# Building Software Infrastructure for Research Mathematics

Antony Della Vecchia

TU Berlin

2024-06-20

# Overview

- MaRDI and the FAIR Principles
- `Julia` and `OSCAR`
- Serialization and Datasets

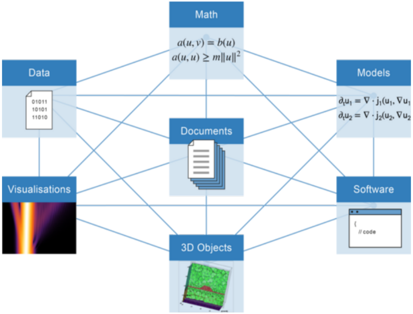
# The FAIR Guiding Principles for scientific data management and stewardship

M. Wilkinson et al. 2019

- Findable
- Accessible
- Interoperable
- Reusable

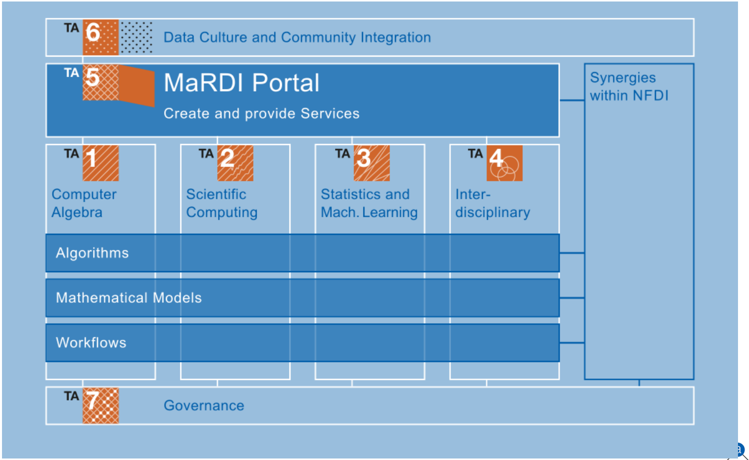
# Mathematics Research Data Initiative

- Develop a mathematical research data infrastructure.
- Set standards for confirmable workflows and certified mathematical research data.
- Provide services to both the mathematical and wider scientific community.




Find the MaRDI proposal here `https://zenodo.org/records/6552436`.

# MaRDI Task Area Breakdown



# Computer Algebra Task Area

## Principle Investigators

- Claus Fieker (RPTU Kaiserslautern)
- Michael Joswig (TU Berlin)

## On Going Projects

- Confirmable workflows (OSCAR Book) Lars Kastner
- Technical Peer Review (ANTS, LuCANT, MEGA) Jereon Hanselman
- Containerization and environments (MaPS) Aaruni Kaushik
- Serialization and Databases (`.mrdi` File Format)

# Some Key Features of `Julia`

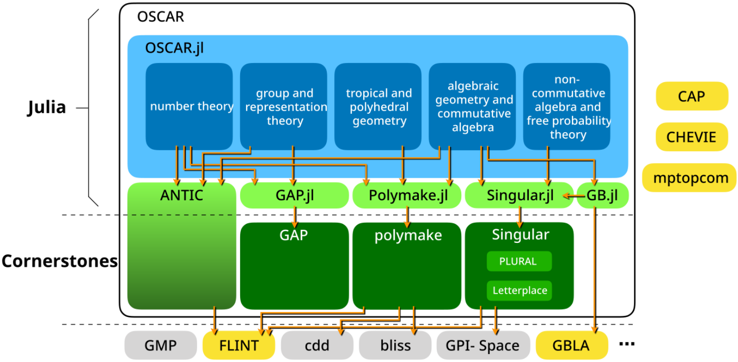
## Reproducibility

- Built-in package manager.
- Intuitive and user friendly environment functionality.
- Uses git hash when recreating environments.

## Interoperability

- Consistent tool for building and wrapping binaries. (Binary builder)
- Excellent Multiple dispatch functionality.

# OSCAR



OSCAR
OSCAR.jl
Julia
number theory
group and representation theory
tropical and polyhedral geometry
algebraic geometry and commutative algebra
non-commutative algebra and free probability theory
ANTIC
GAP.jl
Polymake.jl
Singular.jl
GB.jl
CAP
CHEVIE
mptopcom
Cornerstones
GAP
polymake
Singular
PLURAL
Letterplace
GMP
FLINT
cdd
bliss
GPI- Space
GBLA
...

# Serialization

## Why Files?

- People often have a preferred software system.

## Serialization

### Why Files?

- People often have a preferred software system.
- Computations can be expensive.

# Serialization

## Why Files?

- People often have a preferred software system.
- Computations can be expensive.
- Verification of results is at most as computationally expensive.

## Serialization

### Why Files?

- People often have a preferred software system.
- Computations can be expensive.
- Verification of results is at most as computationally expensive.
- Software changes often.

## Storing Mathematical Data

- It's common to have multiple perspectives on an object in mathematics.

## Storing Mathematical Data

- It's common to have multiple perspectives on an object in mathematics.
- While storing mathematical data a choice of perspective must me made.

## Storing Mathematical Data

- It's common to have multiple perspectives on an object in mathematics.
- While storing mathematical data a choice of perspective must me made.

Say we want to store:

$$p = 2y^3z^4 + (a+3)z^2 + 5ay + 1$$

# Storing Mathematical Data

- It's common to have multiple perspectives on an object in mathematics.
- While storing mathematical data a choice of perspective must me made.

Say we want to store:

$$p = 2y^3z^4 + (a+3)z^2 + 5ay + 1$$

- Some technicalities with the coefficients.
- Is x considered as a coefficient of y?

# History of File Formats

```
* Class:       LP
* Rows:        5
* Columns:     2
* Format:      MPS
*
Name          unnamed#0
ROWS
 N  C0000000
   ...
 G  R0000003
COLUMNS
    x1   C0000000  2        R0000000  1
    x1   R0000001  -1
    x2   C0000000  3        R0000002  1
    x2   R0000003  -1
RHS
    B    C0000000  -1       R0000000  -1
    B    R0000001  -1       R0000002  -1
    B    R0000003  -1
BOUNDS
 FR BND       x1
 FR BND       x2
ENDATA
```

- The LP file format and the MPS file format. IBM [1970s] (industry standards)
- `Mathematica` Notebooks. Wolfram Mathematica [1988]
- `OpenMath` (tree structure). Mike Dewar [2000]
- `IPython 0.12` Interactive Browser Notebooks (Jupyter) [2011]
- `polymake` File Format. E. Gawrilow, S. Hampe, and M. Joswig [2016]

# The Polymake File Format

```
<?xml version="1.0" encoding="utf-8"?>
<?pm chk="56e977e8"?>
<object name="square" type="polytope::Polytope&lt;Rational&gt;"
        version="3.0"
        xmlns="http://www.math.tu-berlin.de/polymake/#3">
  <description><![CDATA[cube of dimension 2]]></description>
  <property name="VERTICES">
    <m>
      <v>1 0 0</v>
      ...
    </m>
  </property>
  <property name="FACETS"
            type="SparseMatrix&lt;Rational,NonSymmetric&gt;">
    <m cols="3">
      <v> <e i="1">1</e> </v>
      ...
    </m>
  </property>
  <property name="LINEALITY_SPACE"><m /></property>
  <property name="BOUNDED" value="true" />
  <property name="N_FACETS" value="4" />
  <property name="N_VERTICES" value="4" />
  <property name="VOLUME" value="1/9" />
  <property name="TRIANGULATION">
    <object name="unnamed#0">
      <property name="FACETS">
        <m>
          <v>0 1 2</v>
          <v>1 2 3</v>
        </m>
      </property>
      <property name="F_VECTOR">
        <v>4 5 2</v>
      </property>
    </object>
  </property>
</object>
```

- First version (XML) published in 2016.

# The Polymake File Format


```
{
  "_attrs": {
    "FACETS": {
      "_type": "SparseMatrix<Rational, NonSymmetric>"
    }
  },
  "VERTICES": [["1", "0", "0"],
               ["1", "1/3", "0"],
               ["1", "0", "1/3"],
               ["1", "1/3", "1/3"]],
  "CONE_AMBIENT_DIM": 3,
  "N_VERTICES": 4,
  "_ns": {
    "polymake": [
      "https://polymake.org",
      "4.9"
    ]
  },
  "_info": {
    "description": "cube of dimension 2"
  },
  "TRIANGULATION": [{
      "FACETS": [[ 0, 1, 2],
                 [ 1, 2, 3]],
      "_id": "unnamed#0",
      "F_VECTOR": [4, 5, 2]
  }],
  "N_FACETS": 4,
  "_id": "square",
  "BOUNDED": true,
  "FACETS": [{ "1": "1" }, { "0": "1/3", "1": "-1"},
             { "2": "1" }, { "0": "1/3", "2": "-1" },
             {     "cols": 3    }],
  "_type": "polytope::Polytope<Rational>",
  "VOLUME": "1/9"
}
```


- First version (XML) published in 2016.
- Currently using JSON.

# The Polymake File Format


```
{
  "_attrs": {
    "FACETS": {
      "_type": "SparseMatrix<Rational, NonSymmetric>"
    }
  },
  "VERTICES": [["1", "0", "0"],
              ["1", "1/3", "0"],
              ["1", "0", "1/3"],
              ["1", "1/3", "1/3"]],
  "CONE_AMBIENT_DIM": 3,
  "N_VERTICES": 4,
  "_ns": {
    "polymake": [
      "https://polymake.org",
      "4.9"
    ]
  },
  "_info": {
    "description": "cube of dimension 2"
  },
  "TRIANGULATION": [{
      "FACETS": [[ 0, 1, 2],
                 [ 1, 2, 3]],
      "_id": "unnamed#0",
      "F_VECTOR": [4, 5, 2]
  }],
  "N_FACETS": 4,
  "_id": "square",
  "BOUNDED": true,
  "FACETS": [{ "1": "1" }, { "0": "1/3", "1": "-1"},
             { "2": "1" }, { "0": "1/3", "2": "-1" },
             {     "cols": 3    }],
  "_type": "polytope::Polytope<Rational>",
  "VOLUME": "1/9"
}
```


- First version (XML) published in 2016.
- Currently using JSON.
- Builds on already existing infrastructure.

# The Polymake File Format


```
{
  "_attrs": {
    "FACETS": {
      "_type": "SparseMatrix<Rational, NonSymmetric>"
    }
  },
  "VERTICES": [["1", "0", "0"],
               ["1", "1/3", "0"],
               ["1", "0", "1/3"],
               ["1", "1/3", "1/3"]],
  "CONE_AMBIENT_DIM": 3,
  "N_VERTICES": 4,
  "_ns": {
    "polymake": [
      "https://polymake.org",
      "4.9"
    ]
  },
  "_info": {
    "description": "cube of dimension 2"
  },
  "TRIANGULATION": [{
      "FACETS": [[ 0, 1, 2],
                 [ 1, 2, 3]],
      "_id": "unnamed#0",
      "F_VECTOR": [4, 5, 2]
  }],
  "N_FACETS": 4,
  "_id": "square",
  "BOUNDED": true,
  "FACETS": [{ "1": "1" }, { "0": "1/3", "1": "-1"},
             { "2": "1" }, { "0": "1/3", "2": "-1" },
             {      "cols": 3    }],
  "_type": "polytope::Polytope<Rational>",
  "VOLUME": "1/9"
}
```


- First version (XML) published in 2016.
- Currently using JSON.
- Builds on already existing infrastructure.
- Extensible, tree structure.

# The Polymake File Format


```json
{
  "_attrs": {
    "FACETS": {
      "_type": "SparseMatrix<Rational, NonSymmetric>"
    }
  },
  "VERTICES": [["1", "0", "0"],
               ["1", "1/3", "0"],
               ["1", "0", "1/3"],
               ["1", "1/3", "1/3"]],
  "CONE_AMBIENT_DIM": 3,
  "N_VERTICES": 4,
  "_ns": {
    "polymake": [
      "https://polymake.org",
      "4.9"
    ]
  },
  "_info": {
    "description": "cube of dimension 2"
  },
  "TRIANGULATION": [{
      "FACETS": [[ 0, 1, 2],
                 [ 1, 2, 3]],
      "_id": "unnamed#0",
      "F_VECTOR": [4, 5, 2]
  }],
  "N_FACETS": 4,
  "_id": "square",
  "BOUNDED": true,
  "FACETS": [{ "1": "1" }, { "0": "1/3", "1": "-1"},
             { "2": "1" }, { "0": "1/3", "2": "-1" },
             {      "cols": 3    }],
  "_type": "polytope::Polytope<Rational>",
  "VOLUME": "1/9"
}
```


- First version (XML) published in 2016.
- Currently using JSON.
- Builds on already existing infrastructure.
- Extensible, tree structure.
- Has a Schema.

# The Polymake File Format


```json
{
  "_attrs": {
    "FACETS": {
      "_type": "SparseMatrix<Rational, NonSymmetric>"
    }
  },
  "VERTICES": [["1", "0", "0"],
               ["1", "1/3", "0"],
               ["1", "0", "1/3"],
               ["1", "1/3", "1/3"]],
  "CONE_AMBIENT_DIM": 3,
  "N_VERTICES": 4,
  "_ns": {
    "polymake": [
      "https://polymake.org",
      "4.9"
    ]
  },
  "_info": {
    "description": "cube of dimension 2"
  },
  "TRIANGULATION": [{
      "FACETS": [[ 0, 1, 2],
                 [ 1, 2, 3]],
      "_id": "unnamed#0",
      "F_VECTOR": [4, 5, 2]
  }],
  "N_FACETS": 4,
  "_id": "square",
  "BOUNDED": true,
  "FACETS": [{ "1": "1" }, { "0": "1/3", "1": "-1"},
             { "2": "1" }, { "0": "1/3", "2": "-1" },
             {     "cols": 3    }],
  "_type": "polytope::Polytope<Rational>",
  "VOLUME": "1/9"
}
```


- First version (XML) published in 2016.
- Currently using JSON.
- Builds on already existing infrastructure.
- Extensible, tree structure.
- Has a Schema.
- Older formats will be upgraded on load.

# The Polymake File Format


```
{
  "_attrs": {
    "FACETS": {
      "_type": "SparseMatrix<Rational, NonSymmetric>"
    }
  },
  "VERTICES": [["1", "0", "0"],
              ["1", "1/3", "0"],
              ["1", "0", "1/3"],
              ["1", "1/3", "1/3"]],
  "CONE_AMBIENT_DIM": 3,
  "N_VERTICES": 4,
  "_ns": {
    "polymake": [
      "https://polymake.org",
      "4.9"
    ]
  },
  "_info": {
    "description": "cube of dimension 2"
  },
  "TRIANGULATION": [{
      "FACETS": [[ 0, 1, 2],
                 [ 1, 2, 3]],
      "_id": "unnamed#0",
      "F_VECTOR": [4, 5, 2]
  }],
  "N_FACETS": 4,
  "_id": "square",
  "BOUNDED": true,
  "FACETS": [{ "1": "1" }, { "0": "1/3", "1": "-1"},
           { "2": "1" }, { "0": "1/3", "2": "-1" },
           {     "cols": 3    }],
  "_type": "polytope::Polytope<Rational>",
  "VOLUME": "1/9"
}
```


- First version (XML) published in 2016.
- Currently using JSON.
- Builds on already existing infrastructure.
- Extensible, tree structure.
- Has a Schema.
- Older formats will be upgraded on load.

## The `mrdi` File Format

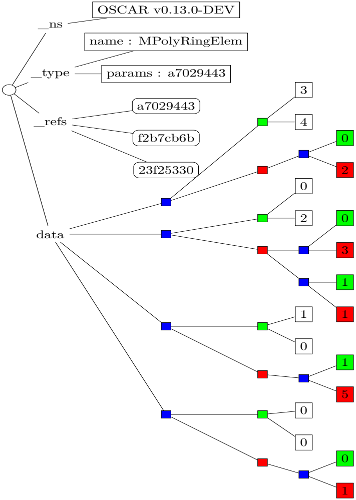
**2024, joint work with Michael Joswig and Benjamin Lorenz**

- JSON based file format.
- Similar to `polymake` format but generalizes to include algebraic data.
- Uses namespaces for semantic seperation.
- Uses references stored with UUIDs.
- Prototype developed using `OSCAR`.

# Demo

```
julia> using Oscar

julia> F = GF(7)
Prime field of characteristic 7

julia> L, a = finite_field(x^2 + 1)
(Finite field of degree 2 and characteristic 7, o)

julia> Lyz, (y, z) = L[:y, :z]
(Multivariate polynomial ring in 2 variables over L, FqMPolyRingElem[y, z])

julia> p = 2 * z^4 * y^3 + (a + 3) * z^2 + 5 * a * y + 1
2*y^3*z^4 + 5*o*y + (o + 3)*z^2 + 1

julia> q = z^2 + 3 * y
3*y + z^2

julia> save("p.mrdi", p)

julia> save("q.mrdi", q)

julia> Oscar.reset_global_serializer_state()
Dict{Base.UUID, Any}()

julia> load("p.mrdi") * load("q.mrdi")
6*y^4*z^4 + 2*y^3*z^6 + o*y^2 + (o + 2)*y*z^2 + 3*y + (o + 3)*z^4 + z^2
```

# Tree Structure




$2y^3z^4$

$(a+3)z^2$

$5ay$

$1$

# Example File Serialized with OSCAR


```
{
  "_ns": { "Oscar": [ "https://github.com/oscar-system/Oscar.jl", "1.0.0" ]  },
  "_type": {
    "name": "MPolyRingElem",
    "params": "869a359a-43d3-43f4-9821-0af9346be019"
  },
  "data": [[["3", "4"], [["0", "2"]]],
           [["0", "2"], [["0", "3"], ["1", "1"]]],
           [["1", "0"], [["1", "5"]]],
           [["0", "0"], [["0", "1"]]]],
  "_refs": {
    "152ac7bd-e85a-4b36-acc2-743ade2cad4f": {
      "data": { "base_ring": { "data": "7",  "_type": "FqField"},
                "symbols": ["x"] },
      "_type": "PolyRing"
    },
    "869a359a-43d3-43f4-9821-0af9346be019": {
      "data": {
        "base_ring": "a8309b96-caec-443c-bedb-e23bb0634c14",
        "symbols": [ "y", "z" ]
      },
      "_type": "MPolyRing"  },
    "a8309b96-caec-443c-bedb-e23bb0634c14": {
      "data": {
        "def_pol": {
          "data": [["0", "1"], ["2", "1"]],
          "_type": {
            "name": "PolyRingElem",
            "params": "152ac7bd-e85a-4b36-acc2-743ade2cad4f"
          }
        }
      },
      "_type": "FqField"
    }
  }
}
```

# Parallelization

```
channels = Oscar.params_channels(Union{Ring, MatSpace})

Qx, x = QQ["x"]
F, a = number_field(x^2 + x + 1)
MR = matrix_space(F, 2, 2)

Oscar.put_params(channels, Qx)
Oscar.put_params(channels, F)
Oscar.put_params(channels, MR)

c = [MR([a^i F(1); a a + 1]) for i in 1:5]
dets = pmap(det, c)
total = reduce(*, dets)
```

# Beyond OSCAR (within Julia)

```
struct LabelledPolynomial
  p::MPolyRingElem
  l::String
end

function save_object(s::SerializerState, l_p::LabelledPolynomial)
  save_data_dict(s) do _
    save_typed_object(s, LabelledPolynomial.p, :poly)
    save_object(s, LabelledPolynomial.l, :label)
  end
end

function load_object(s::DeserializerState)
  p = load_typed_object(s, :poly)
  l = load_object(s, String, :label)

  return LabelledPolynomial(p, l)
end
```

# Beyond OSCAR (beyond Julia)

## Other Implementations

- Magma
- Sage
- CoCoa
- Lean

# Johnson Solids

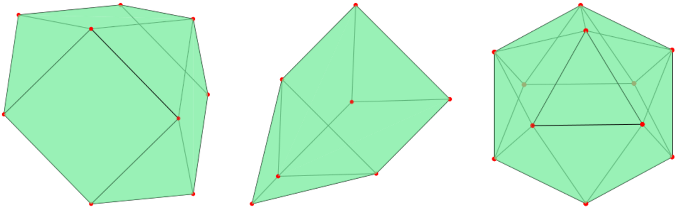


**Fig. 4.** (a) Triangular cupola, (b) elongated triangular pyramid, (c) gyroelongated pentagonal pyramid

```
> jq '.data.float.VERTICES | length' j3
9
> jq '.data.float.VERTICES | length' j7
7
> jq '.data.float.VERTICES | length' j11
11
```

https://zenodo.org/records/10729583

# Other datasets

- Surfaces in $\mathbb{P}^4$
- QSM Models in F-theory
- Small Phylogenetic Tree Website (ongoing work)
- ...

# Schema

Figure: https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/

- A schema defines a structure for data.

# Schema

Figure: `https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/`

- A schema defines a structure for data.
- Schema languages. (RELAX NG [2002], JSON Schema [2022])

## Schema

Figure: https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/

- A schema defines a structure for data.
- Schema languages. (RELAX NG [2002], JSON Schema [2022])
- Is possible to define recursive structure.

# Schema

Figure: https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/

- A schema defines a structure for data.
- Schema languages. (RELAX NG [2002], JSON Schema [2022])
- Is possible to define recursive structure.
- Schemata allow data to be validated before loading.

## Schema

Figure: `https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/`

- A schema defines a structure for data.
- Schema languages. (RELAX NG [2002], JSON Schema [2022])
- Is possible to define recursive structure.
- Schemata allow data to be validated before loading.
- Adds structure to document based databases.

# Schema

Figure: https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/

- A schema defines a structure for data.
- Schema languages. (RELAX NG [2002], JSON Schema [2022])
- Is possible to define recursive structure.
- Schemata allow data to be validated before loading.
- Adds structure to document based databases.
- `PolyDB`, Paffenholz [2017]

You can find more information here

https://arxiv.org/abs/2309.00465

Thank You!