

# PALP in SAGE

## Module `lattice_polytope`

Andrey Novoseltsev

Department of Mathematics  
University of Washington

January 26, 2007

# Outline

- 1 Introduction
  - Lattice and Reflexive Polytopes
  - PALP: A Package for Analyzing Lattice Polytopes
- 2 Module `lattice_polytope`
  - Functions Available for Lattice Polytopes
  - Data Handling and Interaction with PALP
  - Working with Sequences of Polytopes
  - PALP vs. `lattice_polytope`
- 3 Future Development

# Outline

- 1 Introduction
  - Lattice and Reflexive Polytopes
  - PALP: A Package for Analyzing Lattice Polytopes
- 2 Module `lattice_polytope`
  - Functions Available for Lattice Polytopes
  - Data Handling and Interaction with PALP
  - Working with Sequences of Polytopes
  - PALP vs. `lattice_polytope`
- 3 Future Development

# Outline

- 1 Introduction
  - Lattice and Reflexive Polytopes
  - PALP: A Package for Analyzing Lattice Polytopes
- 2 Module `lattice_polytope`
  - Functions Available for Lattice Polytopes
  - Data Handling and Interaction with PALP
  - Working with Sequences of Polytopes
  - PALP vs. `lattice_polytope`
- 3 Future Development

# Definitions

## Definition

A *lattice polytope*  $\Delta$  is the convex hull in  $\mathbb{R}^d$  of finitely many points  $v_1, \dots, v_n \in \mathbb{Z}^d$ .

## Definition

The *polar polytope* to  $\Delta$  is  $\Delta^\circ = \{x : \langle v, x \rangle \geq -1 \ \forall v \in \Delta\}$ .

## Definition

$\Delta$  is *reflexive* if  $\Delta^\circ$  is a lattice polytope.

Note:  $(\Delta^\circ)^\circ = \Delta$ . A necessary condition for  $\Delta$  to be reflexive: the origin is the only interior lattice point. It is sufficient for  $d=2$ .

# Definitions

## Definition

A *lattice polytope*  $\Delta$  is the convex hull in  $\mathbb{R}^d$  of finitely many points  $v_1, \dots, v_n \in \mathbb{Z}^d$ .

## Definition

The *polar polytope* to  $\Delta$  is  $\Delta^\circ = \{x : \langle v, x \rangle \geq -1 \ \forall v \in \Delta\}$ .

## Definition

$\Delta$  is *reflexive* if  $\Delta^\circ$  is a lattice polytope.

Note:  $(\Delta^\circ)^\circ = \Delta$ . A necessary condition for  $\Delta$  to be reflexive: the origin is the only interior lattice point. It is sufficient for  $d=2$ .

# Definitions

## Definition

A *lattice polytope*  $\Delta$  is the convex hull in  $\mathbb{R}^d$  of finitely many points  $v_1, \dots, v_n \in \mathbb{Z}^d$ .

## Definition

The *polar polytope* to  $\Delta$  is  $\Delta^\circ = \{x : \langle v, x \rangle \geq -1 \ \forall v \in \Delta\}$ .

## Definition

$\Delta$  is *reflexive* if  $\Delta^\circ$  is a lattice polytope.

Note:  $(\Delta^\circ)^\circ = \Delta$ . A necessary condition for  $\Delta$  to be reflexive: the origin is the only interior lattice point. It is sufficient for  $d=2$ .

# Definitions

## Definition

A *lattice polytope*  $\Delta$  is the convex hull in  $\mathbb{R}^d$  of finitely many points  $v_1, \dots, v_n \in \mathbb{Z}^d$ .

## Definition

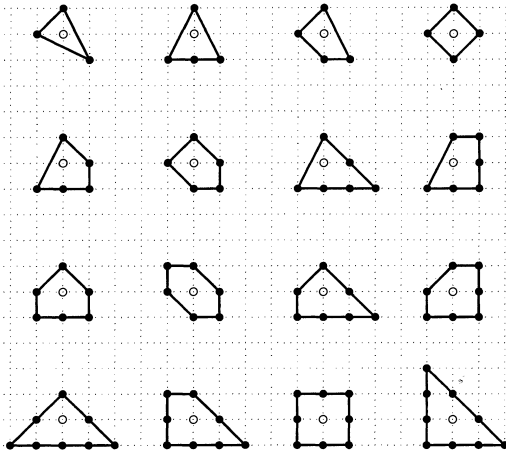
The *polar polytope* to  $\Delta$  is  $\Delta^\circ = \{x : \langle v, x \rangle \geq -1 \ \forall v \in \Delta\}$ .

## Definition

$\Delta$  is *reflexive* if  $\Delta^\circ$  is a lattice polytope.

Note:  $(\Delta^\circ)^\circ = \Delta$ . A necessary condition for  $\Delta$  to be reflexive: the origin is the only interior lattice point. It is sufficient for  $d=2$ .

# Low-Dimensional Reflexive Polytopes



Up to the action of  $GL(\mathbb{Z}^2)$  there are 16 reflexive polygons.

For  $d = 3$ : 4319.

For  $d = 4$ : 473,800,776.

The image is taken from Poonen, B. & Rodriguez-Villegas, F., Lattice polygons and the number 12, Amer. Math. Monthly 107 (2000), no. 3, 238–250.

# Toric Varieties

- A reflexive polytope  $\Delta$  generates a fan  $\mathcal{F}$  of rational polyhedral cones, which generates a toric variety  $\mathbb{P}_{\mathcal{F}}$ .
- Such toric varieties are used as ambient spaces for constructing Calabi-Yau manifolds as:
  - hypersurfaces;
  - complete intersections of hypersurfaces, corresponding to NEF-partitions of the vertex set.
- A triangulation of  $\partial\Delta$  with vertices at all lattice points generates a fan  $\mathcal{F}'$  and  $\mathbb{P}_{\mathcal{F}'}$  resolves singularities of  $\mathbb{P}_{\mathcal{F}}$ .
- Batyrev-Borisov mirror symmetry conjecture relates Calabi-Yau manifolds in toric varieties corresponding to a pair of reflexive polytopes  $\Delta$  and  $\Delta^\circ$ .

# Toric Varieties

- A reflexive polytope  $\Delta$  generates a fan  $\mathcal{F}$  of rational polyhedral cones, which generates a toric variety  $\mathbb{P}_{\mathcal{F}}$ .
- Such toric varieties are used as ambient spaces for constructing Calabi-Yau manifolds as:
  - hypersurfaces;
  - complete intersections of hypersurfaces, corresponding to NEF-partitions of the vertex set.
- A triangulation of  $\partial\Delta$  with vertices at all lattice points generates a fan  $\mathcal{F}'$  and  $\mathbb{P}_{\mathcal{F}'}$  resolves singularities of  $\mathbb{P}_{\mathcal{F}}$ .
- Batyrev-Borisov mirror symmetry conjecture relates Calabi-Yau manifolds in toric varieties corresponding to a pair of reflexive polytopes  $\Delta$  and  $\Delta^\circ$ .

# Toric Varieties

- A reflexive polytope  $\Delta$  generates a fan  $\mathcal{F}$  of rational polyhedral cones, which generates a toric variety  $\mathbb{P}_{\mathcal{F}}$ .
- Such toric varieties are used as ambient spaces for constructing Calabi-Yau manifolds as:
  - hypersurfaces;
  - complete intersections of hypersurfaces, corresponding to NEF-partitions of the vertex set.
- A triangulation of  $\partial\Delta$  with vertices at all lattice points generates a fan  $\mathcal{F}'$  and  $\mathbb{P}_{\mathcal{F}'}$  resolves singularities of  $\mathbb{P}_{\mathcal{F}}$ .
- Batyrev-Borisov mirror symmetry conjecture relates Calabi-Yau manifolds in toric varieties corresponding to a pair of reflexive polytopes  $\Delta$  and  $\Delta^\circ$ .

# Toric Varieties

- A reflexive polytope  $\Delta$  generates a fan  $\mathcal{F}$  of rational polyhedral cones, which generates a toric variety  $\mathbb{P}_{\mathcal{F}}$ .
- Such toric varieties are used as ambient spaces for constructing Calabi-Yau manifolds as:
  - hypersurfaces;
  - complete intersections of hypersurfaces, corresponding to NEF-partitions of the vertex set.
- A triangulation of  $\partial\Delta$  with vertices at all lattice points generates a fan  $\mathcal{F}'$  and  $\mathbb{P}_{\mathcal{F}'}$  resolves singularities of  $\mathbb{P}_{\mathcal{F}}$ .
- Batyrev-Borisov mirror symmetry conjecture relates Calabi-Yau manifolds in toric varieties corresponding to a pair of reflexive polytopes  $\Delta$  and  $\Delta^\circ$ .

# What is PALP?

- A **P**ackage for **A**nalyzing **L**attice **P**olytopes was created by Maximilian Kreuzer and Harald Skarke. It includes `nef.x`, which was created by Erwin Riegler.
- They used it to classify reflexive polytopes in 4 dimensions and other tasks.
- It is a small packet (its size is less than 1MB) of optimized C-programs for processing lists of lattice polytopes.
- The same action is determined for all polytopes during the program call.
- Parameters such as the maximum number of vertices must be specified before compiling the program.

# What is PALP?

- A **P**ackage for **A**nalyzing **L**attice **P**olytopes was created by Maximilian Kreuzer and Harald Skarke. It includes `nef.x`, which was created by Erwin Riegler.
- They used it to classify reflexive polytopes in 4 dimensions and other tasks.
- It is a small packet (its size is less than 1MB) of optimized C-programs for processing lists of lattice polytopes.
- The same action is determined for all polytopes during the program call.
- Parameters such as the maximum number of vertices must be specified before compiling the program.

# What is PALP?

- A **P**ackage for **A**nalyzing **L**attice **P**olytopes was created by Maximilian Kreuzer and Harald Skarke. It includes `nef.x`, which was created by Erwin Riegler.
- They used it to classify reflexive polytopes in 4 dimensions and other tasks.
- It is a small packet (its size is less than 1MB) of optimized C-programs for processing lists of lattice polytopes.
- The same action is determined for all polytopes during the program call.
- Parameters such as the maximum number of vertices must be specified before compiling the program.

# What is PALP?

- A **P**ackage for **A**nalyzing **L**attice **P**olytopes was created by Maximilian Kreuzer and Harald Skarke. It includes `nef.x`, which was created by Erwin Riegler.
- They used it to classify reflexive polytopes in 4 dimensions and other tasks.
- It is a small packet (its size is less than 1MB) of optimized C-programs for processing lists of lattice polytopes.
- The same action is determined for all polytopes during the program call.
- Parameters such as the maximum number of vertices must be specified before compiling the program.

# Example of PALP output: `poly.x -h`

This is ``poly.x``: computing data of a polytope P

Usage: `poly.x [-<Option-string>] [in-file [out-file]]`

Options (concatenate any number of them into <Option-string>):

<code>h</code>	print this information		<code>n</code>	do not complete polytope or
<code>f</code>	use as filter			calculate Hodge numbers
<code>g</code>	general output:		<code>i</code>	incidence information
	P reflexive: numbers of (dual)		<code>s</code>	check for span property
	points/vertices, Hodge numbers			(only if P from CWS)
	P not reflexive: numbers of		<code>I</code>	check for IP property
	points, vertices, equations		<code>S</code>	number of symmetries
<code>p</code>	points of P		<code>T</code>	upper triangular form
<code>v</code>	vertices of P		<code>N</code>	normal form
<code>e</code>	equations of P/vertices of P-dual		<code>t</code>	traced normal form computation
<code>m</code>	pairing matrix between vertices		<code>V</code>	IP simplices among vertices of P*
	and equations		<code>P</code>	IP simplices among points of P*
<code>d</code>	points of P-dual			(with $1 \leq \text{codim} \leq \#$ when # is set)
	(only if P reflexive)		<code>Z</code>	lattice quotients for IP simplices
<code>a</code>	all of the above except h,f		<code>#</code>	$\#=1,2,3$ fibers spanned by IP
<code>l</code>	LG-‘Hodge numbers’ from single			simplices with $\text{codim} \leq \#$
	weight input		<code>##</code>	$\##=11,22,33,(12,23)$ : all (fibered)
<code>r</code>	ignore non-reflexive input			fibers with specified $\text{codim}(s)$
<code>D</code>	dual polytope as input (ref only)			when combined: $\### = (\##)\#$

Input: degrees and weights ‘`d1 w11 w12 ... d2 w21 w22 ...`’  
or ‘`d np`’ or ‘`np d`’ ( $d$ =Dimension,  $np$ =#[points]) and  
(after newline)  $np \cdot d$  coordinates

Output: as specified by options

# Example of PALP output: `poly.x -i`

```

Degrees and weights  `d1 w11 w12 ... d2 w21 w22 ...'
  or `#lines #columns' (= `PolyDim #Points' or `#Points PolyDim'):
2 4
Type the 8 coordinates as dim=2 lines with #pts=4 columns:
1 0 -1 0
0 1 0 -1
Incidences as binary numbers [F-vector=(4 4)]:
v[d][i]: sum_j Incidence(i'th dim-d-face, j-th vertex) x 2^j
v[0]: 1000 0001 0100 0010
v[1]: 1001 1100 0011 0110
f[d][i]: sum_j Incidence(i'th dim-d-face, j-th facet) x 2^j
f[0]: 0011 0101 1010 1100
f[1]: 0001 0010 0100 1000
Degrees and weights  `d1 w11 w12 ... d2 w21 w22 ...'
  or `#lines #columns' (= `PolyDim #Points' or `#Points PolyDim'):

```

# Construction of Lattice Polytopes

- Module `sage.geometry.lattice_polytope` is included into SAGE since the version 1.6 and uses PALP for some internal operations (it is not a “SAGE interface”).
- `LatticePolytope` function is available directly for constructing polytopes from matrices or files.
- The convex hull of given points may be computed during construction (one of the operations using PALP: calls "`poly.x -fv`").
- Given vertices (or points) must span the space.

# Construction of Lattice Polytopes

- Module `sage.geometry.lattice_polytope` is included into SAGE since the version 1.6 and uses PALP for some internal operations (it is not a “SAGE interface”).
- `LatticePolytope` function is available directly for constructing polytopes from matrices or files.
- The convex hull of given points may be computed during construction (one of the operations using PALP: calls "`poly.x -fv`").
- Given vertices (or points) must span the space.

## Member Functions

- For any polytope it is possible to compute:
  - vertices, points, facets, faces, and their numbers;
  - distances between facets and vertices;
  - check if the polytope is reflexive (by computing polar);
  - output of `poly.x` with any keys applied to the polytope.
- For any reflexive polytope:
  - polar polytope and its data (duality between faces of the polytope and its polar is preserved);
  - codimension two NEF-partitions and  $m_{i,f}$  vectors;
  - output of `nef.x` with any keys applied to the polytope.
- For any 3-dimensional polytope: 3D graph.
- For any face: boundary and interior points and their numbers.

## Member Functions

- For any polytope it is possible to compute:
  - vertices, points, facets, faces, and their numbers;
  - distances between facets and vertices;
  - check if the polytope is reflexive (by computing polar);
  - output of `poly.x` with any keys applied to the polytope.
- For any reflexive polytope:
  - polar polytope and its data (duality between faces of the polytope and its polar is preserved);
  - codimension two NEF-partitions and  $m_{i,f}$  vectors;
  - output of `nef.x` with any keys applied to the polytope.
- For any 3-dimensional polytope: 3D graph.
- For any face: boundary and interior points and their numbers.

## Member Functions

- For any polytope it is possible to compute:
  - vertices, points, facets, faces, and their numbers;
  - distances between facets and vertices;
  - check if the polytope is reflexive (by computing polar);
  - output of `poly.x` with any keys applied to the polytope.
- For any reflexive polytope:
  - polar polytope and its data (duality between faces of the polytope and its polar is preserved);
  - codimension two NEF-partitions and  $m_{i,f}$  vectors;
  - output of `nef.x` with any keys applied to the polytope.
- For any 3-dimensional polytope: 3D graph.
- For any face: boundary and interior points and their numbers.

## Member Functions

- For any polytope it is possible to compute:
  - vertices, points, facets, faces, and their numbers;
  - distances between facets and vertices;
  - check if the polytope is reflexive (by computing polar);
  - output of `poly.x` with any keys applied to the polytope.
- For any reflexive polytope:
  - polar polytope and its data (duality between faces of the polytope and its polar is preserved);
  - codimension two NEF-partitions and  $m_{i,f}$  vectors;
  - output of `nef.x` with any keys applied to the polytope.
- For any 3-dimensional polytope: 3D graph.
- For any face: boundary and interior points and their numbers.

## (Almost) All Data are Cached

- Member functions save computed values:

```
def points(self):  
    try:  
        return self._points  
    except AttributeError:  
        self._polytope._face_compute_points(self)  
        return self._points
```

- To prevent data corruption, returned values are immutable.
- Polytopes cannot be changed after construction. But they do accumulate computed data.
- “Equal”  $\equiv$  “the same memory block.”

## (Almost) All Data are Cached

- Member functions save computed values:

```
def points(self):  
    try:  
        return self._points  
    except AttributeError:  
        self._polytope._face_compute_points(self)  
        return self._points
```

- To prevent data corruption, returned values are immutable.
- Polytopes cannot be changed after construction. But they do accumulate computed data.
- “Equal”  $\equiv$  “the same memory block.”

## (Almost) All Data are Cached

- Member functions save computed values:

```
def points(self):  
    try:  
        return self._points  
    except AttributeError:  
        self._polytope._face_compute_points(self)  
        return self._points
```

- To prevent data corruption, returned values are immutable.
- Polytopes cannot be changed after construction. But they do accumulate computed data.
- “Equal”  $\equiv$  “the same memory block.”

## Pickling (`dump/load`)

- Allows caching between sessions.
- During pickling most lists and matrices are discarded. This saves space and **TIME**.
- Faces and the number of their (boundary) points are saved.
- Strange Python behaviour:

```
sage: time K3.dump("test")
CPU times: user 27.06 s, sys: 0.50 s, total: 27.56 s
sage: time K3 = load("test")
CPU times: user 23.89 s, sys: 0.46 s, total: 24.35 s
sage: time K3 = load("test")
CPU times: user 31.39 s, sys: 0.16 s, total: 31.55 s
sage: time K3 = load("test")
CPU times: user 35.88 s, sys: 0.11 s, total: 35.98 s
sage: time K3 = load("test")
CPU times: user 37.58 s, sys: 0.20 s, total: 37.78 s
```

## Pickling (`dump/load`)

- Allows caching between sessions.
- During pickling most lists and matrices are discarded. This saves space and **TIME**.
- Faces and the number of their (boundary) points are saved.
- Strange Python behaviour:

```
sage: time K3.dump("test")
CPU times: user 27.06 s, sys: 0.50 s, total: 27.56 s
sage: time K3 = load("test")
CPU times: user 23.89 s, sys: 0.46 s, total: 24.35 s
sage: time K3 = load("test")
CPU times: user 31.39 s, sys: 0.16 s, total: 31.55 s
sage: time K3 = load("test")
CPU times: user 35.88 s, sys: 0.11 s, total: 35.98 s
sage: time K3 = load("test")
CPU times: user 37.58 s, sys: 0.20 s, total: 37.78 s
```

## Pickling (`dump/load`)

- Allows caching between sessions.
- During pickling most lists and matrices are discarded. This saves space and **TIME**.
- Faces and the number of their (boundary) points are saved.
- Strange Python behaviour:

```
sage: time K3.dump("test")
CPU times: user 27.06 s, sys: 0.50 s, total: 27.56 s
sage: time K3 = load("test")
CPU times: user 23.89 s, sys: 0.46 s, total: 24.35 s
sage: time K3 = load("test")
CPU times: user 31.39 s, sys: 0.16 s, total: 31.55 s
sage: time K3 = load("test")
CPU times: user 35.88 s, sys: 0.11 s, total: 35.98 s
sage: time K3 = load("test")
CPU times: user 37.58 s, sys: 0.20 s, total: 37.78 s
```

# Using PALP

If the performed action requires data computed by PALP, then:

- a pipe to `poly.x/nef.x` with required keys is created;
- vertices of the polytope are used as an input after conversion to the text representation of PALP;
- the output is read and checked for errors:
  - a message in `stderr`,
  - empty output,
  - the exclamation mark in the output;
- correct output is converted to SAGE data types (matrices, lists, sequences, etc) .

# Improving Performance for Sequences of Polytopes

- All data required to compute the output value of any member function are computed automatically.
- Almost any first operation with a polytope will call a program from PALP.
- Processing all polytopes during one call is much more efficient (up to 500 times).
- Functions `lattice_polytope.all_*` compute data for many polytopes with a single call to PALP.
- SAGE gives immediate access to the source code of any function, so it is easy to find out what exactly should be precomputed.

# Improving Performance for Sequences of Polytopes

- All data required to compute the output value of any member function are computed automatically.
- Almost any first operation with a polytope will call a program from PALP.
- Processing all polytopes during one call is much more efficient (up to 500 times).
- Functions `lattice_polytope.all_*` compute data for many polytopes with a single call to PALP.
- SAGE gives immediate access to the source code of any function, so it is easy to find out what exactly should be precomputed.

# Improving Performance for Sequences of Polytopes

- All data required to compute the output value of any member function are computed automatically.
- Almost any first operation with a polytope will call a program from PALP.
- Processing all polytopes during one call is much more efficient (up to 500 times).
- Functions `lattice_polytope.all_*` compute data for many polytopes with a single call to PALP.
- SAGE gives immediate access to the source code of any function, so it is easy to find out what exactly should be precomputed.

# Example

## Looking for simplexes among 4319 reflexive polytopes of dimension 3:

```
sage: K3 = lattice_polytope.read_all_polytopes("K3vertices", "K3 %4d")
sage: K3[100]
K3 100: 3-dimensional, 5 vertices.
sage: time r = lattice_polytope.filter_polytopes(lambda p: p.nfacets() == 4, K3)
CPU times: user 14.54 s, sys: 57.97 s, total: 72.52 s
Wall time: 175.50
sage: len(r)
48
sage: time r = lattice_polytope.filter_polytopes(lambda p: p.nfacets() == 4, K3)
CPU times: user 0.09 s, sys: 0.01 s, total: 0.10 s
Wall time: 0.11
sage: len(r)
48
sage: K3 = lattice_polytope.read_all_polytopes("K3vertices", "K3 %4d")
sage: time lattice_polytope.all_faces(K3)
CPU times: user 12.86 s, sys: 0.23 s, total: 13.08 s
Wall time: 13.52
sage: time r = lattice_polytope.filter_polytopes(lambda p: p.nfacets() == 4, K3)
CPU times: user 0.07 s, sys: 0.00 s, total: 0.07 s
Wall time: 0.07
sage: len(r)
48
```

# PALP vs. `lattice_polytope`

PALP is a **fast** compiled package with many features that are not (yet) realized in `lattice_polytope` module. But:

- using `lattice_polytope` in SAGE is much more interactive and flexible, adding new functionality is as easy as using the module
- storing polytope data using SAGE data types gives convenient access to all other components of SAGE
- speed of interaction with PALP and internal computations can be increased by using SageX
- since PALP is a part of SAGE, it is always possible to use **any** features of PALP!

## PALP vs. `lattice_polytope`

PALP is a **fast** compiled package with many features that are not (yet) realized in `lattice_polytope` module. But:

- using `lattice_polytope` in SAGE is much more interactive and flexible, adding new functionality is as easy as using the module
- storing polytope data using SAGE data types gives convenient access to all other components of SAGE
- speed of interaction with PALP and internal computations can be increased by using SageX
- since PALP is a part of SAGE, it is always possible to use **any** features of PALP!

# PALP vs. `lattice_polytope`

PALP is a **fast** compiled package with many features that are not (yet) realized in `lattice_polytope` module. But:

- using `lattice_polytope` in SAGE is much more interactive and flexible, adding new functionality is as easy as using the module
- storing polytope data using SAGE data types gives convenient access to all other components of SAGE
- speed of interaction with PALP and internal computations can be increased by using SageX
- since PALP is a part of SAGE, it is always possible to use **any** features of PALP!

# PALP vs. `lattice_polytope`

PALP is a **fast** compiled package with many features that are not (yet) realized in `lattice_polytope` module. But:

- using `lattice_polytope` in SAGE is much more interactive and flexible, adding new functionality is as easy as using the module
- storing polytope data using SAGE data types gives convenient access to all other components of SAGE
- speed of interaction with PALP and internal computations can be increased by using SageX
- since PALP is a part of SAGE, it is always possible to use **any** features of PALP!

## PALP vs. `lattice_polytope`

PALP is a **fast** compiled package with many features that are not (yet) realized in `lattice_polytope` module. But:

- using `lattice_polytope` in SAGE is much more interactive and flexible, adding new functionality is as easy as using the module
- storing polytope data using SAGE data types gives convenient access to all other components of SAGE
- speed of interaction with PALP and internal computations can be increased by using SageX
- since PALP is a part of SAGE, it is always possible to use **any** features of PALP!

## Sample Applications

- For any reflexive polygon

$$l(\partial\Delta) + l(\partial\Delta^\circ) = 12.$$

(Four different proofs are discussed by Poonen, B. & Rodriguez-Villegas, F.)

- For any 3-dimensional reflexive polytope

$$\sum_{e \in E(\Delta)} (l^*(e) + 1)(l^*(\check{e}) + 1) = 24.$$

(Reference: Charles Doran. Checked by computation in MAGMA and SAGE.)

## Sample Applications

- For any reflexive polygon

$$l(\partial\Delta) + l(\partial\Delta^\circ) = 12.$$

(Four different proofs are discussed by Poonen, B. & Rodriguez-Villegas, F.)

- For any 3-dimensional reflexive polytope

$$\sum_{e \in E(\Delta)} (l^*(e) + 1)(l^*(\check{e}) + 1) = 24.$$

(Reference: Charles Doran. Checked by computation in MAGMA and SAGE.)

# Future Development

- Database of 4-dimensional reflexive polytopes (SQLite).
- Custom pickling: small size, high speed.
- Using SageX for computations.
- Using C-library interface to PALP instead of pipes (or keeping several open pipes).
- Support for other features of PALP (e.g. Hodge numbers).
- Interaction with Polymake.
- Automated adjusting of PALP compilation parameters.

# Future Development

- Database of 4-dimensional reflexive polytopes (SQLite).
- Custom pickling: small size, high speed.
- Using SageX for computations.
- Using C-library interface to PALP instead of pipes (or keeping several open pipes).
- Support for other features of PALP (e.g. Hodge numbers).
- Interaction with Polymake.
- Automated adjusting of PALP compilation parameters.

# Future Development

- Database of 4-dimensional reflexive polytopes (SQLite).
- Custom pickling: small size, high speed.
- Using SageX for computations.
- Using C-library interface to PALP instead of pipes (or keeping several open pipes).
- Support for other features of PALP (e.g. Hodge numbers).
- Interaction with Polymake.
- Automated adjusting of PALP compilation parameters.

# Future Development

- Database of 4-dimensional reflexive polytopes (SQLite).
- Custom pickling: small size, high speed.
- Using SageX for computations.
- Using C-library interface to PALP instead of pipes (or keeping several open pipes).
- Support for other features of PALP (e.g. Hodge numbers).
- Interaction with Polymake.
- Automated adjusting of PALP compilation parameters.

# Future Development

- Database of 4-dimensional reflexive polytopes (SQLite).
- Custom pickling: small size, high speed.
- Using SageX for computations.
- Using C-library interface to PALP instead of pipes (or keeping several open pipes).
- Support for other features of PALP (e.g. Hodge numbers).
- Interaction with Polymake.
- Automated adjusting of PALP compilation parameters.

# Future Development

- Database of 4-dimensional reflexive polytopes (SQLite).
- Custom pickling: small size, high speed.
- Using SageX for computations.
- Using C-library interface to PALP instead of pipes (or keeping several open pipes).
- Support for other features of PALP (e.g. Hodge numbers).
- Interaction with Polymake.
- Automated adjusting of PALP compilation parameters.

# Future Development

- Database of 4-dimensional reflexive polytopes (SQLite).
- Custom pickling: small size, high speed.
- Using SageX for computations.
- Using C-library interface to PALP instead of pipes (or keeping several open pipes).
- Support for other features of PALP (e.g. Hodge numbers).
- Interaction with Polymake.
- Automated adjusting of PALP compilation parameters.

# Summary

- PALP is included into standard distribution of SAGE.
- Module `lattice_polytope` provides convenient access to it (needs care for effective work with many polytopes).
- Further functionality should be added and it should be easy to do. Some optimization would be nice.