NmzIntegrate 1.0

Winfried Bruns and Christof Söger

wbruns@uos.de
csoeger@uos.de

1 The objectives of NmzIntegrate

We assume in the following that the reader is familiar with Normaliz, in particular with its treatment of Ehrhart series and quasipolynomials. NmzIntegrate requires Normaliz 2.9 or higher.

Normaliz computes certain data for a monoid

$$M = C \cap L$$

where $C \subset \mathbb{R}^n$ is a rational, polyhedral and pointed cone, and $L \subset \mathbb{Z}^n$ is a sublattice. These data are defined by the input to Normaliz. NmzIntegrate requires that M has been endowed with a grading deg (see the manual of Normaliz 2.9).

For such graded monoids Normaliz can compute the volume of the rational polytope

$$P = \{x \in \mathbb{R}_+ M : \deg x = 1\},$$

the Ehrhart series of P, and the quasipolynomial representing the Ehrhart function. (Here $\mathbb{R}_+ M$ is the cone generated by the elements of M; it may be smaller than C if L has rank $< n$.)

These computations can be understood as integrals of the constant polynomial $f = 1$, namely with respect to the counting measure defined by L for the Ehrhart function, and with respect to the (suitably normed) Lebesgue measure for the volume. NmzIntegrate generalizes these computations to arbitrary polynomials f in n variables with rational coefficients. (Mathematically, there is no need to restrict oneself to rational coefficients for f.)

More precisely, set

$$E(f,k) = \sum_{x \in M, \deg x = k} f(x),$$

and call $E(f,\_)$ the *generalized Ehrhart function* for f. (With $f=1$ we simply count lattice points.) The *generalized Ehrhart series* is the ordinary generating function

$$E_f(t)=\sum_{k=0}^{\infty}E(f,k)t^k.$$

It turns out that $E_f(t)$ is the power series expansion of a rational function at the origin, and can always be written in the form

$$E_f(t)=\frac{Q(t)}{(1-t^{\ell})^{\operatorname{totdeg} f+\operatorname{rank} M}},\qquad Q(t)\in\mathbb{Q}[t],\ \deg Q<\operatorname{totdeg} f+\operatorname{rank} M.$$

Here $\operatorname{totdeg} f$ is the total degree of the polynomial f, and ℓ is the least common multiple of the degrees of the extreme integral generators of M. See [2] for an elementary account and the algorithm used by NmzIntegrate.

It follows from the general theory of rational generating functions that there exists a quasipolynomial $q(k)$ with rational coefficients and of degree $\le \operatorname{totdeg} f+\operatorname{rank} M-1$ that evaluates to $E(f,k)$ for all $k\ge 0$. A quasipolynomial is a "polynomial" with periodic coefficients: there exists a *period* $\pi\in\mathbb{N}$ and true polynomials $q^j\in\mathbb{Q}[X]$, $j=0,\dots,\pi-1$, such that

$$q(k)=q^{(j)}(k)\qquad\text{if}\quad k\equiv j\quad(\pi).$$

Each of the polynomials $q^{(j)}$ is given as

$$q^{(j)}(k)=q_0^{(j)}+q_1^{(j)}X+\dots+q_{\operatorname{totdeg} f+\operatorname{rank} M-1}^{(j)}X^{\operatorname{totdeg} f+\operatorname{rank} M-1}$$

with constant coefficients in $\mathbb{Q}$. The period π divides ℓ.

Let $m=\operatorname{totdeg} f$ and f_m be the degree m homogeneous component of f. By letting k go to infinity and approximating f_m by a step function that is constant on the meshes of $\frac{1}{k}L$ (with respect to a fixed basis), one sees

$$q_{\operatorname{totdeg} f+\operatorname{rank} M-1}^{(j)}=\int_P f_m\,d\lambda$$

where $d\lambda$ is the Lebesgue measure that takes value 1 on a basic mesh of $L\cap\mathbb{R}M$ in the hyperplane of degree 1 elements in $\mathbb{R}M$. In particular, the *virtual leading coefficient* $q_{\operatorname{totdeg} f+\operatorname{rank} M-1}^{(j)}$ is constant and depends only on f_m. If the integral vanishes, the quasipolynomial q has smaller degree, and the true leading coefficient need not be constant. Following the terminology of commutative algebra and algebraic geometry, we call

$$(\operatorname{totdeg} f+\operatorname{rank} M-1)!\cdot q_{\operatorname{totdeg} f+\operatorname{rank} M-1}$$

the *virtual multiplicity* of M and f. It is an integer if f has integral coefficients and P is a lattice polytope.

NmzIntegrate computes

- (ES) the generalized Ehrhart series and its quasipolynomial,
- (Int) the Lebesgue integral of f over P, or
- (LC) the virtual leading coefficient and the virtual multiplicity.

The user controls the type of computation by a command line option. (ES) contains (LC), and (LC) is just the evaluation of (Int) on the highest homogeneous component of f. It is presently not possible to compute the Ehrhart series and the integral together if f is not homogeneous.

Acknowledgement. We gratefully acknowledge the support we received from John Abbott and Anna Bigatti in using CoCoALib, on which the multivariate polynomial algebra in NmzIntegrate is based.

2 Input files

2.1 Files produced by Normaliz

As usual, Normaliz starts from the file `<project>.in`. One runs Normaliz with the option

`-T` (or `-y`) for (Int) and (LC),
`-y` for (ES).

(It is allowed to combine `-T` and `-y`.)

This will produce the files with the following suffixes (in addition to `<project>.out` and possibly further output files determined by the Normaliz options `-f` and `-a`):

`-T` `inv`, `tgn`, `tri`
`-y` `inv`, `tgn`, `dec`.

NmzIntegrate reads

- the grading from `<project>.inv`,
- the rays of the triangulation from `<project>.tgn`,
- the triangulation from `<project>.tri` (for (Int) and (LC)) and
- the Stanley decomposition from `<project>.dec` (for (ES)).

If `<project>.tri` does not exist for one of the tasks (Int) or (LC), NmzIntegrate checks for the existence of `<project>.dec` and reads the triangulation from it.

NmzIntegrate does not read other files, neither `<project>.in` nor any other output file of Normaliz.

2.2 The polynomial

The polynomial is read from the file `<project>.pnm`. The input format is defined by the following rules:

1. The polynomial is a product of factors.
2. The factors are separated by the character $*$.
3. A factor is a sum of terms.
4. A term is a product of a rational number and a monomial. The number 1 can of course be omitted.
5. A monomial is a (possibly empty) product of indeterminates `x[<i>]` or powers `x[<i>]^<j>` of indeterminates where `<i>` represents an index between 1 and n and `<j>` represents a nonnegative integer.
6. Spaces and line breaks act as separators: they are not allowed within numbers, indeterminates or powers of indeterminates.
7. The brackets `(` and `)` can be used for visual structuring. They have no mathematical meaning (so far), and act like spaces.

Note that the names of the variables are fixed: `x[1]`,...,`x[<n>]` where `<n>` represents the number n. Since spaces act as separators, `x [1]`, `x[ 1]` or `x[2]^ 3` are illegal (so far), as well as `1/ 2` or `1 /2`. Furthermore it should be noted that the character $*$ is only allowed between factors, but not within a factor, and in particular not in a term or a monomial.

An example:

```
1/120*(x[1]+x[2]^2)*(-2x[3]x[4])
```

is a well formed input polynomial. Spaces (or line breaks) can be inserted on one or both sides of the characters `*+-` and also between factors of a term. Like brackets they do not change the polynomial. Therefore

```
1/120 * x[1]+x[2]^2 * -2x[3]x[4]
```

represents the same polynomial.

3 Running NmzIntegrate

There are three ways to run NmzIntegrate:

1. direct call from the command line
2. call from Normaliz (see Normaliz manual)
3. from jNormaliz via Normaliz.

The shortest possible command to start NmzIntegrate is

```
nmzIntegrate <project>
```

This will run the default computation (ES) on the `<project>`. The full input syntax is

```
nmzIntegrate [-cEIL] [-x=<T>] <project>
```

where `-c` and `-x=<T>` have the same meaning as for Normaliz:

`-c` activates the verbose mode in which control information is written to the terminal,

`-x=<T>` limits the number of parallel threads to `<T>`.

The remaining options control the type of computation:

`-E` activates the computation (ES) (the default mode, can be omitted),

`-I` activates the computation (Int),

`-L` activates the computation (LC).

These three options can be accumulated. If at least two options are set, the computations are carried out according to the following rules:

- If `-E` is present, `-L` will be suppressed since its result is contained in that of `-E`.
- If `-I` is present, then it will be suppressed if one of `-E` or `L` is set and the polynomial is homogeneous since `-L` and `-I` are identical for homogeneous polynomials.

If two different computations are carried out, then their output will appear consecutively in the output file.

Note that NmzIntegrate may need much more memory than Normaliz, especially with a high number of parallel threads. This is due to the fact that it may have to cope with very long polynomials.

4 The output file

The output will be written to the file `<project>.intOut` (so that it can be clearly distinguished from the Normaliz output file).

NmzIntegrate factors the polynomial, and the factorization is written to the output file. For the computation (LC) the polynomial is first replaced by its leading form, and the output file then contains the factorization of the leading form.

The output file is self explanatory, but see the Normaliz documentation for the interpretation of the format in which the generalized Ehrhart series and the quasipolynomial are printed.

Please have a look at the files

`rationalES.intOut`, `rationalInt.intOut` and `rationalLC.intOut`.

They were all produced from the example file `rational.in` in the Normaliz distribution and the file `rational.pnm`, and `rational.intOut` was suitably renamed.

The directory `example` contains further input files suited for NmzIntegrate- Look out for files with the suffix `pnm`.

5 Distribution and installation

The basic package (source, documentation, examples) for NmzIntegrate is contained in the basic package of Normaliz that you can download from `http://www.math.uos.de/normaliz`

The installation is described in the Normaliz documentation.

Likewise the executable of NmzIntegrate is contained in the Normaliz executable package for your system.

Therefore NmzIntegrate does not need a separate installation.

6 Compilation

Before the compilation of NmzIntegrate you must compile Normaliz 2.9 and install CoCoLib 0.9951 [1] (not contained in the Normaliz distribution). NmzIntegrate will not compile with later versions of CoCoALib or earlier versions of Normaliz.

Important: **after** the configuration of CoCoALib , but **before** its compilation via `make` you must modify the file `configuration/autoconf.mk` in the following way: add the flag

`-DCoCoA_THREADSAFE_HACK`

to the definition of `CXXFLAGS_COMMON` (probably near line 24 of `configuration/autoconf.mk`).

Under Linux or Mac OS navigate to the directory `genEhrhart` and run `make`. You should move the executable `nmzIntegrate` to the directory that contains `normaliz`.

Depending on the location of CoCoALib, you may have to adjust the path leading to it in the `Makefile` in `genEhrhart`.

If you should want to compile NmzIntegrate under MS Windows, please contact the authors.

7 Copyright and how to cite

NmzIntegrate 1.0 is free software licensed under the GNU General Public License, version 3. You can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

It is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the program. If not, see http://www.gnu.org/licenses/.

Please refer to Normaliz in any publication for which NmzIntegrate 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`.

You can add a reference to [2] in order to indicate that NmzIntegrate has been used.

References

[1] J. Abbott and A. Bigatti, *CoCoALib*. A GPL C++ library for doing Computations in Commutative Algebra. Available from `http://cocoa.dima.unige.it/cocoalib/`

[2] W. Bruns and C. Söger, *Generalized Ehrhart series and Integration in Normaliz*. `arXiv:1211.5178`