equations
```

If the input file contains no further matrices, Normaliz has computed the Hilbert basis of the subcone of $\mathbb{R}^3_+$ defined by the equation $x_1 + x_1 - 2x_3 = 0$.

Example input files: `4x4.in`, `5x5.in`.

3.2.4 Type 6, `congruences`

We consider the rows of a matrix of type 6 to have length $n+1$. Each row $(\xi_1,\dots,\xi_n,c)$ represents a congruence

$$\xi_1 z_1 + \cdots + \xi_n z_n \equiv 0 \mod c$$

for the elements $(z_1,\dots,z_n) \in \mathbb{Z}^n$.

Example:

```
Input            Hilbert basis
1                2 0
3                1 1
1 1 2            0 2
congruences
```

If no other matrix is in the input file, then Normaliz computes the Hilbert basis of the positive orthant intersected with the lattice of all integral vectors (z_1,z_2) such that $z_1 + z_2 \equiv 0 \mod 2$ and the result is the same as in 3.1.2 above.

Example input file: `3x3magiceven.in`.

3.2.5 The constraints combined

Let L be the sublattice of $\mathbb{Z}^n$ that consists of the solutions of the system of congruences defined by the input matrix of type 6 ($L = \mathbb{Z}^n$ if there is no matrix of type 6). Moreover let A be the matrix of type 4 and B be the matrix of type 5. Then the cone C is given by

$$C = \{x \in \mathbb{R}^n : Ax \geq 0,\ Bx = 0\}.$$

and the Hilbert basis of $C \cap L$ is computed.

The ambient lattice $\mathbb{A}$ is $\mathbb{Z}^n$, and the essential lattice is $\mathbb{E} = L \cap \mathbb{R}C$.

If there is no matrix of type 5, then the system of equations is empty, satisfied by all vectors of $\mathbb{R}^n$.

Note that there is always a matrix of type 4, either explicitly in the input, or implicitly, namely the $n \times n$ unit matrix, if there is no matrix of type 4 in the input file (but one of type 5 or 6).

See Section 6.2.3 for an example combining types 5 and 6.

Example input file: `3x3magiceven.in`.

3.3 Relations

Relations are another type of constraints. They do not select a sublattice of $\mathbb{Z}^n$ or a subcone of $\mathbb{R}^n$, but define a monoid as a quotient of $\mathbb{Z}^n_+$ modulo a system of congruences (in the semigroup sense!).

Let U be a subgroup of $\mathbb{Z}^n$. Then the natural image M of $\mathbb{Z}^n_+ \subset \mathbb{Z}^n$ in the abelian group $G = \mathbb{Z}^n/U$ is a submonoid of G. In general, G is not torsionfree, and therefore M may not be an affine monoid. However the image N of M in the lattice $L = G/\text{torsion}$ is an affine monoid. Normaliz chooses an embedding $L \hookrightarrow \mathbb{Z}^r$, $r = n - \operatorname{rank} U$, such that N becomes a submonoid of $\mathbb{Z}^r_+$. In general there is no canonical choice for such an embedding, but one can always find one, provided N has no invertible element except 0. The ambient lattice is then $\mathbb{A} = \mathbb{Z}^r$, and the essential lattice is L, realized as a sublattice of $\mathbb{A}$.

The typical starting point is an ideal $J \subset K[X_1, \ldots, X_n]$ generated by binomials

$$X_1^{a_1} \cdots X_n^{a_n} - X_1^{b_1} \cdots X_n^{b_n}.$$

The image of $K[X_1, \ldots, X_n]$ in the residue class ring of the Laurent polynomial ring $S = K[X_1^{\pm 1}, \ldots, X_n^{\pm 1}]$ modulo the ideal JS is exactly the monoid algebra $K[M]$ of the monoid M above if we let U be the subgroup of $\mathbb{Z}^n$ generated by the differences

$$(a_1, \ldots, a_n) - (b_1, \ldots, b_n).$$

Ideals of type JS are called lattice ideals if they are prime. Since Normaliz automatically passes to $G/\text{torsion}$, it replaces JS by the smallest lattice ideal containing it.

3.3.1 Type 10, `lattice_ideal`

The rows of the input matrix of type 10 are interpreted as generators of the subgroup U, and Normaliz performs the computation as just explained.

As an example we consider the binomials $X_1X_3 - X_2^2$, $X_1X_4 - X_2X_3$:

```
Input            Hilbert basis
2                3 0
4                2 1
1 -2  1 0        1 2
```

```
1 -1 -1 1       0 3
lattice_ideal
```

In this example $\mathbb{Z}^4/U$ is torsionfree, but we can replace each of the vectors in the input matrix by a nonzero integral multiple without changing the result.

Type 10 cannot be combined with any other input type (except `grading`)—such a combination would not make sense.

Example input file: `lattice_ideal.in`.

3.4 Gradings

Normaliz can compute the Hilbert series and the Hilbert (quasi)polynomial of a graded monoid. A *grading* of a monoid M is simply a homomorphism $\deg : M \to \mathbb{Z}^g$ where $\mathbb{Z}^g$ contains the degrees. The *Hilbert series* of M with respect to the grading is the formal Laurent series

$$H(T) = \sum_{d\in\mathbb{Z}^g} \#\{x \in M : \deg x = d\} T_1^{d_1} \cdots T_g^{d_g},$$

provided all sets $\{x \in M : \deg x = d\}$ are finite.

At the moment, Normaliz can only handle the case $g = 1$. A $\mathbb{Z}$-valued grading can be specified in two ways:

(1) *explicitly* by including a grading in the input, or
(2) *implicitly*. In this case Normaliz checks whether the extreme integral generators of the monoid lie in an (affine) hyperplane A. If so, then the (unique) primitive $\mathbb{Z}$-linear form λ that affords an equation $\lambda(x) = b$ for A is used as the grading.

The basic fact about $H(t)$ in the $\mathbb{Z}$-graded case is that it is the Laurent expansion of a rational function at the origin:

$$H(t) = \frac{R(t)}{(1-t^e)^r}, \qquad R(t) \in \mathbb{Z}[t], \tag{1}$$

where r is the rank of M and e is the least common multiple of the degrees of the extreme integral generators of the cone.

Usually one can find denominators for $H(t)$ of much lower degree than in equation (1), and Normaliz tries to give a more economical presentations of $H(t)$ as a quotient of two polynomials. One should note that it is not clear what the most natural presentation of $H(t)$ is in general (when $e > 1$). We discuss this problem in [6, Section 4] and in 6.1.3. The examples in Section 6, especially 6.1.3 and 6.2.3, may serve as an illustration.

A rational cone C and a grading together define the rational polytope $P = C \cap A_1$ where $A_1 = \{x : \deg x = 1\}$. In this sense the Hilbert series is nothing but the Ehrhart series of P.

Note: In previous versions we used *height* as a synonym for *degree*.

A grading is explicitly specified by an $1 \times n$ matrix for cones embedded in $\mathbb{R}^n$, and its type is fixed by the attribute `grading`, for example

```
1
2
3 2
grading
```

We have not assigned a numerical type to matrices (effectively, vectors) specifying the grading.

Normaliz checks whether all generators of the monoid have positive degree.

Before Normaliz can apply the degree, it must be restricted to the effective lattice $\mathbb{E}$. Even if the entries of the grading vector are coprime, it often happens that all degrees of vectors in $\mathbb{E}$ are divisible by a greatest common divisor $d > 1$. Then d is extracted from the degrees, and it will appear as `denominator` in the output file.

Special rules apply to some input types that we explain in the following.

3.4.1 `polytope`

Cones defined by lattice polytopes always have an implicit grading in which the lattice points in the polytope have degree 1 (roughly speaking). Therefore it is not possible to use an explicit grading together with this input type.

If it should be necessary to apply a different grading, then one converts the input of type `polytope` to `integral_closure` by appending `1` to each row of the input matrix and adds the grading to be used.

3.4.2 `rees_algebra`

Suppose that the rows of the input matrix specify vectors of length n. Then these are embedded into $\mathbb{R}^{n+1}$, and therefore the grading must have $n+1$ components. Example:

```
3
3
0 1 2
2 0 2
1 1 1
rees_algebra
1
4
1 1 1 -1
grading
```

Note that the Rees algebra has an implicit grading if and only if all the monomials have the same total degree, say g. Then the grading vector chosen automatically is $(1,\dots,1,-(g-1))$.

3.4.3 `lattice_ideal`

In this case the unit vectors correspond to generators of the monoid. Therefore the degrees assigned to them must be positive. Moreover, the vectors in the input represent binomial relations, and these must be homogeneous. In other words, both monomials in a binomial must have the same degree. This amounts to the condition that the input vectors have degree 0. Normaliz checks this condition. Example:

```
1
4
1 1 -1 -1
lattice_ideal
1
4
1 2 1 2
grading
```

3.5 Pointedness

For Hilbert basis computations and triangulations Normaliz requires the cone to be pointed ($x, -x \in C \implies x = 0$). Whenever the condition of pointedness is violated at a step where it is crucial, Normaliz will stop computations.

Pointedness is checked by testing whether the dual cone of C is full dimensional, and if not, then the constructor of the cone complains as follows:

```
Full Cone error: Matrix with rank = number of columns needed in
the constructor of the object Full_Cone. Probable reason: Cone
not full dimensional(<=> dual cone not pointed)!
```

3.6 The zero cone

The zero cone with an empty Hilbert basis is a legitimate object for Normaliz. Nevertheless a warning message is issued if the zero cone is encountered.

4 Running Normaliz

The simplest way to call Normaliz from the command line is

```
normaliz <projectname>
```

for example

```
normaliz rafa2416
```

The project name is `rafa2416`. Normaliz reads the input file `rafa2416.in` (hopefully existing), computes everything it can compute, and writes the output to `rafa2416.out`.

The full syntax for calling Normaliz from the command line is

```
normaliz [-stvnhpN1dcBefaT] [-x=<T>] [<projectname>]
```

where the options and `<projectname>` are optional. (We assume that the executable `normaliz` or `normaliz.exe` is in the search path. Otherwise you have to prefix it with a suitable relative or absolute path.) If no `<projectname>` is given, the program will ask you for it or display a help screen.

The option `-x` differs from the other ones: `<T>` represents a positive number assigned to `-x`; see Section 4.3.

The help screen can also be displayed by `normaliz -?`.

Normaliz will look for `<projectname>.in` as input file.

For example, if you input the command

```
normaliz -c -p -a -T rafa2416     or     normaliz -cpaT rafa2416
```

then the program will take the file `rafa2416.in` as input, control data will be displayed on your terminal, the support hyperplanes, the triangulation, the multiplicity, the Hilbert series and the Hilbert (quasi)polynomial will be computed and all the possible output files will be produced.

If you inadvertently typed `rafa2416.in` as the project name, then Normaliz will first look for `rafa2416.in.in` as the input file. If this file doesn't exist, `rafa2416.in` will be loaded.

In the following we explain the various options of Normaliz. The full text names given appear in the help screen as well as in the menus of jNormaliz which allows you to choose options interactively.

The default computation mode is `-h`. All options that can be activated are switched off by default.

Options are evaluated from left to right. Therefore the last of mutually exclusive options is used.

Note: In 2.7 and earlier versions the default computation mode was `-n`. The change to `-h` reflects that the extra time for the computation of the Hilbert series can be neglected now (actually, already in 2.7). The choice of `-h` ensures that Normaliz computes all the information accessible to it.

4.1 Computation modes

4.1.1 Support hyperplanes and extreme rays

The least that Normaliz can do is

`-s support hyperplanes`: only the support hyperplanes of the cone under consideration and the extreme rays are computed.

4.1.2 Modes with full triangulation

The following modes form an ascending chain. All of them compute a full triangulation:

`-t triangulation`: in addition to the support hyperplanes the triangulation is computed, but not evaluated (it can be written to the output). Why `-t` is nevertheless useful will be explained in Section 9.

`-v volume`: includes `-t`. Now Normaliz evaluates the triangulation and computes the multiplicity (or normalized volume) if a grading is available.

`-n Hilbert basis volume`: includes `-v` and Normaliz computes the Hilbert basis.

`-h Hilbert basis series`: includes `-n`. If a grading is given, Normaliz computes the Hilbert series and the Hilbert (quasi)polynomial. This computation mode yields the maximum information Normaliz can produce.

If only the Hilbert series is to be computed, then one uses

`-p Hilbert series`.This mode is faster than `-h`. It also computes the degree 1 elements of the Hilbert basis.

Note: some of the options had slightly different names in 2.7.

4.1.3 Modes with partial triangulation

For the computation of Hilbert bases and/or degree 1 elements it is enough to use a partial triangulation (see [3]). The following two computation modes take advantage of this fact:

`-N Hilbert basis`: includes `-s` and computes the Hilbert basis.

`-1 degree 1 elements`: includes `-s`, only degree 1 elements are computed. (This is the fastest mode for computing the lattice points in a polytope.)

4.1.4 The dual algorithm

If a cone is defined by constraints, it is often (but not always) faster to use a Hilbert basis algorithm originally due to Pottier [9] that we call the *dual* algorithm, in contrast to the *primal* (triangulation based) algorithm of Normaliz. (See [5] for our version of the dual algorithm.) The dual algorithm is invoked by

`-d dual`

The dual algorithm computes only Hilbert bases (and degree 1 elements if a grading is given).

The dual algorithm can be used with all input types. See Section 8 for a comparison of performance on various examples.

4.2 Control of output files

In the default setting Normaliz writes only the output file <projectname>.out. The amount of output files can be increased as follows:

- -f Normaliz writes the additional output files with suffixes gen, cst, and inv, provided the data of these files have been computed.
- -a includes -f, Normaliz writes all available output files except the triangulation data.
- -T Normaliz writes the triangulation data should these have been computed.

In order to see all available output files one uses -aT.

The triangulation data are treated separately since triangulations can become very large and may exhaust memory if they must be stored for output.

For the list of potential output files and their interpretation see Section 7.

4.3 Control of execution

The options that control the execution are:

- -c activates the verbose ("control") behavior of Normaliz in which Normaliz writes additional information about its current activities to the standard output.
- -e activates the overflow error check of Normaliz. Ignored if used with -B.
- -B Switches Normaliz to infinite precision.

-x=<T> Here <T> stands for a positive integer limiting the number of threads that Normaliz is allowed access on your system. The default value is set by the operating system. If you want to run Normaliz in a strictly serial mode, choose -x=1.

The number of threads can also be controlled by the environment variable OMP_NUM_THREADS. See Section 8 for further discussion.

4.4 Obsolete options

The options -i and -m of version 2.2 have become obsolete. They will be ignored if present.

The options -SVHP of versions 2.5 and 2.7 are now synonymous with -svhp and can still be used.

4.5 Numerical limitations

Even in low dimensions, the range of 64 bit integers may not be sufficient for the computations of Normaliz. Therefore normaliz can be switched to infinite precision by the option -B.

Computations with -B typically run about 5 times slower than those without it. In examples that look critical, it may be useful first to try normaliz without -B, but with the error check option activated. This costs time, too, but hardly more than 50% extra.

The user should run the example `critical64.in` in the subdirectory `examples` with `normaliz -e` in order to see the failure of 64 bit arithmetic. (Running it with `-B` takes a while and requires much memory.)

5 The output file

The data you will find in the output file depend on the input type and on the computation mode. The output file starts with an "abstract" that collects various numerical and qualitative data, for example the number of elements in the Hilbert basis. The lists of vectors, equations etc. follow the abstract.

The output file `<projectname>.out` will contain the following data as far as computed:

- only for type 10: the original system of generators (see below);
- the Hilbert basis H computed;
- the extreme rays of the cone C generated by H;
- the rank of the lattice $\mathbb{E}$;
- the index of the lattice generated by the original input vectors in $\mathbb{E}$;
- the support hyperplanes of C;
- a system of equations defining the vector space generated by C;
- a system of congruences defining $\mathbb{E}$ as a sublattice of $\mathbb{A}$ (together with the equations);
- the number of simplicial cones in the triangulation and the sum of the absolute values of their determinants.

In the presence of a grading the following extra data may be printed:

- the linear form defining the degree;
- the degree 1 elements of the Hilbert basis;
- the multiplicity;
- the Hilbert series and the coefficients of the Hilbert (quasi)polynomial.

The degrees of the extreme rays are listed in the abstract. If the whole Hilbert basis is of degree 1, this fact is indicated. Moreover, Normaliz tells you whether the original system of generators contains the Hilbert basis by indicating whether the original monoid is integrally closed.

Please note:

(1) The equations and support hyperplanes *together* define the cone C. While support hyperplanes will always be present (except for the zero cone), equations will only be printed if necessary, namely when $\dim C < \operatorname{rank} \mathbb{A}$.

Similarly, congruences will only be printed if the lattice $\mathbb{E}$ is not given by $\mathbb{R}C \cap \mathbb{A}$. This can only happen with input matrices of type 1 or 6. The lattice $\mathbb{E}$ is defined simultaneously by the equations and the congruences.

Even if the cone and the lattice are defined by constraints, the inequalities, equations and congruences will in general not be reproduced, but replaced by an equivalent system.

(2) The extreme rays are given by the first points in $\mathbb{E}$ on them (the extreme integral generators with respect to $\mathbb{E}$).
(3) In order to lift the grading from $\mathbb{E}$ to $\mathbb{A}$ it may be necessary to replace it by a multiple (in order to avoid fractions as coefficients). The necessary factor appears as "denominator". The Hilbert series and the Hilbert (quasi)polynomial do always refer to the degree in $\mathbb{E}$.
(4) Input matrices of types 0, 1, 2 or 3 contain an explicit system of generators. For the other types $\neq 10$ the extreme rays computed by Normaliz take their place. For type 10 Normaliz first computes the monoid M generated by the residue classes of the canonical basis of $\mathbb{Z}^n$ (compare Section 3.3), and they are considered the original system of generators.

In type $= 3$ (`rees_algebra`), the data in the output file refer to the integral closure $\overline{\mathscr{R}}$ of the Rees algebra. In addition to what has been mentioned already, the following data are computed:

- the generators of the integral closure of the ideal;
- if the ideal is primary to the irrelevant maximal ideal, the multiplicity of the ideal (not to be confused with the multiplicity of the monoid).

6 Examples

6.1 Generators

6.1.1 Type 0, `integral_closure`

The file `rproj2.in` contains the following (here typeset in 2 columns):

```
16
 7
1 0 0 0 0 0 0          1 0 1 0 1 0 1
0 1 0 0 0 0 0          1 0 0 1 0 1 1
0 0 1 0 0 0 0          1 0 0 0 1 1 1
0 0 0 1 0 0 0          0 1 1 0 0 1 1
0 0 0 0 1 0 0          0 1 0 1 1 0 1
0 0 0 0 0 1 0          0 1 0 0 1 1 1
1 1 1 0 0 0 1          0 0 1 1 1 0 1
1 1 0 1 0 0 1          0 0 1 1 0 1 1
                       0
```

This means that we wish to compute the the cone C generated by the 16 vectors

$$(1,0,0,0,0,0,0)\,, \quad (0,1,0,0,0,0,0)\,, \quad \ldots\,, \quad (0,0,1,1,0,1,1)$$

in $\mathbb{R}^7$ with respect to the full lattice $\mathbb{Z}^7$, as indicated by the final digit that specifies the type. (Actually, the vectors generate the full lattice so that a replacement of type 0 by type 1 would not change anything.)

Running `normaliz` with no option (or option `-h`, `Hilbert basis series`) produces the file `rproj2.out` which has the following content (partially typeset in 2 or 3 columns):

```
17 Hilbert basis elements                    multiplicity = 72
16 Hilbert basis elements of degree 1
16 extreme rays                              Hilbert series:
24 support hyperplanes                       1 9 31 25 6
                                             denominator with 7 factors:
rank = 7 (maximal)                           1: 7
index = 1
original monoid is not integrally closed     Hilbert polynomial:
                                             60 194 284 245 130 41 6
size of triangulation = 67                   with common denominator = 60
resulting sum of |det|s = 72

grading:
1 1 1 1 1 1 -2

degrees of extreme rays:
1: 16

Hilbert basis elements are not of degree 1

******************************************************************************

17 Hilbert basis elements:                 24 support hyperplanes:
 0 0 0 0 0 1 0                               0  0  0  1  0  0  0
 0 0 0 0 1 0 0                               0  0  0  0  1  0  0
 0 0 0 1 0 0 0                               0  0  0  0  0  1  0
 0 0 1 0 0 0 0                               0  0  0  0  0  0  1
 0 0 1 1 0 1 1                               0  0  1  0  0  0  0
 0 0 1 1 1 0 1                               0  1  0  0  0  0  0
 0 1 0 0 0 0 0                               0  1  1  0  0  1 -1
 0 1 0 0 1 1 1                               0  1  0  0  1  1 -1
 0 1 0 1 1 0 1                               0  1  0  1  1  0 -1
 0 1 1 0 0 1 1                               0  0  1  1  0  1 -1
 1 0 0 0 0 0 0                               0  1  1  1  1  1 -2
 1 0 0 0 1 1 1                               0  0  1  1  1  0 -1
 1 0 0 1 0 1 1                               1  0  0  0  0  0  0
 1 0 1 0 1 0 1                               1  1  1  1  1  1 -3
 1 1 0 1 0 0 1                               1  0  0  0  1  1 -1
 1 1 1 0 0 0 1                               1  0  0  1  0  1 -1
 1 1 1 1 1 1 2                               1  0  1  0  1  0 -1
                                             1  0  1  1  1  1 -2
16 extreme rays:                             1  1  1  0  0  0 -1
 1 0 0 0 0 0 0                               1  1  1  1  0  1 -2
 0 1 0 0 0 0 0                               1  1  0  1  0  0 -1
```

```
0 0 1 0 0 0 0                              1  1  1  0  1  1 -2
0 0 0 1 0 0 0                              1  1  1  1  1  0 -2
0 0 0 0 1 0 0                              1  1  0  1  1  1 -2
0 0 0 0 0 1 0
1 1 1 0 0 0 1              16 degree 1 Hilbert basis elements:
1 1 0 1 0 0 1               0 0 0 0 0 1 0
1 0 1 0 1 0 1               0 0 0 0 1 0 0
1 0 0 1 0 1 1               0 0 0 1 0 0 0       0 1 1 0 0 1 1
1 0 0 0 1 1 1               0 0 1 0 0 0 0       1 0 0 0 0 0 0
0 1 1 0 0 1 1               0 0 1 1 0 1 1       1 0 0 0 1 1 1
0 1 0 1 1 0 1               0 0 1 1 1 0 1       1 0 0 1 0 1 1
0 1 0 0 1 1 1               0 1 0 0 0 0 0       1 0 1 0 1 0 1
0 0 1 1 1 0 1               0 1 0 0 1 1 1       1 1 0 1 0 0 1
0 0 1 1 0 1 1               0 1 0 1 1 0 1       1 1 1 0 0 0 1
```

From this, we see that there are 17 elements in the Hilbert basis, of which 16 are of degree 1, and 16 extreme rays, that the sublattice generated by the input vectors has index 1 in $\mathbb{Z}^7$, and that the corresponding support hyperplanes are given by the linear forms $(0,0,0,1,0,0,0)$, $(0,0,0,0,1,0,0)$, $\ldots$, $(1,1,0,1,1,1,-2)$.

We are also given the information that there is a grading (defined implicitly) and what it is. The multiplicity with respect to this grading is 72. By definition, the multiplicity is the $\mathbb{E}$-normalized volume of the polytope obtained by intersecting the cone with the hyperplane at degree 1.

The degrees of the extreme rays are given in multiset notation:

```
1:  16
```

indicates that 16 extreme rays have degree 1. (The input file contains no explicit grading. The implicitly defined grading requires that all extreme rays have the same degree, but it need not be 1 as in this case.)

Since there is a grading, the degree 1 elements of the Hilbert basis, the Hilbert series and Hilbert polynomial of the monoid generated by the Hilbert basis are also computed. The Hilbert series is given as a rational function. Its numerator polynomial is

$$1+9t+31t^2+25t^3+6t^4$$

as we can see from the vector below the heading `Hilbert series`. The denominator is given in multiset notation: `1:  7` specifies the denominator

$$(1-t^1)^7.$$

More general cases will be discussed in 6.1.3 and 6.2.3 below.

The Hilbert polynomial is given by

$$P(k)=\frac{60}{60}+\frac{194}{60}k+\frac{284}{60}k^2+\frac{245}{60}k^3+\frac{130}{60}k^4+\frac{41}{60}k^5+\frac{6}{60}k^6\,.$$

The Hilbert polynomial gives the number of elements of degree k, starting from degree 0, as is always the case for normal monoids. Note that the multiplicity m can also be read from the leading coefficient c of the Hilbert polynomial:

$$c = \frac{m}{(r-1)!}, \qquad r = \text{rank}, \tag{2}$$

in our case

$$c = \frac{1}{10} = \frac{72}{720}.$$

The lines

```
size of triangulation = 67
resulting sum of |det|s = 72
```

give some information about Normaliz' (not so hard) work: It produced a triangulation of 67 simplicial cones, and the sum of the absolute values of the determinants of their generator matrices is 72. It is not surprising that this number equals the multiplicity. This is always the case if only degree 1 vectors appear in the generator matrix.

We omit an example of type 1 since it does not add anything new.

6.1.2 Type 2, `polytope`

The file `polytop.in`:

```
4
3
0 0 0
2 0 0
0 3 0
0 0 5
polytope
```

The Ehrhart monoid of the integral polytope with the 4 vertices

$$(0,0,0)\,, \quad (2,0,0)\,, \quad (0,3,0) \quad \text{and} \quad (0,0,5)$$

in $\mathbb{R}^3$ is to be computed. (Note the last line, indicating the polytope type 2.)

Running `normaliz` without an option (or option `-h`) produces the file `polytop.out`:

```
19 Hilbert basis elements
18 Hilbert basis elements of degree 1
4 extreme rays
4 support hyperplanes

rank = 4 (maximal)
index = 30
original monoid is not integrally closed

multiplicity = 30

Hilbert series:
1 14 15
denominator with 4 factors:
1: 4

Hilbert polynomial:
1 4 8 5
```

```
size of triangulation = 1                          with common denominator = 1
resulting sum of |det|s = 30

grading:
0 0 0 1

degrees of extreme rays:
1: 4

Hilbert basis elements are not of degree 1

**********************************************************************

19 Hilbert basis elements:     18 Hilbert basis elements of degree 1:
 0 0 5 1                        0 0 5 1
 0 3 0 1                        0 3 0 1
 2 0 0 1                        2 0 0 1
 0 0 0 1                        0 0 0 1
 0 1 3 1                        0 1 3 1
 0 1 2 1                        0 1 2 1
 0 1 1 1                        0 1 1 1
 0 1 0 1                        0 1 0 1
 0 0 4 1                        0 0 4 1
 0 0 3 1                        0 0 3 1
 0 0 2 1                        0 0 2 1
 0 0 1 1                        0 0 1 1
 0 2 0 1                        0 2 0 1
 0 2 1 1                        0 2 1 1
 1 0 0 1                        1 0 0 1
 1 0 1 1                        1 0 1 1
 1 0 2 1                        1 0 2 1
 1 1 0 1                        1 1 0 1
 1 2 4 2

4 extreme rays:                4 support hyperplanes:
 0 0 0 1                        -15 -10  -6  30
 2 0 0 1                          1   0   0   0
 0 3 0 1                          0   1   0   0
 0 0 5 1                          0   0   1   0
```

For the polytopal interpretation one must observe that all data are in homogenized coordinates for which we Normaliz has appended 1 to the input vectors that (in this case) are the vertices of the polytope. In the cone produced the lattice points of the polytope are on degree 1. Therefore the 18 Hilbert basis elements of degree 1 represent the lattice points of the polytope, starting from $(0,0,5)$ and ending with $(1,1,0)$. The extreme rays represent the 4 vertices.

From the fact that there are 19 Hilbert basis elements, but only 18 of degree 1, we see that the lattice points in the polytope do not yield the Hilbert basis of the Ehrhart monoid (or the cone over the polytope).

That there is only one simplicial cone in the triangulation is not surprising since our polytope is a simplex.

The support hyperplanes give us a description of the polytope by inequalities: it is the solution of the system of the 4 inequalities

$$x_3 \geq 0\,, \quad x_2 \geq 0\,, \quad x_1 \geq 0 \quad \text{and} \quad 15x_1 + 10x_2 + 6x_3 \leq 30\,,$$

The dimension of the polytope is 3 since the cone over it has dimension 4. The polytope has $\mathbb{Z}^3$-normalized volume 30 as indicated by the multiplicity.

The Ehrhart series (we use the more general term Hilbert series) is

$$\frac{1 + 14t + 15t^2}{(1-t)^4}$$

and its Ehrhart polynomial (again we use a more general term in the output file) of the polytope is

$$p(k) = 1 + 4k + 8k^2 + 5k^3\,.$$

6.1.3 A rational polytope

We want to investigate the Ehrhart series of the triangle P with vertices

$$(1/2, 1/2),\ (-1/3, -1/3),\ (1/4, -1/2).$$

The input file is `rational.in`. Running Normaliz yields the following output:

```
8 Hilbert basis elements
1 Hilbert basis elements of degree 1
3 extreme rays
3 support hyperplanes

rank = 3 (maximal)
index = 15
original monoid is not integrally closed

size of triangulation   = 1
resulting sum of |det|s = 15

grading:
0 0 1

degrees of extreme rays:
```

```
Hilbert series with cyclotomic denominator:
-1 -1 -1 -3 -4 -3 -2
cyclotomic denominator:
1: 3  2: 2  3: 1  4: 1

Hilbert quasi-polynomial of period 12:
  0:   48  28  15
  1:   11  22  15
  2:  -20  28  15
  3:   39  22  15
  4:   32  28  15
  5:   -5  22  15
  6:   12  28  15
  7:   23  22  15
  8:   16  28  15
  9:   27  22  15
```

```
2: 1  3: 1  4: 1                          10:   -4  28  15
                                          11:    7  22  15
multiplicity = 5/8                       with common denominator = 48

Hilbert series:
1 0 0 3 2 -1 2 2 1 1 1 1 2
denominator with 3 factors:
1: 1  2: 1  12: 1


**********************************************************************

8 Hilbert basis elements:         3 extreme rays:
  1 -2  4                           1  1  2
 -1 -1  3                          -1 -1  3
  1  1  2                           1 -2  4
  0  0  1
  0 -1  3                         3 support hyperplanes:
  1  0  3                           2  7  3
  1 -1  4                          -8  2  3
  0 -2  5                           1 -1  0

                                  1 Hilbert basis elements of degree 1:
                                   0 0 1
```

The 3 extreme rays have reproduced the vertices (don't forget that the last coordinate can be interpreted as a denominator), and the 3 support hyperplanes represent the 3 inequalities that define the polytope as an intersection of affine halfspaces like in 6.1.2. The Hilbert basis element of degree 1 shows that there is a single lattice point in the polytope, namely the origin $(0,0)$. Except that P has non-integral vertices now, these data are completely analogous to those of the lattice polytope in 6.1.2.

The multiplicity is a rational number. Since in dimension 2 the normalized area (of full-dimensional polytopes) is twice the Euclidean area, we see that P has Euclidean area $5/16$.

The Hilbert (or Ehrhart) function counts the lattice points in kP, $k \in \mathbb{Z}_+$. The corresponding generating function is a rational function $H(t)$ with numerator

$$1+3t^3+2t^4-t^5+2t^6+2t^7+t^8+t^9+t^{10}+t^{11}+2t^{12}$$

and denominator

$$(1-t)(1-t^2)(1-t^{12}).$$

As a rational function, $H(t)$ has degree -3. This implies that $3P$ is the smallest integral multiple of P that contains a lattice point in its interior.

Normaliz gives also a representation as a quotient of coprime polynomials with the denominator factored into cyclotomic polynomials. The multiset notation lists the orders of the

cyclotomic polynomials and their multiplicities. In this case we have

$$H(t) = -\frac{1+t+t^2+t^3+4t^4+3t^5+2t^6}{\zeta_1^3\zeta_2^2\zeta_3\zeta_4}$$

where ζ_i is the i-th cyclotomic polynomial ($\zeta_1 = t-1$, $\zeta_2 = t+1$, $\zeta_3 = t^2+t+1$, $\zeta_4 = t^2+1$).

Normaliz transforms the representation with cyclotomic denominator into one with denominator of type $(1-t^{e_1})\cdots(1-t^{e_r})$, $r =$ rank, by choosing e_r as the least common multiple of all the orders of the cyclotomic polynomials appearing, e_{r-1} as the lcm of those orders that have multiplicity ≥ 2 etc.

There are other ways to form a suitable denominator with 3 factors $1-t^e$, for example $g(t) = (1-t^2)(1-t^3)(1-t^4) = -\zeta_1^3\zeta_2^2\zeta_3\zeta_4$. Of course, $g(t)$ is the optimal choice in this case. However, P is a simplex, and in general such optimal choice may not exist. We will explain the reason for our standardization below.

Let $p(k)$ be the number of lattice points in kP. Then $p(k)$ is a quasipolynomial:

$$p(k) = p_0(k) + p_1(k)k + \cdots + p_{r-1}(k)k^{r-1},$$

where the coefficients depend on k, but only to the extent that they are periodic of a certain period $\pi \in \mathbb{N}$. In our case $\pi = 12$ (the lcm of the orders of the cyclotomic polynomials).

The table giving the quasipolynomial is to be read as follows: The first column denotes the residue class j modulo the period and the corresponding line lists the coefficients $p_i(j)$ in ascending order of i, multiplied by the common denominator. So

$$p(k) = 1 + \frac{7}{12}k + \frac{5}{16}k^2, \qquad k \equiv 0 \quad (12),$$

etc. The leading coefficient is the same for all residue classes and equals the Euclidean volume as in equation (2).

Our choice of denominator for the Hilbert series is motivated by the following fact: e_i is the common period of the coefficients $p_{r-i}, \ldots, p_{r-1}$. The user should prove this fact or at least verify it by several examples.

6.1.4 Type 3, `rees_algebra`

Next, let us discuss the example `rees.in`:

```
10
6                       0 1 1 0 0 1
1 1 1 0 0 0             0 1 0 1 1 0
1 1 0 1 0 0             0 1 0 0 1 1
1 0 1 0 1 0             0 0 1 1 1 0
1 0 0 1 0 1             0 0 1 1 0 1
1 0 0 0 1 1             rees_algebra
```

Comparing with the data in `rproj2.in` shows that `rees` is the origin of `rproj2`.

Here we want to compute the integral closure of the Rees algebra of the ideal generated by the monomials corresponding to the above 10 exponent vectors. The output in `rees.out` coincides with that in `rproj2.out`, up to notions and the supplementary information on the integral closure of the ideal:

```
10 generators of integral closure of the ideal:
 0 0 1 1 0 1
 0 0 1 1 1 0
 0 1 0 0 1 1                1 0 0 1 0 1
 0 1 0 1 1 0                1 0 1 0 1 0
 0 1 1 0 0 1                1 1 0 1 0 0
 1 0 0 0 1 1                1 1 1 0 0 0
```

A brief look at `rproj2.out` shows that exactly the generators with the last coordinate 1 have been extracted. (So the ideal is integrally closed. This is not surprising because we have chosen squarefree monomials.)

6.2 Constraints

6.2.1 Type 4, `hyperplanes`

The file `dual.in` is

```
24
7
0 0 0 1 0 0  0             1 0 0 0 0 0  0
0 0 0 0 1 0  0             1 1 1 1 1 1 -3
0 0 0 0 0 1  0             1 0 0 1 0 1 -1
0 0 0 0 0 0  1             1 0 0 0 1 1 -1
0 0 1 0 0 0  0             1 0 1 0 1 0 -1
0 1 0 0 0 0  0             1 0 1 1 1 1 -2
0 1 0 1 1 0 -1             1 1 0 1 0 0 -1
0 1 0 0 1 1 -1             1 1 1 0 0 0 -1
0 1 1 0 0 1 -1             1 1 1 1 0 1 -2
0 0 1 1 1 0 -1             1 1 1 0 1 1 -2
0 0 1 1 0 1 -1             1 1 1 1 1 0 -2
0 1 1 1 1 1 -2             1 1 0 1 1 1 -2
                           hyperplanes
```

This means that we wish to compute the Hilbert basis of the cone cut out from $\mathbb{R}^7$ by the 24 inequalities. (It is the dual of the cone spanned by the 24 linear forms in $(\mathbb{R}^7)^*$.) The inequalities represent exactly the support hyperplanes from the file `rproj2.out`. The output in `dual.out` coincides with that in `rproj2.out`.

6.2.2 Type 5, `equations`

Suppose that you have the following "square"

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

and the problem is to find nonnegative values for $x_1, \ldots, x_9$ such that the 3 numbers in all rows, all columns, and both diagonals sum to the same constant $\mathscr{M}$. Sometimes such squares are called *magic* and $\mathscr{M}$ is the *magic constant*. This leads to a linear system of equations

$$\begin{aligned}
x_1+x_2+x_3 &= x_4+x_5+x_6;\\
x_1+x_2+x_3 &= x_7+x_8+x_9;\\
x_1+x_2+x_3 &= x_1+x_4+x_7;\\
x_1+x_2+x_3 &= x_2+x_5+x_8;\\
x_1+x_2+x_3 &= x_3+x_6+x_9;\\
x_1+x_2+x_3 &= x_1+x_5+x_9;\\
x_1+x_2+x_3 &= x_3+x_5+x_7.
\end{aligned}$$

This system of equations is contained in the file `3x3magic.in`. It has input type `equations`. (Don't forget that the sign conditions $x_i \geq 0$ are automatically included if there are no explicit inequalities.)

The magic constant is a natural choice for the grading, and therefore

```
1
9
1 1 1 0 0 0 0 0 0
grading
```

follows the equations.

The output file contains the following:

```
5 Hilbert basis elements
5 Hilbert basis elements of degree 1
4 extreme rays
4 support hyperplanes

rank = 3
index = 2
original monoid is not integrally closed

size of triangulation   = 2
resulting sum of |det|s = 4

multiplicity = 4

Hilbert series:
1 2 1
denominator with 3 factors:
1: 3

Hilbert polynomial:
1 2 2
with common denominator = 1
```

```
grading:
1 1 1 0 0 0 0 0 0
with denominator = 3

degrees of extreme rays:
1: 4

Hilbert basis elements are of degree 1

**********************************************************************

5 Hilbert basis elements:
 1 0 2 2 1 0 0 2 1
 0 2 1 2 1 0 1 0 2
 2 0 1 0 1 2 1 2 0
 1 2 0 0 1 2 2 0 1
 1 1 1 1 1 1 1 1 1

4 extreme rays:
 1 2 0 0 1 2 2 0 1
 2 0 1 0 1 2 1 2 0
 0 2 1 2 1 0 1 0 2
 1 0 2 2 1 0 0 2 1

4 support hyperplanes:
  0 -1  0  0  2  0  0  0  0
  0  1  2  0 -2  0  0  0  0
  0 -1 -2  0  4  0  0  0  0
  0  1  0  0  0  0  0  0  0

6 equations:
 -2  1  4 -3  0  0  0  0  0
 -1  0  1 -1  1  0  0  0  0
 -2  0  2 -1  0  1  0  0  0
 -2  0  3 -2  0  0  1  0  0
  0  0 -2  1  0  0  0  1  0
 -1  0  2 -2  0  0  0  0  1

5 Hilbert basis elements of degree 1:
 1 0 2 2 1 0 0 2 1
 0 2 1 2 1 0 1 0 2
 2 0 1 0 1 2 1 2 0
 1 2 0 0 1 2 2 0 1
 1 1 1 1 1 1 1 1 1
```

The 5 elements of the Hilbert basis represent the magic squares

2	0	1
0	1	2
1	2	0

1	0	2
2	1	0
0	2	1

1	1	1
1	1	1
1	1	1

1	2	0
0	1	2
2	0	1

0	2	1
2	1	0
1	0	2

All other solutions are linear combinations of these squares with nonnegative integer coefficients.

Normaliz tells us that the cone generated by the magic squares can be described by 4 inequalities and 6 linear relations. The number of equations becomes clear when we look at the rank.

The input degree is the magic constant. However. as the denominator 3 shows, the magic constant is always divisible by 3, and therefore the effective degree is $\mathscr{M}/3$. This degree is used for the multiplicity and the Hilbert series.

The Hilbert series is

$$\frac{1+2t+t^2}{(1-t)^3}.$$

The Hilbert polynomial is

$$P(k) = 1 + 2k + 2k^2,$$

and after substituting $\mathscr{M}/3$ for k we obtain the number of magic squares of magic constant $\mathscr{M}$.

6.2.3 Type 6, `congruences`

We change our definition of magic square by requiring that the entries in the 4 corners are all even. Then we have to augment the input file as follows (`3x3magiceven.in`):

```
7                              4
9                              10
1 1 1 -1 -1 -1  0  0  0        1 0 0 0 0 0 0 0 0 2
1 1 1  0  0  0 -1 -1 -1        0 0 1 0 0 0 0 0 0 2
0 1 1 -1  0  0 -1  0  0        0 0 0 0 0 0 1 0 0 2
1 0 1  0 -1  0  0 -1  0        0 0 0 0 0 0 0 0 1 2
1 1 0  0  0 -1  0  0 -1        congruences
0 1 1  0 -1  0  0  0 -1        1
1 1 0  0 -1  0 -1  0  0        9
equations                      1 1 1 0 0 0 0 0 0
                               grading
```

The output changes accordingly:

```
9 Hilbert basis elements                    multiplicity = 1
0 Hilbert basis elements of degree 1
4 extreme rays                              Hilbert series:
4 support hyperplanes                       1 -1 3 1
                                            denominator with 3 factors:
rank = 3                                    1: 1  2: 2
index = 4
original monoid is not integrally closed    Hilbert series with cyclotomic denominator:
                                            -1 1 -3 -1
size of triangulation   = 2                 cyclotomic denominator:
resulting sum of |det|s = 8                 1: 3  2: 2

grading:                                    Hilbert quasi-polynomial of period 2:
1 1 1 0 0 0 0 0 0                            0:   2  2  1
with denominator = 3                         1:  -1  0  1
                                            with common denominator = 2
degrees of extreme rays:
2: 4

**************************************************************************

9 Hilbert basis elements:        4 support hyperplanes:
 2 0 4 4 2 0 0 4 2                 1  0  1  0 -1  0  0  0  0
```

```
 0 4 2 4 2 0 2 0 4
 4 0 2 0 2 4 2 4 0
 2 4 0 0 2 4 4 0 2
 2 2 2 2 2 2 2 2 2
 4 3 2 1 3 5 4 3 2
 2 5 2 3 3 3 4 1 4
 2 3 4 5 3 1 2 3 4
 4 1 4 3 3 3 2 5 2

4 extreme rays:
 2 4 0 0 2 4 4 0 2
 4 0 2 0 2 4 2 4 0
 0 4 2 4 2 0 2 0 4
 2 0 4 4 2 0 0 4 2
```

```
 -1  0  1  0  1  0  0  0  0
  1  0 -1  0  1  0  0  0  0
 -1  0 -1  0  3  0  0  0  0

6 equations:
 -2  1  4 -3  0  0  0  0  0
 -1  0  1 -1  1  0  0  0  0
 -2  0  2 -1  0  1  0  0  0
 -2  0  3 -2  0  0  1  0  0
  0  0 -2  1  0  0  0  1  0
 -1  0  2 -2  0  0  0  0  1

2 congruences:
 0 0 1 0 0 0 0 0 0 2
 1 0 0 0 0 0 0 0 0 2

0 Hilbert basis elements of degree 1:
```

It is not surprising that the support hyperplanes have not changed after the introduction of the congruences, since the latter only modify the lattice $\mathbb{E}$. Similarly the number extreme rays is the same, but the vectors are multiplied by the factor 2 since Normaliz chooses them in $\mathbb{E}$, and therefore these vectors must satisfy the congruences.

Its first representation tells us that the Hilbert series is

$$\frac{1-t+3t^2+t^3}{(1-t)(1-t^2)^2}.$$

As in 6.1.3, the second representation gives coprime numerator and denominator polynomials in which the denominator is a product of cyclotomic polynomials:

$$\frac{-1+t-3t^2-t^3}{\zeta_1^3\zeta_2^2}, \qquad \zeta_1 = t-1, \zeta_2 = t+1.$$

In this case, the two denominators differ by the factor -1. In general, the first representation is not coprime, as we have seen in 6.1.3.

The lattice point enumerator is a quasipolynomial of period 2:

$$|\{x : \deg x = k\}| = \begin{cases} 1+k+k^2/2, & k \equiv 0 \quad (2) \\ -1/2+k^2/2, & k \equiv 1 \quad (2). \end{cases}$$

In general one must expect a non-integral multiplicity if the period is > 1. That the multiplicity is integral, namely 1, in this case must be considered an exception.

As you can see, the equations make two of the input congruences superfluous: it is enough to require the two corners in the first row to be even. The first congruence is to be read as $x_1 \equiv 0 \mod 2$, the second as $x_3 \equiv 0 \mod 2$.

Another good example for Hilbert series and gradings is given by `Condorcet.in`. The reader should run it or have a look at the corresponding output file.

6.3 Relations

6.3.1 Type 10, `lattice_ideal`

As an example, we consider the binomial ideal generated by

$$X_1^2X_2 - X_4X_5X_6,\ X_1X_4^2 - X_3X_5X_6,\ X_1X_2X_3 - X_5^2X_6.$$

We want to find an embedding of the toric ring it defines and the normalization of the toric ring.

The input ideal `lattice_ideal.in` is

```
3
6
2 1  0 -1 -1  -1
1 0 -1  2 -1  -1
1 1  1  0 -2  -1
lattice_ideal
```

It yields the output

```
6 original generators of the toric ring      multiplicity = 10
10 Hilbert basis elements
10 Hilbert basis elements of degree 1        Hilbert series:
5 extreme rays                               1 6 3
5 support hyperplanes                        denominator with 3 factors:
                                             1: 3
rank = 3 (maximal)
index = 1                                    Hilbert polynomial:
original monoid is not integrally closed     1 3 5
                                             with common denominator = 1
size of triangulation   = 3
resulting sum of |det|s = 10

grading:
1 -1 1

degrees of extreme rays:
1: 5

Hilbert basis elements are of degree 1

***********************************************************************

6 original generators:              5 support hyperplanes:
 0 1 2                                1  0  0
 3 2 0                                0  1  0
 0 0 1                                0  0  1
 1 1 1                                3 -2  1
```

```
 1 0 0                                6 -9  7

 1 3 3                              10 Hilbert basis elements of degree 1:
10 Hilbert basis elements:           1 3 3
 1 3 3                               1 0 0
 1 0 0                               1 1 1
 1 1 1                               0 0 1
 0 0 1                               3 2 0
 3 2 0                               0 1 2
 0 1 2                               2 1 0
 2 1 0                               1 2 2
 1 2 2                               2 2 1
 2 2 1                               1 1 1
 1 1 1

5 extreme rays:
 0 1 2
 3 2 0
 0 0 1
 1 0 0
 1 3 3
```

The 6 original generators correspond to the indeterminates $X_1, \ldots, X_6$ in the binomial equations. They represent an embedding of the affine monoid defined by the binomial equations.

7 Optional output files

When one of the options `-f`, `-a` or `-T` is activated, Normaliz writes additional output files whose names are of type `<projectname>.<type>`. The format of the files (with the exception of `inv`) is completely analogous to that of the input file, except that there is usually no last line denoting the type. The main purpose of these files is to give the other systems easy access to the results of Normaliz without complicated parsing. The packages for Singular and Macaulay 2 use the extra output files to retrieve the results of Normaliz. Furthermore they provide additional information not contained in the standard output file.

The option `-f` makes Normaliz write the following files:

`gen` contains the Hilbert basis.

`cst` contains the constraints defining the cone and the lattice in the same format as they would appear in the input: matrices of types 4,5,6 following each other. Each matrix is concluded by the integer denoting its type. Empty matrices are indicated by 0 as the number of rows. Therefore there will always be at least 3 matrices.

If a grading is defined, it will be appended. Therefore this file (with suffix `in`) as input for Normaliz will reproduce the Hilbert basis and all the other data computed, at least in principle.

`inv` contains all the information from the file `out` that is not contained in any of the other files.

If `-a` is activated, then the following files are written *additionally:*

`typ` This is the product of the matrices corresponding to `egn` and `esp`. In other words, the linear forms representing the support hyperplanes of the cone C are evaluated on the Hilbert basis. The resulting matrix, with the generators corresponding to the rows and the support hyperplanes corresponding to the columns, is written to this file.
The suffix `typ` is motivated by the fact that the matrix in this file depends only on the isomorphism type of monoid generated by the Hilbert basis (up to row and column permutations). In the language of [2] it contains the *standard embedding*.

`ext` contains the extreme rays of the cone.

`ht1` contains the degree 1 elements of the Hilbert basis if a grading is defined.

`egn,esp` These contain the Hilbert basis and support hyperplanes in the coordinates with respect to a basis of $\mathbb{E}$.`egn` contains the grading in the coordinates of $\mathbb{E}$ if one exists. Note that no equations for $C \cap \mathbb{E}$ or congruences for $\mathbb{E}$ are necessary.

The option `-T` (independently from `-f` or `-a`) writes the triangulation data:

`tgn, tri` These files together describe the triangulation computed by Normaliz.
The file `tri` lists the simplicial subcones as follows: The first line contains the number of simplicial cones in the triangulation, and the next line contains the number $m+1$ where $m = \operatorname{rank}\mathbb{E}$. Each of the following lines specifies a simplicial cone Δ: the first m numbers are the indices (with respect to the order in the file `tgn`) of those generators that span Δ, and the last entry is the multiplicity of Δ in $\mathbb{E}$, i. e. the absolute value of the determinant of the matrix of the spanning vectors (as elements of $\mathbb{E}$).
If `-t` is combined with `-T`, then the determinants have not been computed, and the last entry of each row is 0 (a forbidden value for the determinant).
The file `tgn` contains a matrix of vectors (in the coordinates of $\mathbb{A}$) spanning the simplicial cones in the triangulation.
Note that these files are not generated with the modes `-d`, `-1` or `-N`.

The file `3x3magiceven.in` has been processed with the option `-aT` activated. We recommend you to inspect all the output files in the subdirectory `example` of the distribution.

8 Performance and parallelization

The executables of Normaliz have been compiled for parallelization on shared memory systems with OpenMP. Parallelization reduces the "real" time of the computations considerably, even on relatively small systems. However, one should not underestimate the administrational overhead involved.

- It is not a good idea to use parallelization for very small problems.
- On multi-user systems with many processors it may be wise to limit the number of threads for Normaliz somewhat below the maximum number of cores.

The number of parallel threads can be limited by the Normaliz option `-x` (see Section 4.3) or by the commands

```
export OMP_NUM_THREADS=<T>      (Linux/Mac)
```

or

```
set OMP_NUM_THREADS=<T>      (Windows)
```

where `<T>` stands for the maximum number of threads accessible to Normaliz. We use

```
export OMP_NUM_THREADS=16
```

on a multi-user system system with 24 cores.

Limiting the number of threads to 1 forces a strictly serial execution of Normaliz.

First we compare the performance of Normaliz on several processor configurations. (The table shows Real times in seconds.)

	mode	i5 M520	i7 870	Xeon	Xeon
cores/threads		2 cor, 4 thr	4 cor, 8 thr	24 cor, 1 thr	24 cor, 16 thr
`medium`	`-h`	1.2	0.7	2.3	0.7
`A443`	`-h`	22	9.4	56	9.3
`A543`	`-h`	1320	466	3580	226
`A443`	`-N`	0.5	0.4	0.7	0.3
`A543`	`-N`	26	9.2	52	7
`A553`	`-N`	4430	1500	14940	1350

See [6] for further performance data of challenging examples.

All computation times are based on the Linux version of `normaliz`.

Finally we compare the primal and the dual algorithm on several examples (computation times measured on the Xeon system with 16 threads).

	`-N`	`-d`
`dual`	0.06	0.004
`small`	3.6	383
`rafad`	9.5	∞
`5x5`	1940	1
`6x6`	∞	12660

As a rule of thumb, one should use `-d` if the number of extreme rays is at least one magnitude larger than that of support hyperplanes. Therefore a previous run with `-s` may help in choosing the right approach. The example `small` is discussed extensively in [5].

9 Running large computations

Normaliz can cope with very large examples, but it is usually difficult to decide a priori whether an example is very large, but nevertheless doable, or simply impossible. Therefore some exploration makes sense.

See [6] for some very large computations. The following hints reflect the authors' experience with them.

(1) Run Normaliz with the option `-cs` and pay attention to the terminal output. The number of extreme rays, but also the numbers of support hyperplanes of the intermediate cones are useful data.

(2) In many cases the most critical size parameter is the number of simplicial cones in the triangulation. It makes sense to determine it as the next step. Even with the fastest potential evaluation (option `-v`), finding the triangulation takes less time, say by a factor between 3 and 10. Thus it makes sense to run the example with `-t` in order to explore the size.

As you can see from [6], Normaliz has successfully evaluated triangulations of size $\approx 5 \cdot 10^{11}$ in dimension 24.

(3) Another critical parameter are the determinants of the generator matrices of the simplicial cones. To get some feeling for their sizes, one can restrict the input to a subset (of the extreme rays computed in (1)) and use the option `-v`.

The output file contains the number of simplicial cones as well as the sum of the absolute values of the determinants. The latter is the number of vectors to be processed by Normaliz in triangulation based calculations.

The number includes the zero vector for every simplicial cone in the triangulation. The zero vector does not enter the Hilbert basis calculation, but cannot be neglected for the Hilbert series.

Normaliz has mastered calculations with $> 10^{15}$ vectors.

(4) If the triangulation is small, we can add the option `-T` is order to actually see the triangulation in a file. Then the individual determinants become visible.

(5) If a cone is defined by inequalities and/or equations consider the dual mode for Hilbert basis calculation, even if you also want the Hilbert series.

(6) The size of the triangulation is *not* dangerous for memory (unless `-T` is set).

10 Distribution and Installation

In order to install Normaliz you should first download the basic package containing the documentation, examples, source code, jNormaliz and the packages for Singular and Macaulay 2. Then unzip the downloaded file `Normaliz2.8.zip` in a directory of your choice. (Any other downloaded zip file for Normaliz should be unzipped in this directory, too.)

This process will create a directory `Normaliz2.8` (called Normaliz directory) and several sub-

directories in Normaliz2.8. The names of the subdirectories created are self-explanatory. Nevertheless we give an overview:

- In the main directory Normaliz2.8 you should find jNormaliz.jar, Copying and subdirectories.
- In the subdirectory source contains the source files and a Makefile for compilation with GCC.
- Subdirectory doc contains the file you are reading and further documentation.
- In the subdirectory example are the input and output files for some examples It contains all input files of examples of this documentation, except the toy examples of Section 3. Some very large output files are contained in an extra zip file accessible from the Normaliz home page.
- The subdirectory singular contains the SINGULAR library normaliz.lib and a PDF file with documentation.
- The subdirectory macaulay2 contains the MACAULAY2 package Normaliz.m2.
- The subdirectory lib contains libraries for jNormaliz.

We provide executables for Windows, Linux (each in a 32 bit and a 64 bit version) and Mac. Download the archive file corresponding to your system Normaliz2.8<systemname>.zip and unzip it. This process will store the the executable in the directory Normaliz2.8. In case you want to run Normaliz from the command line or use it from other systems, you may have to copy the executables to a directory in the search path for executables.

For backward compatibility, we provide shell scripts (batch files) norm64 and normbig. They call normaliz with the parameters passed to them, augmented by -B in the case of normbig.

Please remove old versions of normaliz, norm64 and normbig from your search path.

11 Compilation

11.1 GCC

Produce the executables by calling make in the subdirectory source. You may have to transport the executables to a directory in your search path. **jNormaliz expects them in its own directory.**

Note that normaliz needs GMP (including the C++ wrapper) and the Boost collection. Therefore you must install them first.

We are using OpenMP 3.0. Please make sure that your GCC version is compatible with it (version $\geq$ 4.4).

Note the following **exceptions:**

1. One can compile Windows executables with the Cygwin port of GCC. Unfortunately it is not compatible to OpenMP.
2. Mac versions of GCC older than 4.5 have a bug that makes it impossible to use OpenMP.

In any case, or if you want to avoid parallelization, you can call `make OPENMP=no`.

11.2 Visual Studio project

The Windows executables provided by us have been compiled with MS Visual Studio and Intel C++ Composer XE. (Visual C++ itself can only be used without OpenMP.)

If you want to compile Normaliz yourself in this way, please unzip the corresponding zip file on the Normaliz home page. This will create a subdirectory `Visual Studio` of the Normaliz directory. This directory contains the predefined project. We have provided

1. two configurations: `Release` (with OpenMP) and `ReleaseSerial` (without OpenMP), and
2. two platforms, `Win32` and `x64`.

Instead of GMP we use the MPIR library for the Windows version of `normaliz`. For convenience, the MPIR files have been included in the distribution (in the subdirectory `MPIR` of `Visual Studio`). Please

- copy the library files for Win32 into the `lib` subdirectory of the Visual C++ compiler,
- the library files for x64 to the subdirectory `amd64` (or `x64`) of `lib`, and
- the two header files to the `include` subdirectory of the compiler.

Moreover, you must install the Boost collection available from http://www.boost.org/. We only use Boost libraries that are entirely implemented in their headers. So the only preparation beyond downloading and unzipping is to add the Boost root directory to the list of include paths. In the Visual Studio C++ IDE, click "Tools | Options... | Projects | VC++ directories". Then, in "Show Directories for", select "Include files" and add the path to the Boost root directory.

After the compilation with the Intel compiler you must copy the executable to the directories where they are expected (the Normaliz directory or a directory in the search path).

The source files for Visual Studio are identical to those for GCC.

12 Changes relative to version 2.5

For the history of changes starting from 2.0 see the manual of version 2.7 (still accessible on the web site). Note that some changes have become obsolete later on.

Changes in version 2.7:

User control, input and output:

1. Only one executable `normaliz`. Precision controlled by option `-B`.
2. Slight changes in the wording of the main output file.
3. Introduction of options for large problems. (Obsolete.)

Algorithms and implementation:

1. Separation of front end and kernel (implemented as a library).
2. Pyramid based algorithms for large problems (see [6]).
3. New algorithm for h-vector. No computation of line shellings in this version.
4. Dual mode accessible from all input types.
5. General improvement of memory use (and speed) by more efficient data types.

Changes in version 2.8:

User control, input and output:

1. Use of arbitrary $\mathbb{Z}$-gradings which make rational polytopes accessible.
2. Implied changes in the output files.
3. Simplification of the command line options. ("Large" modes now superfluous.)

Algorithms and implementation:

1. Handling of arbitrary $\mathbb{Z}$-gradings.
2. Substantial improvement of parallelization, based on thorough use of pyramid decompositions (see [6]).
3. Faster evaluation of simplicial cones (see [6]).
4. General overhaul of the code.

13 Copyright and how to cite

Please refer to Normaliz in any publication for which it has been used:

W. Bruns, B. Ichim and C. Söger: Normaliz. Algorithms for rational cones and affine monoids. Available from http://www.math.uos.de/normaliz.

It is now customary to evaluate mathematicians by such data as numbers of publications, citations and impact factors. The data bases on which such dubious evaluations are based do not list mathematical software. Therefore we ask you to cite the article [6] in addition. This is very helpful for the younger members of the team.

References

[1] V. Almendra and B. Ichim. *jNormaliz 1.1*.

[2] W.Bruns and J. Gubeladze. *Polytopes, rings, and K-theory*. Springer 2009.

[3] W.Bruns, R. Hemmecke, B. Ichim, M. Köppe and C. Söger. *Challenging computations of Hilbert bases of cones associated with algebraic statistics* . Exp. Math.20 (2011), 25–33.

[4] W.Bruns and J. Herzog. *Cohen-Macaulay rings*. Rev. ed. Cambridge University Press 1998.

[5] W.Bruns and B. Ichim. *Normaliz: algorithms for rational cones and affine monoids.* J. Algebra **324** (2010) 1098–1113.

[6] W.Bruns, B. Ichim and C. Söger. *The power of pyramid decompositions in Normaliz.* Preprint arXiv:1206.1916v1.

[7] W.Bruns and R. Koch. *Computing the integral closure of an affine semigroup*. Univ. Iagiell. Acta Math. **39** (2001), 59–70.

[8] M. Köppe and S. Verdoolaege. *Computing parametric rational generating functions with a primal Barvinok algorithm.* Electron. J. Comb. 15, No. 1, Research Paper R16, 19 p. (2008).

[9] L. Pottier. *The Euclide algorithm in dimension n*. Research report, ISSAC 96, ACM Press 1996.