NORMALIZ
Computing normalizations of affine semigroups

Winfried Bruns   Robert Koch

with contributions by
Witold Jarnicki   Matthias Siemering

June 28, 2006

Contents

1 Documentation of the program

1.1 Objectives

Our program NORMALIZ computes

(1) the normalization (or integral closure) of an affine semigroup or, in other terms, the Hilbert basis of a rational cone;
(2) the support hyperplanes of the cone;
(3) the lattice points and
(4) the support hyperplanes of an integral polytope;
(5) the generators of the integral closure of the Rees algebra of a monomial ideal $I \subseteq K[X_1, \dots, X_n]$;
(6) the generators of the integral closure of I;
(7) the Hilbert series and Hilbert polynomial of the semigroup in the homogeneous case.

For the theory of affine semigroups and the notions of commutative algebra used in the following we refer the reader to [BH].

There exists a SINGULAR library `normaliz.lib` that makes NORMALIZ accessible from SINGULAR. Thus SINGULAR 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.

1.2 Numerical aspects and limitations

There are two versions of our program: `normaliz` works with 32 bit integers, and `enormalz` uses integers of arbitrary precision. As far as we have observed, `normaliz` is able to do all heuristic examples. But if you feel that some arithmetical problem could have arisen, we recommend using `enormalz`. (It should, however, be clear that this version is significantly slower.)

Nevertheless there is a common limitation for both versions: If you wish to compute the Hilbert series and polynomial along with the generators of the semigroup, `normaliz` and `enormalz` will only work in dimension ≤ 32.

1.3 Distribution

We distribute our program NORMALIZ through the file `normaliz.zip` which can be found in the directory

```
ftp://ftp.mathematik.Uni-Osnabrueck.DE/pub/osm/kommalg/software/
```

on our FTP server. You can log in as `anonymous`.

Download the file `normaliz.zip` to a directory of your choice and unzip it by INFOZIP's `unzip` (or PKUNZIP or WINZIP (take care of the option "create subdirectories")). The names of the subdirectories created are self-explanatory. Nevertheless, some comments may be useful.

Executables for Win32, Sparc (Solaris), Linux (Redhat) and Mac are provided separately. The names of the `zip` files are self-explanatory.

1.3.1 `source/`

Here you find the 14 source files (and a `Makefile`):

```
alloc.c  elemdiv.c  enormalz.cc  error.c  heap.c   hilbert.c  hilbert.h
 homog.c  invest.c  linequ.c  normaliz.cc  reduce.c  shorten.c  sigma.c
```

1.3.2 `doc/`

This contains the LaTeX 2ε documentation file `normaliz.tex`, the compiled version `normaliz.pdf` and a `Makefile`.

1.3.3 `example/`

Here are the input and output files of 8 examples, called `rproj2`, `rafa1409`, `squaref0`, `squaref1`, `rafa2310`, `rafa2416`, `polytop` and `rees`. We thank Rafael Villarreal, who sent us some of these examples.

1.3.4 `Singular/`

It contains the SINGULAR library `normaliz.lib` and the documentation files `nmz_sing.tex` and `nmz_sing.pdf`.

1.4 Compilation

A *Makefile* is provided with the source code of NORMALIZ. The `enormalz` binary is built simply by entering the command

```
make
```

whereas the `normaliz` binary is built by specifying the target through

```
make normaliz.
```

Note that it is crucial to have two libraries installed to link the enormalz binary properly. We use the `NTL` library[1] as a `C++` interface for the `GMP` library[2] which provides thee long integer arithmetic for `enormalz`. (The corresponding variables in the Makefile may have to be customized to your library directory.)

Via the makefile the source code compiles under `Unix`, `Windows/Cygwin`, `Linux/Redhat` and `Macintosh` (provided that the used compiler is compatible with the widespread `GCC`).

However, for `Windows` user we recommend the `DJGPP` compiler. It produces much more efficient code. Since we provide executables for `Windows`, we omit the details of using `DJGPP`.

1.5 Executables

Executables for various platforms are provided separately in the `ftp` folder.

1.6 Running the program

The program `normaliz` is started by the following command line:

```
normaliz [-acfhv] <filename>
```

It expects its input in the file `<filename>.in` and writes its output into `<filename>.out`. Therefore the argument `<filename>` in your command line must *not* contain the suffix `.in`.

With the option `-f`, not only the standard output file `<filename>.out` will be written, but also several other files. (See Section 1.6.3 for details.)

With the option `-a`, all potentially useful output files will be written (See Section 1.6.3 for details.) The `-a` option includes `-f`.

If you include the option `-h` in your command line, the h-vector and the (coefficients of the) Hilbert polynomial will be written into `<filename>.out`. But note that this will only work if the semigroup is homogeneous. (See Section 1.6.3 for a definition.)

The `-v` option overrides the `-h` option and restricts `normaliz` to those computations which determine the multiplicity. The name has been chosen since this option is particularly interesting for polytopal applications (see the “mode = 2” case below) if one is only interested in effectively computing the volume of a lattice polytope.

Finally, let us explain the `-c` option. This will give you some access to ‘control’ data during the computation. It is designed for users who run complex examples and wish to see how far

[1] `http://www.shoup.net/ntl`
[2] `http://www.swox.com/gmp/`

`normaliz` has come (and if it is still running at all). In the first part of the computation the information is given as

`Done with generator` n

where n runs from $\operatorname{rank} S + 1$ to the number of generators of S. (S is the semigroup whose integral closure is to be computed; it depends on the mode; see below.)

In the second part of the computation the information is given in the format

[<*block*>.<*subblock*>] <*current number*> / <*total number*>

and refers to the triangulation discussed below. The triangulation is partitioned into blocks and subblocks, and the first two counters refer to these, whereas *current number* and *total number* describe the position within the current subblock. (See Section 2.2 and the final remark at the end of the paper for more information.)

1.6.1 The input file

The input file `<filename>.in` is structured as follows.

The first line contains the number of generators of the semigroup S (or the number of lattice points spanning the polytope, or the number of generators of the ideal I defining the Rees algebra).

The second line contains the dimension of the ambient lattice.

The next lines contain the generators of S (or the spanning lattice points, or the exponent vectors of the monomials generating the ideal I, respectively), as shown in the examples below.

The last line contains a single digit, called *mode*, namely:

0, if the program should compute the integral closure of S in the ambient lattice (the Hilbert basis of the cone spanned by the generators of S);
1, if the normalization of S is to be computed (i.e. the integral closure in the sublattice generated by S);
2, if the integral points in a polytope and its Ehrhart semigroup are to be computed;
3, if the integral closure of the Rees algebra of I is to be computed.

1.6.2 The lattices

The output of NORMALIZ depends on the lattices involved. Therefore we define the *ambient lattice* $\mathbb{A}$, the semigroup S and the *effective lattice* $\mathbb{E}$ as follows:

mode 0: $\mathbb{A} = \mathbb{Z}^n$ where n is the number contained in the second line of the input file;
S is the subsemigroup of $\mathbb{A}$ generated by the vectors in the input file;
$\mathbb{E}$ is the smallest direct summand of $\mathbb{A}$ containing the subgroup $\operatorname{gp}(S) \subset \mathbb{A}$ generated by S.

mode 1: $\mathbb{A}$ and S as for mode 0, but $\mathbb{E} = \operatorname{gp}(S)$.

mode 2 (polytopal application): $\mathbb{A} = \mathbb{Z}^{n+1}$;
S is generated by the vectors $(x, 1)$ for the vectors x in the input file;
$\mathbb{E}$ now defined as for mode 0.

mode 3 (Rees application): $\mathbb{A} = \mathbb{Z}^{n+1}$;

S is generated by the unit vectors $e_1, \dots, e_n$ (representing the indeterminates of the polynomial ring) and the vectors $(x, 1)$ for the vectors x in the input file;

$\mathbb{E} = \mathbb{Z}^{n+1}$.

In each case the integral closure of S in $\mathbb{E}$ is computed. See also Subsection 2.1.

1.6.3 The output file(s)

In the case "mode ≤ 1" the output file `<filename>.out` contains the following data:

- the generators of the integral closure of S in $\mathbb{E}$;
- if $\operatorname{rank}\mathbb{E} = \operatorname{rank}\mathbb{A}$, the extreme rays of the cone C generated by S and the (unique) support hyperplanes of C;
- $\operatorname{rank}\operatorname{gp}(S)$;
- the index of $\operatorname{gp}(S)$ in $\mathbb{E}$ in mode 0 (in mode 1 it is 1 by default);
- if S is homogeneous (see below), you will find its multiplicity in the output file, too;
- if, in addition, you have used the option `-h`, there are also the h-vector and the coefficients of the Hilbert polynomial.

Note that the support hyperplanes are not listed if $\operatorname{rank}\mathbb{E} < \operatorname{rank}\mathbb{A}$. However, in this case a description of the cone generated by S in $\mathbb{R}\mathbb{E}$ is available if the option `-a` is used (see below).

We call S *homogeneous* if there is an integer-valued linear form φ on $\mathbb{E}$ such that all the generators v of S given in the input file satisfy $\varphi(v) = 1$. For instance, the examples `rafa2416`, `squaref1` and `rproj2` from the directory `example` are homogeneous.

Recall that the option `-v` suppresses the computation of the Hilbert basis and therefore restricts the output to rank, index, support hyperplanes, and multiplicity. Thus it may happen that, using `-v`, the output file becomes completely empty! In the modes ≥ 2, this option behaves accordingly. It was introduced for users who wish to efficiently determine only the (normalized) volume of a lattice polytope. This application will be discussed now.

The extreme rays are given by the elements in S that define the extreme rays. Therefore these vectors need not have coprime entries.

Attention. `normaliz` uses 32 bit arithmetic for all its computation, including Hilbert functions. It can very well happen, that the computation of the integral closure of S runs without problems, but an overflow occurs in the Hilbert function computation. In this case you must resort to `enormalz`.

A good test for the h-vector is whether its entries are nonnegative and sum up to the multiplicity. *To get a correct denominator for the Hilbert polynomial with* `normaliz`, *one needs* $\operatorname{rank} S \leq 13$. If this condition is not fulfilled, `normaliz` does not compute the Hilbert polynomial.

If "mode $= 2$", the following data can be found in the output file:

- the generators of the semigroup determined by the polytope, called Ehrhart semigroup in the following;
- the lattice points of the polytope;
- its normalized volume;

- its extreme points and its support hyperplanes if it is of full dimension;
- if you have put the option `-h`, there are also the h-vector and the coefficients of the Ehrhart polynomial.

In "mode = 3", the output file contains the following:

- the generators of the integral closure $\bar{\mathscr{R}}$ of the Rees algebra;
- the generators of the integral closure of the ideal;
- the extreme rays and the support hyperplanes;
- the multiplicity of the semigroup if it is homogeneous;
- if the ideal is primary to the irrelevant maximal ideal, the multiplicity of the ideal (not to be confused with the multiplicity of the semigroup);
- if, in addition, the `-h` option has been used, the h-vector and the coefficients of the Hilbert polynomial of $\bar{\mathscr{R}}$ are computed, too.

1.6.4 The `-f` and `-a` options

With the `-f` or `-a` option in the command line, NORMALIZ writes additional output files whose names are of type `<filename>.<type>`. In the following we denote the files simply by their types. Note that `-a` includes `-f`.

With the `-f` option the following files are written, provided the information that should go into them is available:

`gen` The generators of the integral closure are written to the file `out` (provided they have been computed). The format of this file and the other files (with the exception of `inv`) is completely analogous to that of the input file, except that there is no last line denoting the mode.

`sup,val` If $\operatorname{rank}\mathbb{A} = \operatorname{rank}\mathbb{E}$, then the support hyperplanes are written to `sup`. In this case, the support hyperplanes of the cone C are evaluated (as linear forms) on the generators, too. The resulting matrix, with the generators corresponding to the rows and the support hyperplanes corresponding to the columns, is written to the file `val`.

`inv` The file `inv` contains all the information from the file `out` that is not contained in any of the other files. It helps to import these data into `Singular`.

With the `-a` option the following additional files are written, provided certain conditions are satisfied and the information that should go into them is available:

`ext` The file `ext` contains the extreme rays (or points in polytopal mode), provided they have been computed.

`egn,esp,evl` If $\operatorname{rank}\mathbb{E} < \operatorname{rank}\mathbb{A}$ or the mode is 1, there are corresponding files `egn`, `esp` and `evl`. These are defined as `gen`, `sup` and `val`, however with respect to the lattice $\mathbb{E}$ and a basis of $\mathbb{E}$.

Note that these data provide a description of the integral closure of S in the form $\mathbb{E} \cap C$.

`tri` The file `tri` contains the triangulation of the cone C computed by NORMALIZ.

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 input file) of those generators of S 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 $\mathbb{E}$).

1.6.5 Examples

The file `rproj2.in` contains the following:

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

This means that we wish to compute the integral closure of the semigroup generated by the 16 vectors

$$[1,0,0,0,0,0,0], \quad [0,1,0,0,0,0,0], \quad \dots, \quad [0,0,1,1,0,1,1]$$

in dimension 7. We compute it in the ambient lattice $\mathbb{Z}^7$, which is indicated by the final digit 0.

Calling NORMALIZ by the command

```
normaliz rproj2
```

produces the file `rproj2.out` which has the following content:

```
17 generators of integral closure:
  1  0  0  0  0  0  0
  0  1  0  0  0  0  0
  0  0  1  0  0  0  0
  0  0  0  1  0  0  0
  0  0  0  0  1  0  0
  0  0  0  0  0  1  0
  1  1  1  0  0  0  1
  1  1  0  1  0  0  1
  1  0  1  0  1  0  1
  1  0  0  1  0  1  1
  1  0  0  0  1  1  1
  0  1  1  0  0  1  1
  0  1  0  1  1  0  1
  0  1  0  0  1  1  1
  0  0  1  1  1  0  1
  0  0  1  1  0  1  1
  1  1  1  1  1  1  2

16 extreme rays:
  1  1  1  0  0  0  1
  1  1  0  1  0  0  1
  1  0  1  0  1  0  1
  1  0  0  1  0  1  1
  1  0  0  0  1  1  1
  0  1  1  0  0  1  1
  0  1  0  1  1  0  1
  0  1  0  0  1  1  1
  0  0  1  1  1  0  1
  0  0  1  1  0  1  1
  1  0  0  0  0  0  0
  0  1  0  0  0  0  0
  0  0  1  0  0  0  0
  0  0  0  1  0  0  0
  0  0  0  0  1  0  0
  0  0  0  0  0  1  0

(original) semigroup has rank 7 (maximal)
(original) semigroup is of index 1

24 support hyperplanes:
   1   1   1   1   1   0  -2
   1   1   0   1   1   1  -2
   1   1   1   0   1   1  -2
   0   1   1   0   0   1  -1
   0   0   0   0   0   1   0
   1   0   0   1   0   1  -1
   1   0   1   1   1   1  -2
   1   1   1   0   0   0  -1
```

```
   1   1   0   1   0   0  -1
   1   0   1   0   1   0  -1
   0   0   0   0   1   0   0
   0   1   0   1   1   0  -1
   1   0   0   0   0   0   0
   1   1   1   1   1   1  -3
   1   0   0   0   1   1  -1
   0   1   0   0   1   1  -1
   0   1   0   0   0   0   0
   1   1   1   1   0   1  -2
   0   0   1   1   1   0  -1
   0   0   0   1   0   0   0
   0   0   1   1   0   1  -1
   0   0   1   0   0   0   0
   0   1   1   1   1   1  -2
   0   0   0   0   0   0   1

(original) semigroup is homogeneous via the linea
   1   1   1   1   1   1   -2

multiplicity = 72
```

From this, we see that there are 17 generators of the integral closure of the semigroup in $\mathbb{Z}^7$ and 16 extreme rays, that the semigroup has index 1 in $\mathbb{Z}^7$, and that the corresponding support hyperplanes are given by the linear forms $[1,1,1,1,1,0,-2]$, $[1,1,0,1,1,1,-2]$, $\ldots$, $[0,0,0,0,0,0,1]$. Moreover, we are given the information that the semigroup is homogeneous and that its multiplicity is 72.

If we use the same input file `rproj2.in` and call `normaliz` with the option `-h`, `rproj2.out` will contain the additional lines

```
h-vector =  1  9  31  25  6  0  0

Hilbert poly :  1  97/30  71/15  49/12  13/6  41/60  1/10
```

which state that the h-vector of $\bar{S}$ is

$$(h_0, h_1, \ldots, h_6) = (1, 9, 31, 25, 6, 0, 0),$$

and that the Hilbert polynomial of $\bar{S}$ is given by

$$P_{\bar{S}}(t) = 1 + \frac{97}{30}t + \frac{71}{15}t^2 + \frac{49}{12}t^3 + \frac{13}{6}t^4 + \frac{41}{60}t^5 + \frac{1}{10}t^6 .$$

Here is another example from the file `polytop.in`:

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

Here the lattice points 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 $\mathbb{R}^3$ are to be computed. (Note the last line, indicating the polytopal mode 2.)

The command `normaliz polytop` produces the file `polytop.out`:

```
19 generators of Ehrhart ring:
  0  0  0  1
  2  0  0  1
  0  3  0  1
  0  0  5  1
  1  2  4  2
  0  1  3  1
  1  0  2  1
  0  2  1  1
  1  1  0  1
  0  0  4  1
  0  1  2  1
  1  0  1  1
  0  2  0  1
  0  0  3  1
  0  1  1  1
  1  0  0  1
  0  0  2  1
  0  1  0  1
  0  0  1  1

18 lattice points in polytope:
  0  0  0
  2  0  0
  0  3  0
  0  0  5
  0  1  3
  1  0  2
  0  2  1
  1  1  0
  0  0  4
  0  1  2
  1  0  1
  0  2  0
  0  0  3
  0  1  1
  1  0  0
  0  0  2
  0  1  0
  0  0  1

4 extreme points of polytope:
  0  0  0
  2  0  0
  0  3  0
  0  0  5

4 support hyperplanes:
    0    0    1   >=    0
    0    1    0   >=    0
    1    0    0   >=    0
  -15  -10   -6   >=  -30

normalized volume = 30
```

The desired lattice points are the 18 ones listed above. To complete the picture, we also provide all the generators of the Ehrhart ring of the polytope. (There are 19 of them in this example.) Furthermore, the original polytope is the solution of the system of the 4 inequalities

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

and has normalized volume 30.

Again, calling NORMALIZ via the command `normaliz -h polytop` produces additional output in `polytop.out`, namely

```
h-vector =  1  14  15  0

Ehrhart poly :  1  4  8  5
```

This provides the information that the h-vector of the Ehrhart ring is

$$(h_0,h_1,h_2,h_3) = (1,14,15,0),$$

and its Ehrhart polynomial is

$$P(t) = 1 + 4t + 8t^2 + 5t^3.$$

To complete the picture, let us discuss the example in `rees.in`:

```
10
6
1 1 1 0 0 0
1 1 0 1 0 0
1 0 1 0 1 0
1 0 0 1 0 1
1 0 0 0 1 1
0 1 1 0 0 1
0 1 0 1 1 0
0 1 0 0 1 1
0 0 1 1 1 0
0 0 1 1 0 1
3
```

Comparing with the data in `rproj2.in` shows that `rees` is the origin of `rproj2`. (For details see the comments on the reduction of item (3) on page 13.)

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. (Note again the last line, containing 3 in this case.) 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:
  1  1  1  0  0  0
  1  1  0  1  0  0
  1  0  1  0  1  0
  1  0  0  1  0  1
  1  0  0  0  1  1
  0  1  1  0  0  1
  0  1  0  1  1  0
  0  1  0  0  1  1
  0  0  1  1  1  0
  0  0  1  1  0  1
```

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.)

1.7 Copyright

You can use this program freely, provided that you refer to it in the following manner in any publication for which it has been used:

W. Bruns, R. Koch et al.: NORMALIZ, computing normalizations of affine semigroups. Available via anonymous ftp from
`ftp://ftp.mathematik.uni-osnabrueck.de/pub/osm/kommalg/software/`

2 Algorithmic and mathematical background

2.1 Introduction

As can be seen from Section 1.1, we are faced with the following situation.
Given a finite set

$$E \subseteq \mathbb{Z}^n,$$

consider the *affine semigroup*

$$S = S(E) = \mathbb{N} \cdot E = \sum_{v \in E} \mathbb{N} \cdot v$$

generated by E, and its *integral closure*

$$\bar{S}_{\mathbb{Z}^n} = \{x \in \mathbb{Z}^n \,|\, x \text{ integral over } S\} = \{x \in \mathbb{Z}^n \,|\, k \cdot x \in S \text{ for some } k \geq 1\}$$

in $\mathbb{Z}^n$, or its *normalization*

$$\bar{S}_{\mathbb{Z}E} = \{x \in \mathbb{Z}E \,|\, x \text{ integral over } S\}$$

(integral closure in $\mathbb{Z}E$).
Now let

$$C = \operatorname{cone}(E) = \mathbb{R}_{\geq 0} \cdot E = \sum_{v \in E} \mathbb{R}_{\geq 0} \cdot v\,.$$

The importance of the cone C becomes clear from the following

Remark 1: (i) *$\bar{S}_L = C \cap L$ for $L = \mathbb{Z}^n, \mathbb{Z}E$.*
(ii) *$\bar{S}_L$ is a finitely generated semigroup.*

General assumption: In the following, we always assume that C is a strictly convex cone, i.e. C does not contain any nontrivial linear subspace, i.e. for all $x \in C$ we have:

$$-x \in C \quad \Longrightarrow \quad x = 0\,.$$

Under this assumption, we call an element $v \in \bar{S}_L$ *irreducible* if a decomposition

$$v = v_1 + v_2 \quad \text{with } v_i \in \bar{S}_L$$

implies $v_1 = 0$ or $v_2 = 0$.
As you may recall, NORMALIZ is able to compute (compare with the four "modes")

(0) the integral closure of an affine semigroup in $\mathbb{Z}^n$;
(1) the normalization of an affine semigroup;
(2) the lattice points of an integral polytope and its Ehrhart ring;
(3) the integral closure of a monomial ideal $I \subseteq K[X_1,\dots,X_n]$ and the integral closure of its Rees algebra;
(H) the Hilbert series and Hilbert polynomial.

Obviously, (1) can be reduced to (0) by performing a suitable change of coordinates (in order to pass to the effective lattice $\mathbb{E}$). As for (2), let $v_1,\dots,v_m \in \mathbb{Z}^n$ be the vertices of the polytope. The lattice points in the polytope are exactly the vectors $v \in \mathbb{Z}^n$ such that $[v,1] \in \bar{S}_{\mathbb{Z}^{n+1}}$, where S is generated by $[v_1,1],\dots,[v_m,1] \in \mathbb{Z}^{n+1}$. This is how to reduce (2) to (0).

In order to solve (3), one starts as in (2). In fact this realizes the multiplication of the generators of I by an additional indeterminate. Furthermore, one adds the generators

$$[1,0,\dots,0,0],\dots,[0,\dots,0,1,0] \in \mathbb{Z}^{n+1}$$

representing the indeterminates $X_1,\dots,X_n$, and again arrives at (0).

Therefore only the following two problems have to be solved.

(G) Find an irreducible system of generators of $\bar{S}_L$.
(H) Calculate the Hilbert series (and the Hilbert polynomial) of $\bar{S}_L$.

In order to solve problem (G), one proceeds according to the following steps:

(G1) Perform a change of coordinates (if necessary), such that $L = \mathbb{Z}^n$ and $\dim(C) = n$.
(G2) Decompose
$$C = \Sigma_1 \cup \dots \cup \Sigma_t \tag{1}$$
into *simplicial cones* $\Sigma_j \subseteq \mathbb{R}^n$ (i.e. Σ_j is spanned by exactly n linearly independent vectors). This process is called *triangulation*.
(G3) Solve problem (G) for each Σ_j.
(G4) Collect and reduce the generators found in step (G3).

Steps (G2)–(G4) will be discussed in the following three sections, and problem (H) is dealt with in Section 2.5. (You can immediately proceed to Section 2.5 if you are especially interested in problem (H).)

2.2 Constructing the triangulation

The triangulation is constructed inductively. The inductive step can be carried out as follows. Assume that $v_1,\dots,v_s \in \mathbb{Z}^n$ span a cone C_0 with decomposition

$$C_0 = \Sigma_1 \cup \dots \cup \Sigma_r,$$

and $v_1,\dots,v_s,v_{s+1} \in \mathbb{Z}^n$ span a cone C_1. In order to decompose C_1, we proceed in four steps:

(a) Find all support hyperplanes $h_1,\dots,h_k$ of C_0 for which v_{s+1} lies in the negative halfspace. (Actually, the computations of the support hyperplanes and the decomposition of C_0 are carried out simultaneously. For details of the former we refer the reader to [Bu].)

(b) Find all pairs (h_i, Σ_j) such that h_i is a support hyperplane of Σ_j.

(c) For each pair (h_i, Σ_j) determined in (b), build a new simplicial cone from the vectors v_{s+1} and $h_i \cap \Sigma_j \cap \{v_1, \dots, v_s\}$. (Note that the last set will consist of exactly $n-1$ vectors.)

(d) All the simplicial cones found in (c), together with $\Sigma_1, \dots, \Sigma_r$, will provide a decomposition of C_1.

For purposes of efficiency, especially when the Hilbert series is to be computed (and the `-h` option is active), we subdivide the triangulation into *blocks* and *subblocks*. Every block consists of exactly those simplicial cones found in step (c) above. That is, whenever a new vector v_{s+1} has to be integrated, a new block starts and collects all simplicial cones which are constructed due to v_{s+1} (and which in particular contain v_{s+1}). v_{s+1} is then called the *characteristic vector* of the block. A similar, but slightly more difficult operation leads to subblocks within the blocks.

2.3 The simplicial case

Once having computed a complete triangulation as described above, one is faced with the simplicial case. Therefore let $v_1, \dots, v_n \in \mathbb{Z}^n$ span a simplicial cone

$$\Sigma = \sum_{i=1}^{n} \mathbb{R}_{\geq 0} \cdot v_i \subseteq \mathbb{R}^n .$$

How can one calculate generators of the semigroup

$$G := \Sigma \cap \mathbb{Z}^n \,?$$

Clearly, $v_1, \dots, v_n$ and the set

$$B := \{\alpha_1 v_1 + \cdots + \alpha_n v_n \in \mathbb{Z}^n \mid \alpha_i \in \mathbb{Q}, 0 \leq \alpha_i < 1\} \tag{2}$$

generate the semigroup G. Therefore the main step is to explicitly determine set B.

To do this, define the lattice

$$\Lambda := \mathbb{Z}v_1 + \cdots + \mathbb{Z}v_n , \tag{3}$$

and observe that there is a bijection

$$B \overset{1:1}{\longleftrightarrow} \mathbb{Z}^n/\Lambda : \quad b \mapsto b + \Lambda ,$$

so that we have to find those representatives of the residue classes that lie in B. This is achieved as follows.

By elementary row and column operations, one finds matrices $T_r, T_c \in \mathrm{GL}_n(\mathbb{Z})$ such that

$$T_r \cdot (v_1, \dots, v_n) \cdot T_c = \mathrm{diag}(d_1, \dots, d_n)$$

is a diagonal matrix with strictly positive entries $d_i > 0$. Now Λ is spanned by the columns of $(v_1, \dots, v_n) \cdot T_c$, i.e. by the columns of $T_r^{-1} \cdot \mathrm{diag}(d_1, \dots, d_n)$.

Let $T_r^{-1} = (w_1, \dots, w_n)$. This implies

$$\Lambda = \mathbb{Z}d_1 w_1 + \cdots + \mathbb{Z}d_n w_n ,$$

hence

$$B = \{\alpha_i w_i \mod \Lambda \mid i = 1, \dots, n, \, \alpha_i = 0, \dots, d_i - 1\} .$$

2.4 Collecting generators

For a decomposition (1) of the original cone C, step (G3) yields a set U_j of generators of $\Sigma_j \cap \mathbb{Z}^n$ for every Σ_j. If we put

$$U = \bigcup_{j=1}^{t} U_j \,,$$

then the elements of U obviously generate $C \cap \mathbb{Z}^n$. Now it remains to construct a subset $V \subseteq U$ whose elements are irreducible *and* still generate $C \cap \mathbb{Z}^n$.

Let $U = \{u_1, \dots, u_N\}$. Set V is constructed inductively. In the 0-th step, we put $V = \emptyset$. In the k-th step, we check if $u_k - u \in C$ for some $u \in V$. If so, we forget u_k and increase k by 1. If not, we remove all those u from V which satisfy $u - u_k \in C$, and add u_k to V before increasing k by 1.

To test whether a vector $v \in \mathbb{Z}^n$ lies in C, one evaluates the support hyperplanes in v.

2.5 Computing the Hilbert series

Let us recall and extend the notation from Section 2.1, where we start with a finite set $E = \{w_1, \dots, w_m\} \subseteq \mathbb{Z}^n$. The integral closure of the affine semigroup $S = S(E) \subseteq \mathbb{Z}^n$ is denoted by $\bar{S} \subseteq \mathbb{Z}^n$. We may assume that the cone $C = \operatorname{cone}(E)$ satisfies $\dim(C) = n$.

Next we define the corresponding semigroup rings

$$R := K[X^v \mid v \in S] \quad \text{and} \quad \bar{R} := K[X^v \mid v \in \bar{S}] \,.$$

(Here, of course, K is a field, and X is the n-tuple $(X_1, \dots, X_n)$ of indeterminates.) By Remark 1 (ii), $\bar{R}$ is a finite R-module.

Of special interest is the *homogeneous* situation, i.e. there is $\varphi \in \operatorname{Hom}(\mathbb{Z}^n, \mathbb{Z})$ such that $\varphi(w_i) = 1$ for all i. Then there is a natural grading of $\bar{R}$, given by

$$\deg X^v := \varphi(v) \,,$$

and so R is generated in degree 1.

*Throughout this section, we will assume that **R** is homogeneous.* Then, according to the grading, write

$$\bar{R} = \bigoplus_{k=0}^{\infty} \bar{R}_k \,.$$

(Once more we refer the reader to [BH].) As is generally known, the *Hilbert function* of $\bar{R}$ is defined by

$$H(\bar{R}, k) = \dim_K(\bar{R}_k) \quad \text{for } k \geq 0$$

and coincides with the *Hilbert polynomial* $P_{\bar{R}}$ for large values of k:

$$H(\bar{R}, k) = P_{\bar{R}}(k) \quad \text{for } k \gg 0 \,.$$

The *Hilbert series* of $\bar{R}$ is

$$H_{\bar{R}}(t) = \sum_{k=0}^{\infty} H(\bar{R}, k) t^k \,.$$

and can be written as

$$H_{\bar{R}}(t) = \frac{h_0 + h_1 t + \cdots + h_{n-1} t^{n-1}}{(1-t)^n},$$

where $(h_0, h_1, \ldots, h_{n-1})$ is the ***h**-vector* of $\bar{R}$. In particular,

$$H(\bar{R}, k) = P_{\bar{R}}(k) \quad \text{for all } k \geq 0.$$

Before discussing the general case, one should investigate the simplicial case.

2.5.1 The simplicial case

Let $v_1, \ldots, v_n \in \mathbb{Z}^n$ span a simplicial cone $\Sigma \subseteq \mathbb{R}^n$, and define the semigroup $G := \Sigma \cap \mathbb{Z}^n$. Then the corresponding semigroup rings become

$$R = K[X^{v_1}, \ldots, X^{v_n}] \quad \text{and} \quad \bar{R} = K[X^v \,|\, v \in G].$$

An important fact is given by the following

Remark 2: *$\bar{R}$ is free over R with basis $X^B := \{X^b \,|\, b \in B\}$, where B is defined as in* (2).

Proof: Clearly, X^B generates $\bar{R}$ over R. Now assume that $B = \{b_1, \ldots, b_N\}$ and

$$r_1 X^{b_1} + \cdots + r_N X^{b_N} = 0 \quad \text{with } r_i \in R.$$

As it suffices to look at monomial components, we may assume that

$$r_i = \text{monomial} = k_i X^{s_{i1} v_1} \cdots X^{s_{in} v_n} = k_i X^{s_{i1} v_1 + \cdots + s_{in} v_n}$$

and

$$b_i + s_{i1} v_1 + \cdots + s_{in} v_n = b_j + s_{j1} v_1 + \cdots + s_{jn} v_n$$

for all i, j. But this immediately implies $b_i - b_j \in \Lambda$ (with Λ from (3)), hence $b_i = b_j$. □

Remark 2 has an immediate consequence.

Corollary 3: *We have*

$$H_{\bar{R}}(t) = \frac{h_0 + h_1 t + \cdots + h_{n-1} t^{n-1}}{(1-t)^n}$$

with the h-vector given by $h_i = \#\{b \in B \,|\, \deg X^b = i\}$.

Proof: The remark yields

$$\bar{R} = \bigoplus_{b \in B} R X^b,$$

hence

$$H_{\bar{R}} = \sum_{b \in B} H_{RX^b}.$$

Recalling that

$$H_{RX^b}(t) = \frac{t^{\deg X^b}}{(1-t)^n}$$

finishes the proof. □

2.5.2 The general case

In order to handle the general case, a combinatorial approach to the Hilbert function and series is helpful.

For a subset $\emptyset \neq T \subseteq C = \mathrm{cone}(E)$, define the Hilbert function of T by

$$H(T,k) := \#\{v \in T \cap \mathbb{Z}^n \,|\, \varphi(v) = k\} \quad \text{for } k \geq 0,$$

and (formally) the Hilbert series of T by

$$H_T(t) = \sum_{k=0}^{\infty} H(T,k)t^k.$$

For $T = \emptyset$, we set $H_\emptyset(t) = H(\emptyset,0) = 1$.

Next we claim that this definition makes sense, i.e. $H(T,k) < \infty$ for all $k \geq 0$ and $T \subseteq C$. The proof is quite simple: It suffices to consider $H(C,k)$. Now if $w \in C$ satisfies $\varphi(w) = k$, then there is a representation

$$w = \sum_{v \in E} \alpha_v \cdot v$$

with $\alpha_v \geq 0$ and $k = \varphi(w) = \sum_{v\in E} \alpha_v$. Hence every α_v is bounded, and so is every coordinate of w. Altogether we have shown that there is only a finite number of possibilities for the choice of a vector $w \in C$ satisfying $\varphi(w) = k$ and $w \in \mathbb{Z}^n$.

The following remark shows that the Hilbert series H_T of a subset $T \subseteq C$ generalizes the Hilbert series $H_{\bar{R}}$ of $\bar{R}$.

Remark 4: *We have $H(C,k) = H(\bar{R},k)$ for all $k \geq 0$.*

Proof: Simply write

$$\bar{R}_k = \bigoplus_{\substack{v \in \bar{S} \\ \varphi(v)=k}} K \cdot X^v.$$

Therefore

$$H(\bar{R},k) = \#\{v \in \bar{S} \,|\, \varphi(v) = k\} = H(C,k)$$

by Remark 1 (i). □

In particular, the Hilbert series of any convex (in particular: simplicial) subcone $C' \subseteq C$ coincides, of course, with that of the corresponding semigroup ring $K[X^v \,|\, v \in C' \cap \mathbb{Z}^n]$.

Now there is an especially interesting connection between the Hilbert series of subsets of C. We only need the simplicial version.

Lemma 5: *Let $\Sigma \subseteq C$ be the simplicial cone spanned by the vectors $v_1, \ldots, v_n \in \mathbb{Z}^n$. Then*

$$H_\Sigma = \sum_{\sigma \subseteq \{v_1,\ldots,v_n\}} H_{\mathrm{int}(\mathrm{cone}(\sigma))},$$

where $\mathrm{cone}(\sigma) = \mathbb{R}_{\geq 0} \cdot \sigma$ *is the (possibly lower-dimensional) cone spanned by the vectors from σ, and* $\mathrm{int}(\mathrm{cone}(\sigma))$ *is its interior (with respect to the standard topology of $\mathbb{R}^n$).*

Proof: If $v \in \Sigma \cap \mathbb{Z}^n$ has a representation

$$v = \sum_{i \in I} \alpha_i v_i$$

with $\alpha_i > 0$ for all $i \in I$, then

$$v \in \operatorname{int}(\operatorname{cone}\{v_i \mid i \in I\}),$$

and vice versa. The zero vector is also counted correctly due to the convention $H_\emptyset = 1$. $\square$

Finally, we are now able to discuss the general case. For this, one uses the decomposition

$$C = \Sigma_1 \cup \cdots \cup \Sigma_t \tag{1}$$

of C into simplicial subcones Σ_j found in step (G2). One then proceeds by induction. The case $t = 1$ is clear from Section 2.5.1 and Lemma 5. The inductive step can be derived from Lemma 5 (and its proof) and is stated in the following

Recursion formula: *Suppose that*

$$C_1 = C_0 \cup \Sigma,$$

where $\Sigma \subseteq C$ is a simplicial cone spanned by the vectors $v_1, \ldots, v_n \in \mathbb{Z}^n$ (satisfying $\varphi(v_i) = 1$ for $i = 1, \ldots, n$), and $C_0, C_1 \subseteq C$ are finite unions of simplicial cones. Then, with the notation introduced above,

$$H_{C_1} = H_{C_0} + \sum_{\substack{\sigma \subseteq \{v_1, \ldots, v_n\} \\ \sigma \not\subseteq \text{simplicial subcone of } C_0}} H_{\operatorname{int}(\operatorname{cone}(\sigma))}. \tag{4}$$

This is the formula we use in our implementation. However, we do not calculate $H_{\operatorname{int}(\operatorname{cone}(\sigma))}$ directly from the definition, but use the representation

$$H_{\operatorname{int}(\operatorname{cone}(\sigma))}(t) = \frac{h'_1 t + \cdots + h'_{|\sigma|} t^{|\sigma|}}{(1-t)^{|\sigma|}},$$

where $|\sigma| := \#\sigma \geq 1$ and

$$h'_i := \#\{b \in B \cap \operatorname{cone}(\sigma) \mid \varphi(b) = |\sigma| - i\}$$

for $1 \leq i \leq |\sigma|$ (and B is as in (2)).

The validity of this approach can be seen from the proofs of Remark 2 and Corollary 3: One shows that $K[X^v \mid v \in \operatorname{int}(\operatorname{cone}(\sigma)) \cap \mathbb{Z}^n]$ is free over $K[X^v \mid v \in \sigma]$ with basis $X^{B'}$. Here

$$B' := \left\{ \sum_{v_i \in \sigma} \alpha_i v_i \in \mathbb{Z}^n \mid \alpha_i \in \mathbb{Q}, 0 < \alpha_i \leq 1 \right\}$$

is in a one-to-one correspondence with B_σ via the bijection

$$\psi : B_\sigma \to B' : v \mapsto \sum_{v_i \in \sigma} v_i - v.$$

Hence h_i' must equal

$$\#\{b \in B' \mid \varphi(b) = i\} = \#\{b \in B_\sigma \mid \varphi(b) = |\sigma| - i\}\,.$$

Some final remarks: (1) The subdivision of the triangulation into blocks and subblocks (see Section 2.2) allows us to simplify the summation in (4): σ must contain the characteristic vector of the current (sub-)block, and it must not be a face of any (already accounted) simplex of this (sub-)block. This reduces the investigation to the current subblock and speeds up the computation considerably, by a factor of at least 3. In order to check whether σ has to be accounted, we also use bit operations now. This reduces the computation times by another factor of 5, resulting in a considerably better performance of the program, compared to earlier versions.

(2) We only have restricted the computation of the Hilbert series to the homogeneous case because it is considerably easier to implement than the general case, in which the denominator of the Hilbert series has the form $\prod_{i=1}^{n}(1-t^{a_i})$, and the a_i vary along with the simplicial subcones.

References

[BH] W. Bruns, J. Herzog: *Cohen-Macaulay rings.* Cambridge University Press, Cambridge 1993.

[Bu] E. Burger: *Über homogene lineare Ungleichungssysteme.* Z. angew. Math. Mech. **36** (1956), 135–139.

Universität Osnabrück
Fachbereich Mathematik/Informatik
D–49069 Osnabrück
Germany

<wbruns@uos.de>