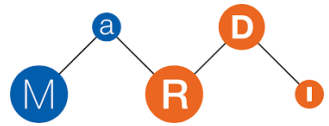
The `mrdi` File Format

Antony Della Vecchia

Joint work with M. Joswig and B. Lorenz

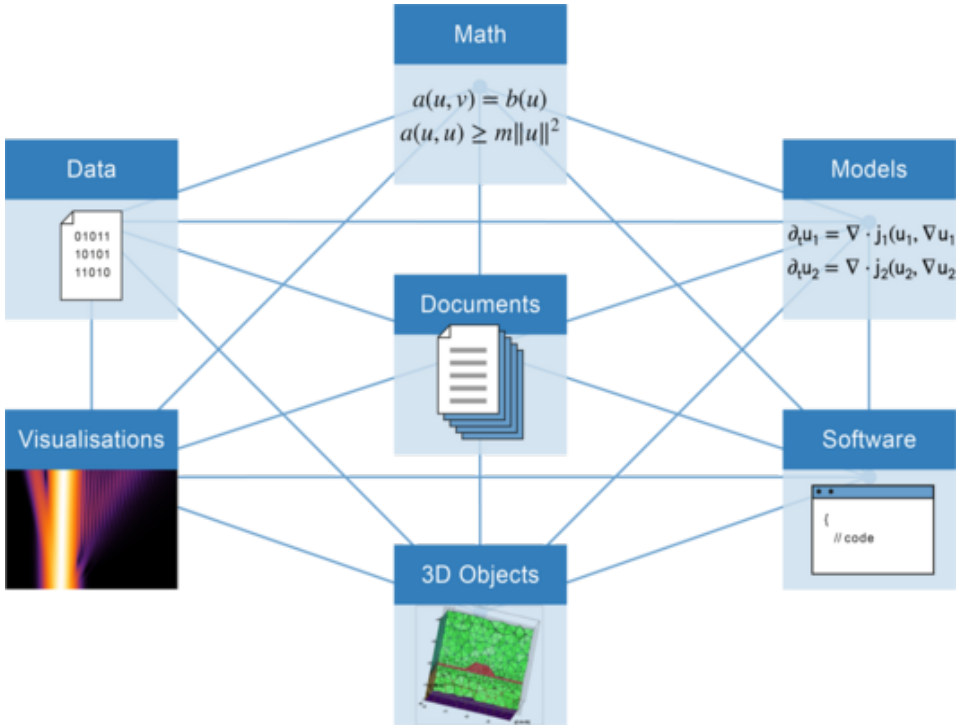
Technische Universität Berlin

2024-02-05



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]



Alice and Bob

[4]:
```
using Oscar
```

Alice's section of the Notebook

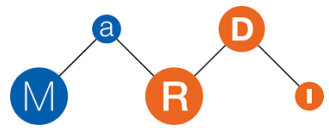
[5]:
```
Qxy, (x, y) = QQ[:x, :y]
I = ideal([x * y - 1, x^2 + y^2 -4])
gb = groebner_basis(I)
```

[5]:
```
Gröbner basis with elements
1 -> x*y - 1
2 -> x^2 + y^2 - 4
3 -> y^3 + x - 4*y
with respect to the ordering
degrevlex([x, y])
```

Bob's section of the Notebook

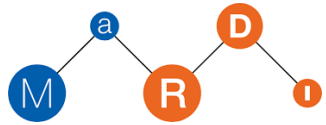
[6]:
```
[evaluate(p, [1//2, 4]) for p in gb]
```

[6]:
```
3-element Vector{QQFieldElem}:
 1
 49//4
 97//2
```



Alice's File

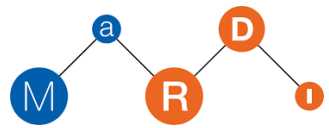
```
julia> using Oscar

julia> Qxy, (x, y) = QQ[:x, :y]
(Multivariate polynomial ring in 2 variables over QQ, QQMPolyRingElem[x, y])

julia> o = monomial_ordering(Qxy, :lex)
lex([x, y])

julia> I = ideal(Qxy, [x*y - 1, x^2 + y^2 - 4])
Ideal generated by
  x*y - 1
  x^2 + y^2 - 4

julia> gb = groebner_basis(I; ordering=o)
Gröbner basis with elements
1 -> y^4 - 4*y^2 + 1
2 -> x + y^3 - 4*y
with respect to the ordering
lex([x, y])

julia> save("./gb.mrdi", gb)
```



Bob loads

Bob can now choose which file to load and to do similar computations with.

```
[2]: using Oscar

[11]: gb = load("./gb.mrdi")

[11]: Ideal generating system with elements
      1 -> y^4 - 4*y^2 + 1
      2 -> x + y^3 - 4*y
      with associated ordering
      lex([x, y])

[12]: [evaluate(p, [1//2, 4]) for p in gb]

[12]: 2-element Vector{QQFieldElem}:
       193
       97//2
```



Bob's File

Bob wants to change labels, and use polynomial ring that lives in his environment.

```
[2]: using Oscar
```

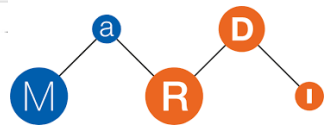
```
[13]: Qzw, (z, w) = QQ[:z, :w]
gb = load("./gb.mrdi"; params=Qzw)
```

```
[13]: Ideal generating system with elements
1 -> w^4 - 4*w^2 + 1
2 -> z + w^3 - 4*w
with associated ordering
lex([z, w])
```

```
[23]: v = [QQ(1//2), QQ(4)]
new_polys = [z * p + w for p in gb]
calc = [evaluate(p, v) for p in new_polys]
```

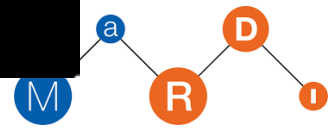
```
[23]: 2-element Vector{QQFieldElem}:
 201//2
 113//4
```

```
[25]: result = (v = v, calc = calc, new_polys = new_polys)
save("./result.mrdi", result)
```



result.mrdi

```
{
  "_ns": {"Oscar": ["https://github.com/oscar-system/Oscar.jl", "0.15.0" ]  },
  "_type": {
    "name": "NamedTuple",
    "params": { "tuple_params": [ ...   ],
      "names": ["v", "calc", "new_polys" ]
    }
  },
  "data": [["1//2", "4" ],
           [ "201//2", "113//4"],
    [ ...
    ]
  ],
  "_refs": {
    "2c15b14c-ab1a-43bc-b70c-54746845323c": {
      "_type": "MPolyRing",
      "data": {
        "base_ring": {
          "_type": "QQField"
        },
        "symbols": [
          "z",
          "w"
        ]
      }
    }
  }
}
```



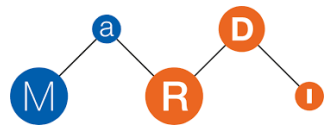
Parsing Outside OSCAR

```
[13]: using JSON
      f = open("./result.mrdi")
      result_dict = JSON.parse(f)
      result_dict["_type"]["params"]["names"]
```

```
[13]: 3-element Vector{Any}:
       "v"
       "calc"
       "new_polys"
```

```
[20]: calc_strs = result_dict["data"][2]
      calc = parse.(Rational{Int}, calc_strs)
```

```
[20]: 2-element Vector{Rational{Int64}}:
       201//2
       113//4
```



10 Years Later

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

Latest OSCAR version is 2.10.5.

- There have been many contributions by other parties.
- They all used different OSCAR versions.

10 Years Later

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

Latest `OSCAR` version is `2.10.5`.

- There have been many contributions by other parties.
- They all used different `OSCAR` versions.

Solution

- `OSCAR` provides update scripts.
- Bob and Alice decide to define a schema based on the `mrdi` schema.

Collaboration of Processes

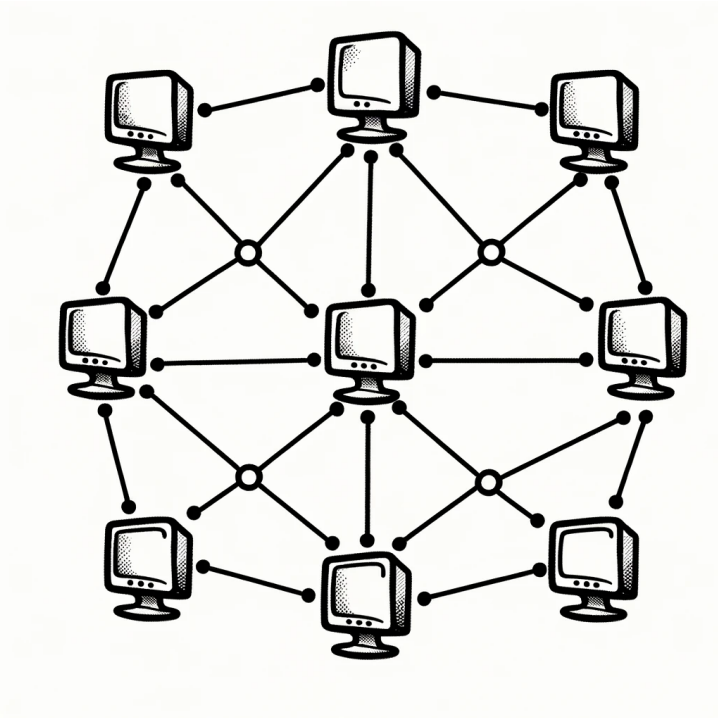
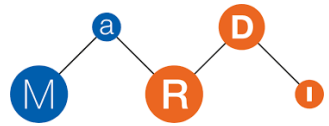


Figure: ChatGPT

```
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)
```

Alice and Bob's Database (Future Work)

- MongoDB is a document based database.
- Structure is determined by providing a schema.
- `PolyDB`, Paffenholz [2017].



End

Thank you!

See here for the details

`https://arxiv.org/abs/2309.00465`