Thm (Haase, Nill, Payne)

A lattice polytope of degree s is a Cayley polytope of length at least $d+1-\frac{1}{2}(s^2+19s-4)$

($\Leftrightarrow$ P is a Cayley polytopes of lattice polytopes in dim at most $\frac{1}{2}(s^2+19s-4)$)

Conj: The bound is linear.

Thm: There exist only finitely many h^*-polynomials of lattice polytopes of degree s with leading coefficient h_s^*.

This again follows from a volume bound:

P Cayley polytope of degree s and length at least $d-1-n$, then $\mathrm{vol}\, P \le n!\, n^n \underbrace{V(n, h_s^*)}^n$

$\downarrow$

volume bound for n-polytopes with h_s^* interior pts

15. Short Rational Generating Functions

→ want to find an efficient algorithm to compute rational generating functions

So far we have the following methods:

(1) unimodular cone $C := \mathrm{cone}(a_1, \dots, a_d)$
with a_i primitive

$$\rightsquigarrow \quad g_C(x) = \frac{1}{\prod (1 - x^{a_i})}$$

(2) simplicial cone $C = \mathrm{cone}(a_1, \dots, a_d)$

$$\rightsquigarrow \quad g_C(x) = \frac{\sum_{u \in \Pi(C) \cap \mathbb{Z}^d} x^u}{\prod (1 - x^{a_i})}$$

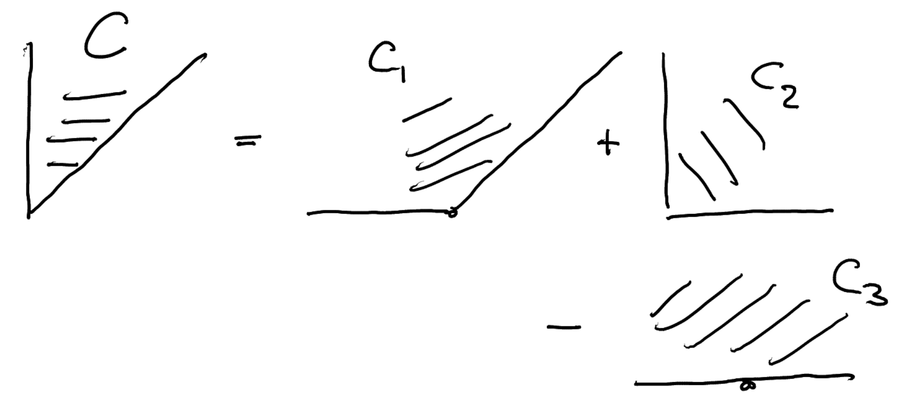
(3) general cone:
use triangulation and inclusion/exclusion or half open decomposition.

(4) Thm of Brion:
We can decompose generating series of cones or polytopes "modulo spaces with lines"

$$g_P(x) = \sum_{v \text{ vertex}} g_{T_P v}(x)$$

and, more generally, whenever there is a cone with lineality in a decomposition, then we need not consider it at the level of generating functions, as φ maps cones with lineality to 0

So e.g.



$$G_C(x) = G_{C_1}(x) + G_{C_2}(x) - G_{C_3}(x)$$

$$\rightsquigarrow g_C(x) = g_{C_1}(x) + g_{C_2}(x) \; (-\,0\,)$$

(5) We obtain the h^*-polynomial by specializing $g_{C(P)}(x)$ at $x = (t, \mathbb{1})$

We obtain $|P \cap \mathbb{Z}^d|$ by specializing $g_P(x)$ at $x = \mathbb{1}$

$\rightsquigarrow$ $x = \mathbb{1}$ is a removable pole of $g_P(x)$

$\rightsquigarrow$ need a method to efficiently resolve this

(series expansion is <u>not</u> efficient)

However, the approach via triangulation does not lead to a polynomial time algorithm
(where we mean polynomial in the size of the input)

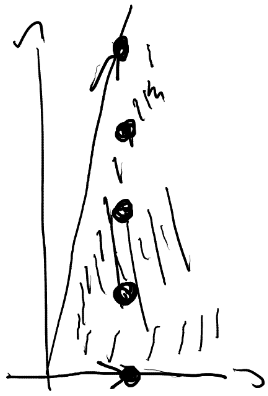
Ex: Consider

$$C = \operatorname{cone}\left(\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\m\end{pmatrix}\right)$$

$$\rightsquigarrow \Pi(C) \cap \mathbb{Z}^2 = \left\{\begin{pmatrix}1\\1\end{pmatrix}, \dots, \begin{pmatrix}1\\m-1\end{pmatrix}\right\}$$

So neither enumerating $\Pi \cap \mathbb{Z}^2$ to obtain the numerators in (2)
nor triangulating into unimodular cones as in (1)
is polynomial in the input size $\mathcal{O}(\text{size}(\text{input}))$

The key idea is now to realize that the generating series just record one monomial for every lattice point.



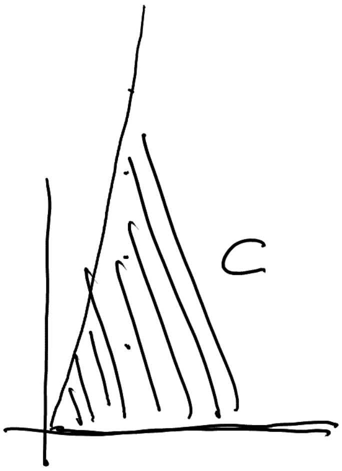
$\rightsquigarrow$ It is not necessary to use only positive decompositions by triangulating the cone.

We could also include monomials for points outside C to obtain a "nicer" cone, as long as we later subtract them

$\rightsquigarrow$ signed decompositions (Barvinok)

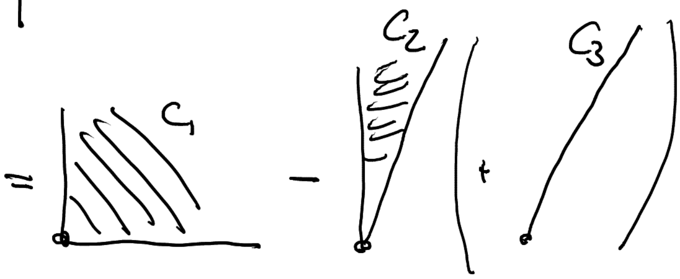
16.5

Here is an example:



With the given decompositions, both cones are unimodular!

$$\Rightarrow \quad g_C(x) = \frac{1}{(1-x)(1-y)}$$



$$- \frac{1}{(1-y)(1-xy^m)} \left(+ \frac{1}{(1-xy^m)} \right)$$

Compare with $\quad g_C(x) = \dfrac{1+xy+xy^2+xy^3+\ldots+xy^{m-1}}{(1-x)(1-xy^m)}$

The two functions are the same theoretically:

$$\frac{1-xy^m}{(1-x)(1-y)(1-xy^m)} - \frac{1-x}{(1-x)(1-y)(1-xy^m)} + \frac{(1-x)(1-y)}{(1-x)(1-y)(1-xy^m)}$$

$$= \frac{(1+xy+\ldots xy^{m-1})(1-y)}{(1-x)(1-y)(1-xy^m)} = \frac{1+xy+\ldots+xy^{m-1}}{(1-x)(1-xy^m)}$$

but not algorithmically! Computing the numerator $1+xy+xy^2+\ldots+xy^{m-1}$ can not be done in polynomial time in the size of the input.

However, to use this approach efficiently we have to check that we can find such a signed decomposition into

(1) a polynomial number of cones in

(2) a polynomial number of decomposition steps.

Let C be a simplicial cone

$$C = \mathrm{cone}(a_1, \dots, a_d)$$

with primitive $a_1, \dots, a_d$.

We define the <u>index</u> of C as

$$\mathrm{ind}\, C := |\Pi(C) \cap \mathbb{Z}^d| = |\det(a_1, \dots, a_d)|$$
$$= \mathrm{vol}\, \Pi(a_1, \dots, a_d)$$

Then C unimodular $\Leftrightarrow$ $\mathrm{ind}\, C = 1$

<u>Note</u> (1) $\mathrm{ind}\, C \in \mathbb{Z}_{\geq 1}$

(2) F face of C $\Rightarrow$ $\mathrm{ind}\, F \leq \mathrm{ind}\, C$

We need to subdivide cones in our collection into cones of smaller index as long as we can still find cones of index greater than 1

16.7

We use the Theorem of Minkowski, which provides us with a lattice vector of a short length to construct a decomposition

→ Note that the proof of the Thm of Minkowski was not constructive

→ we deal with this problem in the next section

Let us consider

$$C = \mathrm{cone}(v_1, \dots, v_d)$$

$$K := \left\{ \frac{1}{\sqrt[d]{\mathrm{ind}\, C}} \sum \lambda_i v_i \;\middle|\; -1 \leq \lambda_i \leq 1 \right\}$$

→ K is a compact, convex, centrally symmetric convex body

$$\mathrm{vol}\, K = \frac{1}{\mathrm{ind}\, C} \cdot 2^d\, \mathrm{ind}\, C = 2^d$$

Hence, there is $w \in (K \cap \mathbb{Z}^d) \setminus \{0\}$

and $w = \sum \lambda_i v_i$ for $0 \leq |\lambda_i| \leq \frac{1}{\sqrt[d]{\mathrm{ind}\, C}}$

< 1 for $\mathrm{ind}\, C > 1$

By passing to $-w$ if necessary we can assume that w and $v_1, \dots, v_d$ lie on the same side of some hyperplane

We define for $1 \le j \le d$:

$$C_j := \mathrm{cone}(v_1, \dots, v_{j-1}, w, v_{j+1}, \dots, v_d)$$

Then

$$\begin{aligned}\mathrm{ind}\, C &= |\det(v_1, \dots, v_{j-1}, w, v_{j+1}, \dots, v_d)| \\ &= |\lambda_j|\, |\det(v_1, \dots, v_d)| \\ &\le \frac{1}{\sqrt[d]{\mathrm{ind}\, C}}\, \mathrm{ind}\, C = (\mathrm{ind}\, C)^{\frac{d-1}{d}}\end{aligned}$$

and this is strictly less than ind C if ind $C \ge 2$.

Now this subdivision may require "subtraction" of cones.

We define a sign function:

$$\varepsilon_j := \begin{cases} 0 & \text{if } \dim C_j < d \\ 1 & \text{if } \det(v_1, \dots, v_d) \cdot \det(v_1, \dots, v_{j-1}, w, v_{j+1}, \dots, v_d) > 0 \\ -1 & \text{otherwise.} \end{cases}$$

We obtain:

$$G_C(x) = \sum_{j=1}^{d} \varepsilon_j\, G_{C_j}(x) + \text{lower dimensional contributions for inclusion-exclusion.}$$

In this decomposition we get at most

- d full-dimensional cones
- $2^d d$ cones in total

16.9

We need to check how many iterations we need to get the index of all cones below 2:

After n iterations a cone D in our list has index

$$\operatorname{ind} D \leq (\operatorname{ind} C)^{\left(\frac{d-1}{d}\right)^n}$$

→ need to find smallest n s.th. RHS is less than 2

(and thus $\operatorname{ind} D = 1$)