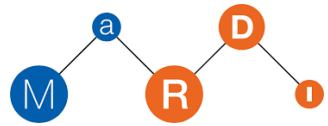
A FAIR File Format for Mathematical Software

Antony Della Vecchia

Joint work with M. Joswig and B. Lorenz

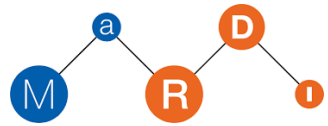
Technische Universität Berlin

Towards a digital infrastructure for mathematical research
2023-09-25



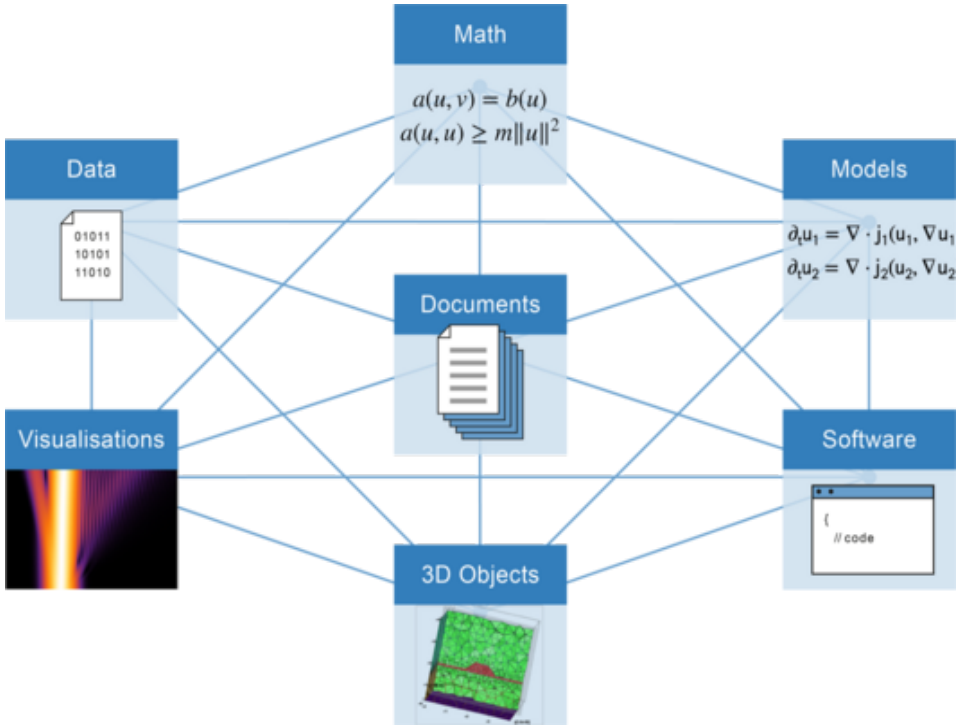
Overview

- MaRDI and the FAIR principles
- History of Files in Mathematical Software
- Technicalities with Algebraic Data
- Current Status of Prototype
- The File Format Specification
- Future Work



MaRDI and the FAIR Principles

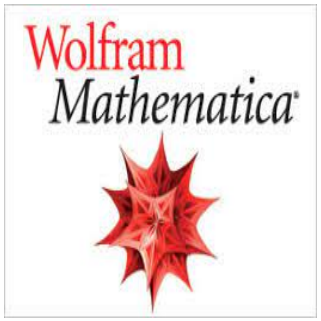
- **Ma**thematics **R**esearch **D**ata Initiative.
- Develop a mathematical research data infrastructure.
- Set standards for confirmable workflows and certified mathematical research data.
- Provide services to both the mathematical and wider scientific community.
- **F**indable **A**ccessible **I**nteroperable **R**eusable [M. D. Wilkinson et al. 2016]



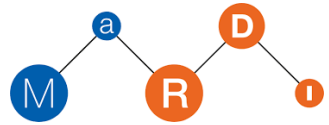
The Importance of Files for Computer Algebra

- People often have a preferred software system.

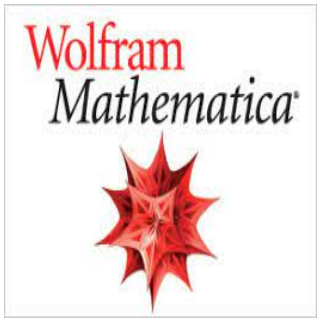
The Importance of Files for Computer Algebra



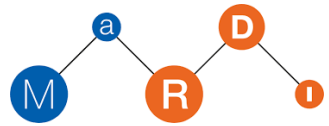
- People often have a preferred software system.
- Computations can be expensive.



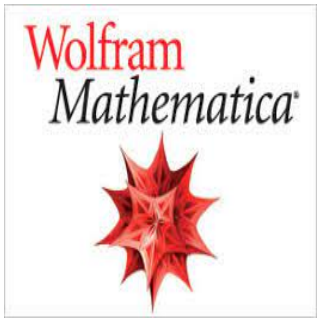
The Importance of Files for Computer Algebra



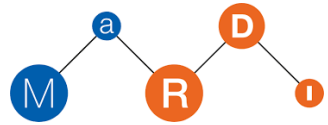
- People often have a preferred software system.
- Computations can be expensive.
- Software changes often and requires maintenance.



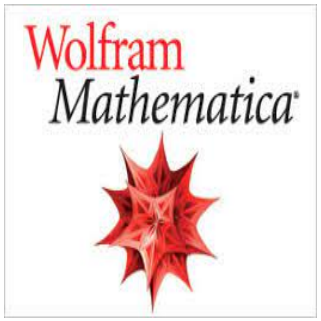
The Importance of Files for Computer Algebra



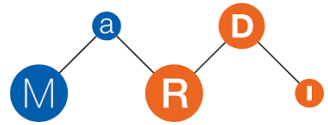
- People often have a preferred software system.
- Computations can be expensive.
- Software changes often and requires maintenance.
- Data is often more valuable than the software.



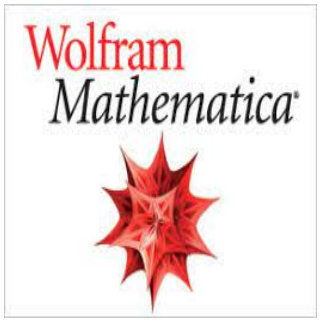
The Importance of Files for Computer Algebra



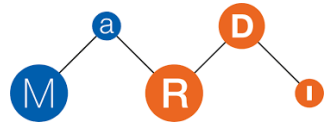
- People often have a preferred software system.
- Computations can be expensive.
- Software changes often and requires maintenance.
- Data is often more valuable than the software.
- Verification of results is at most as computationally expensive.



The Importance of Files for Computer Algebra



- People often have a preferred software system.
- Computations can be expensive.
- Software changes often and requires maintenance.
- Data is often more valuable than the software.
- Verification of results is at most as computationally expensive.

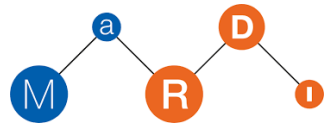


Storing Computer Algebra Data

- It's common to have multiple perspectives on an object in mathematics.
- While storing mathematical data a choice of perspective must me made.
- Such a choice might not be describeable in an email.

Say we want to store:

$$p = 2y^3z^4 + (\mathbf{a} + 3)z^2 + 5\mathbf{a}y + 1$$



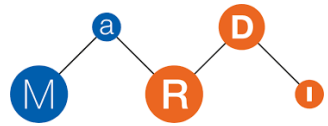
Storing Computer Algebra Data

- It's common to have multiple perspectives on an object in mathematics.
- While storing mathematical data a choice of perspective must me made.
- Such a choice might not be describeable in an email.

Say we want to store:

$$p = 2y^3z^4 + (\mathbf{a} + 3)z^2 + 5\mathbf{a}y + 1$$

- Some technicalities with the coefficients.
- Is y considered a coefficient of z?
- What is **a**?
- How can we guarantee the objects behave as expected on load?

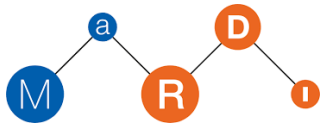


History of File Formats

```
* 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
```

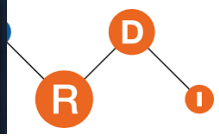

```
{
  "_attrs": {
    "FACETS": {
      "_type": "SparseMatrix<Rational, NonSymmetric>"
    }
  },
  "VERTICES": [["1", "0", "0"],
               ["1", "1/3", "0"],
               ["1", "0", "1/3"],
               ["1", "1/3", "1/3"]],
  "CONE_AMBIENT_DIM": 3,
  "N_VERTICES": 4,
  "_ns": {
    "polymake": [
      "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"
}
```


- The LP file format, the MPS file format. IBM [1970s]
- `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]

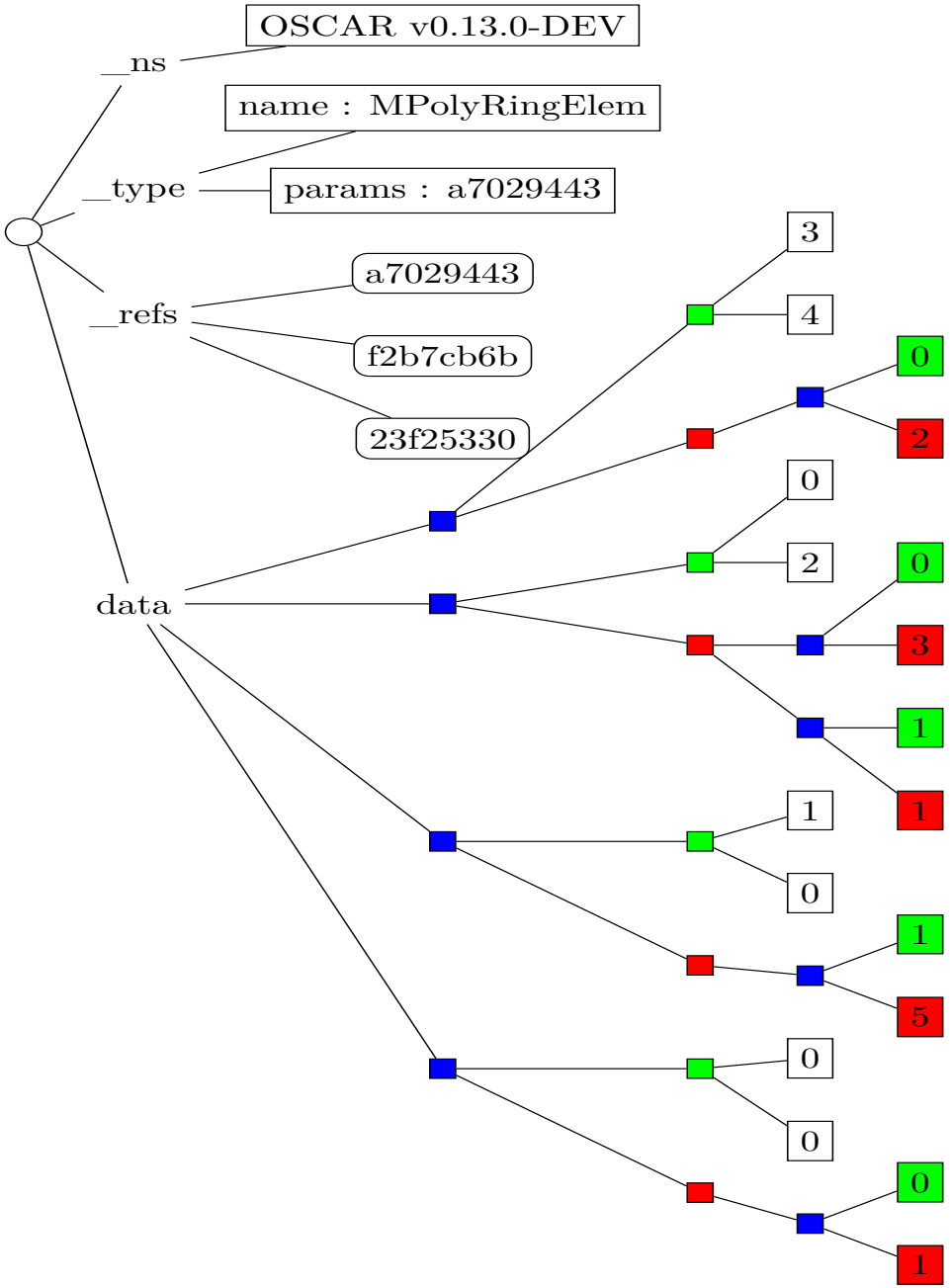


OSCAR Demo

```
julia> using Oscar

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

julia> Fx, x = F["x"]
(Univariate polynomial ring in x over GF(7), x)

julia> L, a = FiniteField(x^2 + x + 1)
(Finite field of degree 2 over GF(7), o)

julia> Lyz, (y, z) = L["y", "z"]
(Multivariate polynomial ring in 2 variables over GF(7^2), fqPolyRepMPolyRingElem[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> 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
```



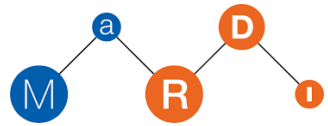
Tree Structure



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

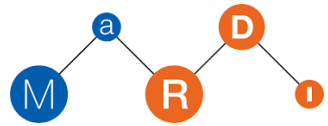
Example File Serialized with OSCAR


```
{
  "_ns": { "Oscar": [ "https://github.com/oscar-system/Oscar.jl", "0.13.0-DEV" ]  },
  "_type": {
    "name": "MPolyRingElem",
    "params": "869a359a-43d3-43f4-9821-0af9346be019"
  },
  "_refs": {
    "152ac7bd-e85a-4b36-acc2-743ade2cad4f": {
      "_type": "PolyRing",
      "data": { "base_ring": { "data": "7",  "_type": "Nemo.fpField"},
                "symbols": ["x"] }
    },
    "869a359a-43d3-43f4-9821-0af9346be019": {
      "_type": "MPolyRing",
      "data": {
        "base_ring": "a8309b96-caec-443c-bedb-e23bb0634c14",
        "symbols": [ "y", "z" ]
      }
    },
    "a8309b96-caec-443c-bedb-e23bb0634c14": {
      "_type": "fqPolyRepField",
      "data": {
        "def_pol": {
          "_type": {
            "name": "PolyRingElem",
            "params": "152ac7bd-e85a-4b36-acc2-743ade2cad4f"
          },
          "data": [["0", "1"], ["1", "1"], ["2", "1"]]
        }
      }
    }
  }
  "data": [[["3", "4"], [["0", "2"]]],
           [["0", "2"], [["0", "3"], ["1", "1"]]],
           [["1", "0"], [["1", "5"]]],
           [["0", "0"], [["0", "1"]]]]
}
```

Current Status and Features of OSCAR Serialization

- Over 100* registered types.
- Can store sessions over multiple files.
- Parameter Overriding.
- Serialization extensible from outside OSCAR due to Julia multiple dispatch.
- Option to attach metadata (name and ORCID for author).
- Upgrade scripts.



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]

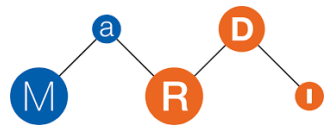
File Format Specification

```
julia> mrdi_schema = Schema(JSON.parsefile(schema_path))
A JSONSchema

julia> jsondict = JSON.parsefile(polynomial_path)
Dict{String, Any} with 4 entries:
  "data"  => Any[Any[Any["3", "4"], Any[Any["0", "2"]]], Any[Any["0", "2"], Ar
  "_ns"   => Dict{String, Any}("Oscar"=>Any["https://github.com/oscar-system/O
  "_type" => Dict{String, Any}("name"=>"MPolyRingElem", "params"=>"869a359a-43
  "_refs" => Dict{String, Any}("869a359a-43d3-43f4-9821-0af9346be019"=>Dict{St

julia> validate(mrdi_schema, jsondict)
```


```
{
  "$id": "https://oscar-system.github.io/schemas/data.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "_ns": { "type": "object" },
    "_type": {"oneOf": [
      {"type": "string"},
      {"type": "object",
       "properties": {
         "name": {"type": "string"},
         "params": {"$ref": "#/$defs/data"}
       }}
    ]},
    "data": {"$ref": "#/$defs/data"},
    "_refs": {
      "type": "object",
      "patternProperties":{
        "^[0-9a-fA-F]{8}-([0-9a-fA-F]{4}-){3}[0-9a-fA-F]{12}$":
        {"$ref": "#"}
      }
    }
  },
  "required": ["_type"],
  "$defs": {
    "data": {
      "oneOf": [
        {"type": "string"},
        {"type": "object",
         "patternProperties": {
           "[a-z]": { "$ref": "#/$defs/data" }
         }
        },
        {"type": "array", "items": {"$ref": "#/$defs/data"}}
      ]
    }
  }
}
```

On Going Work

- Add Functionality for most `OSCAR` types.
- Minimal example loaders in other software systems.
- Aim to be software independant.
- Setup small databases with collaborators using the File Format.



References

Thank You!

You can find more information here

https://arxiv.org/abs/2309.00465