

Tutorial: Serialization

Antony Della Vecchia

Technische Universität Berlin

polymake Workshop
2024-02-02



- What is Serialization?
- `polymake` Serialization
- OSCAR Serialization
- Differences and Similarities



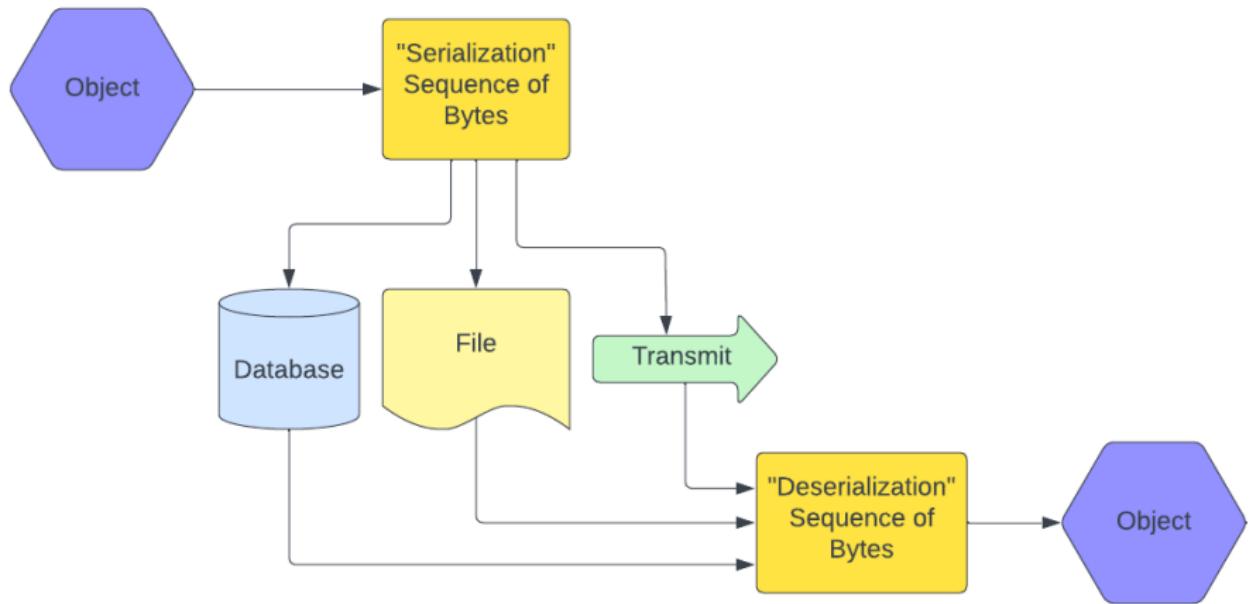
Serialization



generated using ChatGPT



Serialization: High Level Overview



Serialization: Formats

```
<?xml version="1.0" encoding="utf-8"?>
<?pm chk="56e977e8"?>
<object name="square" type="polytope::Polytope<Rational>" 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>
            <v>1 1/3 0</v>
            <v>1 0 1/3</v>
            <v>1 1/3 1/3</v>
        </m>
    </property>
    <property name="FACETS">
        type="SparseMatrix<Rational,NonSymmetric>;">
        <m cols="3">
            <v> <e i="1">1</e> </v>
            <v> <e i="0">1/3</e> <e i="1">-1</e> </v>
            <v> <e i="2">1</e> </v>
            <v> <e i="0">1/3</e> <e i="2">-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>
```

- Text (MPS, LP)
- CSV
- XML (polymake)
- JSON (polymake, OSCAR)
- YAML
- ...



Polymake Demo

```
polytope > $c = cube(2);
polytope > $c->add("LP", LINEAR_OBJECTIVE=>[0, 1, -1]);
polytope > save($c, "c.json");
polytope > poly2lp($c, $c->LP, 0, "c.lp");
polytope > poly2mps($c, $c->LP, new Set<Int>(), "c.mps");
```

```
MINIMIZE
    obj: +1 x1 -1 x2
Subject To
    ie0: +1 x1 >= -1
    ie1: -1 x1 >= -1
    ie2: +1 x2 >= -1
    ie3: -1 x2 >= -1
BOUNDS
    x1 free
    x2 free
END
```

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



polymake JSON File

```
{  
    "_id": "c.json",  
    "_info": {  
        "description": "cube of dimension 2\\n"  
    },  
    "_ns": {  
        "polymake": [  
            "https://polymake.org",  
            "4.11"  
        ]  
    },  
    "_type": "polytope::Polytope<Rational>",  
    "BOUNDED": true,  
    "CONE_AMBIENT_DIM": 3,  
    "CONE_DIM": 3,  
    "FACETS": [ { "0": "1", "1": "1" },  
                { "0": "1", "1": "-1" },  
                { "0": "1", "2": "1" },  
                { "0": "1", "2": "-1" },  
                { "cols": 3 } ],  
    "VERTICES_IN_FACETS": [[0, 2], [1, 3], [0, 1], [2, 3], {"cols": 4}],  
    "_attrs": {  
        "FACETS": {  
            "_type": "SparseMatrix<Rational, NonSymmetric>"  
        }  
    }  
}
```



Say we want to store:

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



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?



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*a*y + (o + 3)*z^2 + 1

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

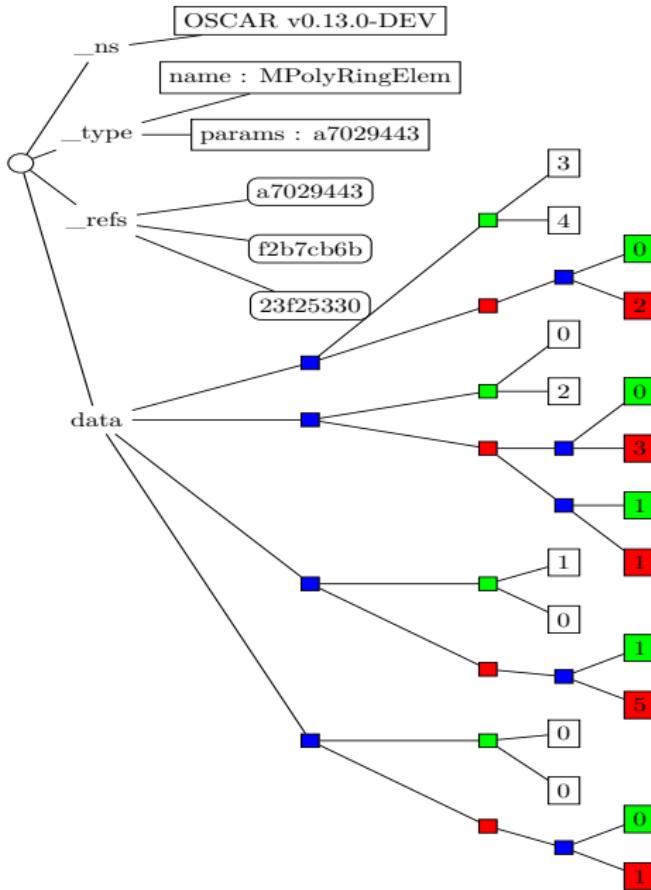
julia> save("p.mrdi", p)

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"]]]]]  
            ]  
        ]  
    }  
}
```



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
```



Schema



- A schema defines a structure for data.

Figure:

[https://www.pexels.com/photo/
plastic-shape-shorter-toy-11030155/](https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/)



Schema



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

Figure:

[https://www.pexels.com/photo/
plastic-shape-shorter-toy-11030155/](https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/)



Schema



- 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-shorter-toy-11030155/](https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/)



Schema



- 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-shorter-toy-11030155/](https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/)



Schema



- 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-shorter-toy-11030155/](https://www.pexels.com/photo/plastic-shape-shorter-toy-11030155/)



Schema



Figure:

[https://www.pexels.com/photo/
plastic-shape-shorter-toy-11030155/](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:

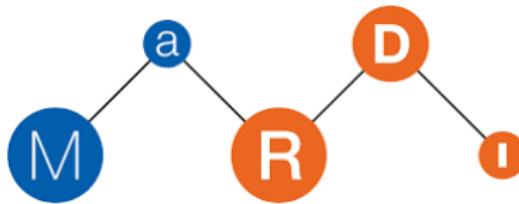
```
"data" => Any[Any[Any["3", "4"], Any[Any["0", "2"]]], Any[Any["0", "2"], Any["1", "3"]], Any["5"], Any["6"])
"_ns" => Dict{String, Any}("Oscar"=>Any["https://github.com/oscar-system/Oscar.jl"])
"_type" => Dict{String, Any}("name"=>"MPolyRingElem", "params"=>"869a359a-43d3-43f4-9821-0af9346be019")
"_refs" => Dict{String, Any}("869a359a-43d3-43f4-9821-0af9346be019"=>Dict{String, Any}("id"=>Any["https://oscar-system.github.io/schemas/data.json"], "schema"=>Any["https://json-schema.org/draft/2020-12/schema"]))
```

```
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]{0,}{-(0-9a-fA-F){0,}-{1,}}[0-9a-fA-F]{12}[^$]": {
          "$ref": "#"
        }
      }
    }
  },
  "required": ["_type"],
  "defs": {
    "data": {
      "oneOf": [
        { "type": "string" },
        { "type": "object" },
        { "patternProperties": {
            "[x-z]": { "$ref": "#/defs/data" }
          } }
        ],
        { "type": "array", "items": { "$ref": "#/defs/data" } }
      ]
    }
  }
}
```



- 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.
- Findable Accessible Interoperable Reusable [M. D. Wilkinson et al. 2016]



Benefits of Julia

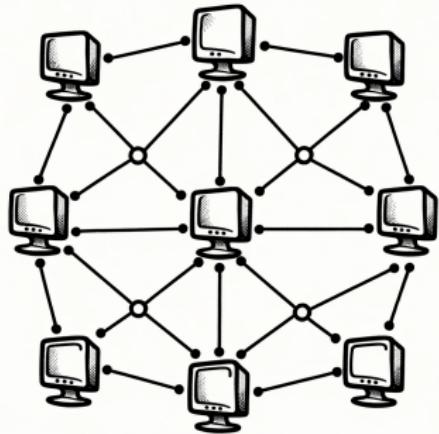


Figure: ChatGPT

```
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)
```



End

Now try and serialize something.