

# The mrdi File Format

Antony Della Vecchia

Technische Universität Berlin

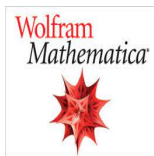
2024-02-05



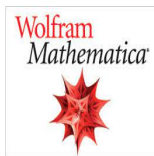
# The Importance of Files for Computer Algebra



- People often have a preferred software system.



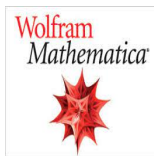
# The Importance of Files for Computer Algebra



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



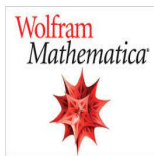
# The Importance of Files for Computer Algebra



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



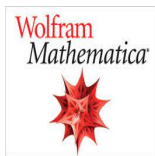
# The Importance of Files for Computer Algebra



- 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.



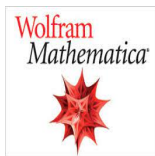
# The Importance of Files for Computer Algebra



- 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.



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

Say we want to store:

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





- It's common to have multiple perspectives on an object in mathematics.
- While storing mathematical data a choice of perspective must be 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  $\mathbf{a}$ ?
- How can we guarantee the objects behave as expected on load?



```
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
```



# 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" ] ] ] ]
}
```



# Namespaces

```
polytope > $c = cube(4);  
  
polytope > save($c, "~/c.poly");  
  
polytope >  
[2]+ Stopped polymake  
vecchia@priort:~> julia-1.10
```

```
| Documentation: https://docs.julialang.org  
| Type "?" for help, "]"? for Pkg help.  
| Version 1.10.0 (2023-12-25)  
| Official https://julialang.org/ release
```

```
julia> using Oscar
```

```
...combining (and extending) ANTIC, GAP, Polymake and Singular  
Version 0.14.0 ...  
... which comes with absolutely no warranty whatsoever  
Type: '?Oscar' for more information  
(c) 2019-2024 by The OSCAR Development Team
```

```
julia> load("~/c.poly")  
Polyhedron in ambient dimension 4
```





- A schema defines a structure for data.

Figure:

<https://www.pexels.com/photo/plastic-shape-sorter-toy-11030155/>





Figure:

<https://www.pexels.com/photo/plastic-shape-sorter-toy-11030155/>

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





Figure:

<https://www.pexels.com/photo/plastic-shape-sorter-toy-11030155/>

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





Figure:

<https://www.pexels.com/photo/plastic-shape-sorter-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.







Figure:

<https://www.pexels.com/photo/plastic-shape-sorter-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.





Figure:

<https://www.pexels.com/photo/plastic-shape-sorter-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"], Any[

"\_ns" => Dict{String, Any}("Oscar"=>Any["https://github.com/oscar-system/

"\_type" => Dict{String, Any}("name"=>"MPolyRingElem", "params"=>"869a359a-43

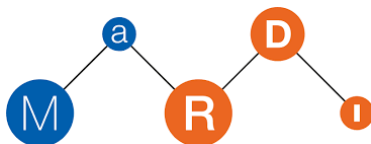
"\_refs" => Dict{String, Any}("869a359a-43d3-43f4-9821-0af9346be019"=>Dict{St

```
julia> validate(mrdischema, 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" }
          }
        ]
      }
    }
  }
}
```



- **M**athematics **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]



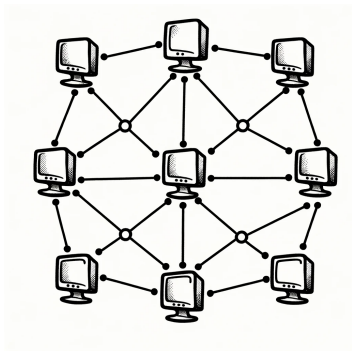


Figure: ChatGPT

```
process_ids = addprocs(5)

@everywhere using Oscar

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)

@test total == F(4)
```



Thank you!

