OSCAR: The Project

Michael Joswig

TU Berlin & MPI-MiS Leipzig

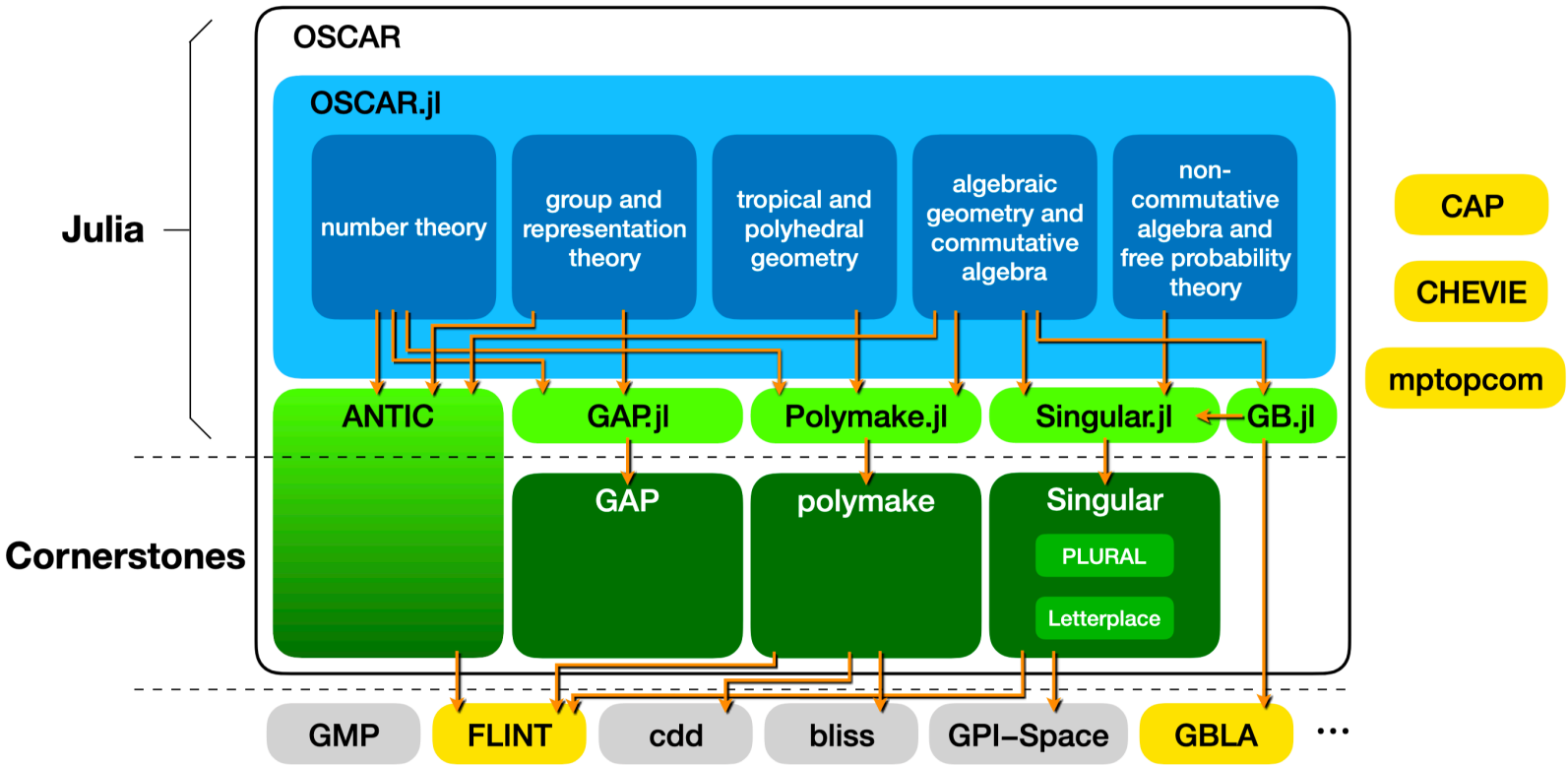
. . . and the OSCAR Development Team

ICERM, Feb 16, 2021

What is OSCAR?

http://oscar-system.org/

- joint software project of the CRC TRR 195, funded by DFG
  - written in Julia
  - planned duration: 2017–2028, three phases



- *beginning of 2nd phase:* fully functional interoperability layer in Julia

https://julialang.org/

- open Source (MIT License)
- JIT compilation: near C performance
- supports Linux, BSD, MacOS, Windows
- friendly C/Python-like (imperative) syntax
- easy/efficient C interoperability; good C++ support
- designed by mathematically minded people

Selected Julia Features

Julia is polymorphic:

```
gcd(a::Int, b::Int)
gcd(a::BigInt, b::BigInt)
gcd(a::Poly{T}, b::Poly{T}) where {T <: Field}
```

Julia supports multimethods:

```
*(a::Int, b::Matrix{Int})
*(a::Matrix{Int}, b::Int)
```

Julia supports triangular dispatch (template parameter chaining):

```
*(x::T, y::S) where {T <: QuotientRing,
                     S <: Poly{T}}
```

Example: Polytope From Group Orbit

```
G = symmetric_group(4)
x = [0,1,2,3]
M = Array(matrix(ZZ, [permuted(x,g) for g in G]))
P = convex_hull(M)
```

```
A Polyhedron of dimension 3
```

```
ambient_dim(P)
```

```
4
```

```
F = facets(P; as = :polyhedra)
n_vertices.(F)
```

```
[6, 6, 4, 6, 4, 4, 6, 4, 6, 6, 4, 6, 6, 4]
```

Example: Galois Groups

```
R, x = PolynomialRing(QQ, "x")
k, a = number_field(x^5-2)

G, C = galois_group(k)
roots(C, 1)
```

```
5-element Array{qadic,1}:
 583730*1048589^0 + O(1048589^1)
 (333313*1048589^0 + O(1048589^1))*a + 655516*1048589^0 + O(1048589^1)
 (715276*1048589^0 + O(1048589^1))*a + 576975*1048589^0 + O(1048589^1)
 (641808*1048589^0 + O(1048589^1))*a + 419477*1048589^0 + O(1048589^1)
 (406781*1048589^0 + O(1048589^1))*a + 910069*1048589^0 + O(1048589^1)
```

```
describe(G)
```

```
"C5 : C4"
```

Example: GIT-Fans with Symmetry

Let $\mathfrak{a}$ be polynomial ideal, homogeneous with respect to the grading given by (columns of) integer matrix Q. Consider the induced torus action on the affine variety $V(\mathfrak{a})$.

- Dolgachev and Hu (1998): **GIT-fan** classifies all possible quotients (of choices of open sets) in the sense of Mumford's geometric invariant theory in terms of a polyhedral fan.

Example

For $\mathbb{C}^* \times \mathbb{C}^2 \to \mathbb{C}^2, \quad t \cdot (x, y) = (tx, ty), \quad Q = (1, 1), \quad \mathfrak{a} = 0$

$$U_1 = \mathbb{C}^2$$
$$U_2 = \mathbb{C}^2 \setminus \{0\}$$

yielding as quotients a point and $\mathbb{P}^1$, respectively.

GIT-fan of affine cone over $\mathbb{G}(2,5)$ I

In the following, a = Pluecker ideal and Q = canonical grading matrix.

```
using GITFans
Q = [ ... ]; n = size(Q, 1)
Qt, T = PolynomialRing(QQ, :T => 1:n)
D = free_abelian_group(size(Q,2))
w = [D(Q[i, :]) for i = 1:n]
R = grade(Qt, w)

a = ideal(R, [
      T[5]*T[10] - T[6]*T[9] + T[7]*T[8],
      T[1]*T[9]  - T[2]*T[7] + T[4]*T[5],
      T[1]*T[8]  - T[2]*T[6] + T[3]*T[5],
      T[1]*T[10] - T[3]*T[7] + T[4]*T[6],
      T[2]*T[10] - T[3]*T[9] + T[4]*T[8],
]);
```

GIT-fan of affine cone over $\mathbb{G}(2,5)$ II

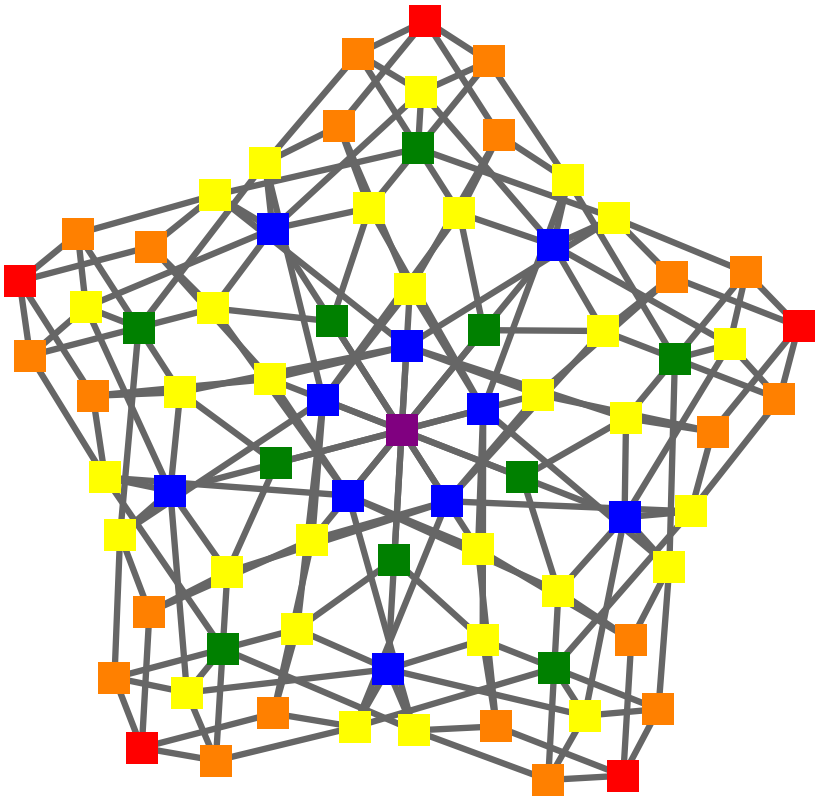
```
perms_list = [ [1,3,2,4,6,5,7,8,10,9],
               [5,7,1,6,9,2,8,4,10,3] ];
S10 = symmetric_group(n);
G, emb = sub([S10(x) for x in perms_list]...);

fanobj = GITFans.git_fan(a, Q, G);
fanobj.F_VECTOR
```

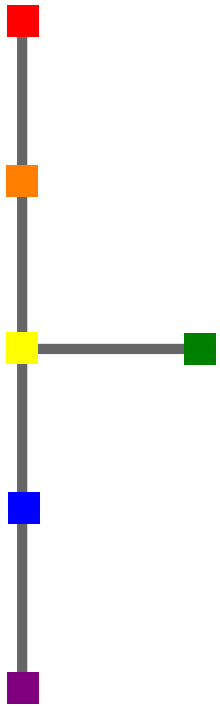
```
pm::Vector<pm::Integer>
20 110 240 225 76
```

J. Boehm, S. Keicher, Y. Ren. Computing GIT-fans with symmetry and the Mori chamber decomposition of $\overline{M}_{0,6}$. Math. Comp. (2020).

GIT-fan of affine cone over $\mathbb{G}(2,5)$ III

Adjacency graph of GIT-cones of maximal dimension / of their orbits.





Contributing to OSCAR

http://oscar-system.org/

Comments and Feature Requests

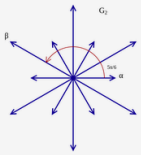
- join us on **Slack**
  - send email to webmaster-oscar@mathematik.uni-kl.de for an invitation
- consider subscribing to the `oscar-dev` mailing list

Contributing Code

- write your own `Julia` package and contact us (see above)
- fork on **GitHub** and submit pull request

Example: JuLie by Uli Thiel (version 0.1 of Feb 03, 2021)

Algebraic Lie theory, representation theory, and relevant combinatorics



JuLie Documentation

Search docs

About

- Using
- Motivation
- Developing
- Contributors

Combinatorics

Lie theory

Version v0.1

Now, you can start using the package as follows:

```
julia> using JuLie
julia> partitions(10)
```

You can get help for a function by putting a question mark in front, e.g.

```
julia> ?partitions
```

Motivation

Especially for combinatorics there's a lot already in other computer algebra systems and this justifies the question: why another package? I will give 3 (interwoven) reasons:

1. I want to create a package that covers the mathematics that I especially care about in a way that I think about it. One distant goal is to have all the material available from the book Introduction to Soergel bimodules with Elias, Makisumi, and Williamson. It will take a lot of time and I don't know if I succeed but it's one motivation.
2. I hope this package will eventually form one pillar of the OSCAR project.
3. What really convinced me of Julia as programming language—and thus of the whole enterprise—is its straightforward high-level syntax (like Python) paired with incredible performance (unlike Python). Have a look at the following examples creating the list (not an iterator) of all partitions of the integer 90 (there are ~56.6 million) in different computer algebra systems.

In Sage (v9.1):

```
sage: time X=Partitions(90).list()
Wall time: 3min 5s
#Uses 26.665GiB mem, quitting Sage takes quite a bit of time
```

https://ulthiel.github.io/JuLie.jl/stable/

Concluding Remarks

- OSCAR v0.5.1 (Feb 12, 2021)
  - Julia v1.5.3
- try this demo at home:
  - https://github.com/micjoswig/oscar-notebooks/blob/master/ICERM-210216/
  - Julia Package Manager: activate / instantiate
- contact us:
  - http://oscar-system.org
  - webmaster-oscar@mathematik.uni-kl.de