Metadata extractor interoperability for materials science and chemistry

Matthew Evans

(UCLouvain & Matgenix)

MaRDA Annual Meeting
February 23 2023

marda-alliance/metadata_extractors

Introductions

Co-leads:

- **Matthew Evans (UCLouvain & Matgenix)**
- **Peter Kraus (TU Berlin)**
- **David Elbert (Johns Hopkins U. & MaRDA)**

The working group:

- 30 or so people spanning many projects/consortia in research data management, materials informatics, software engineering
- Assembled from conversations in 2022 at RDA, MADICES & MaRDA meeting

Scientific Extract-Transform-Load (ETL)

- Extracting data from instrument files, user-provided files, raw arrays and streams
- Transforming data into a tractable format for analysis/storage, e.g., Python objects, HDF5, NeXuS, JSON, XML
- Loading data into a queryable store, e.g., a database, archive

Motivation: User stories

- **Scientist** deposits raw data in archive, alongside a metadata file describing how the files could be parsed, and what the output would look like (enabling indexing/searching over the domain data)
- ELN/data management **software developer** wants to support a new file type: either has to write or use existing parser (with potentially complex dependencies and unclear outputs)
- **Communities/Ecosystems** can be fostered around parser code to improve quality and define more rigorous data models/transformations, thus reducing duplication of efforts.

Goals

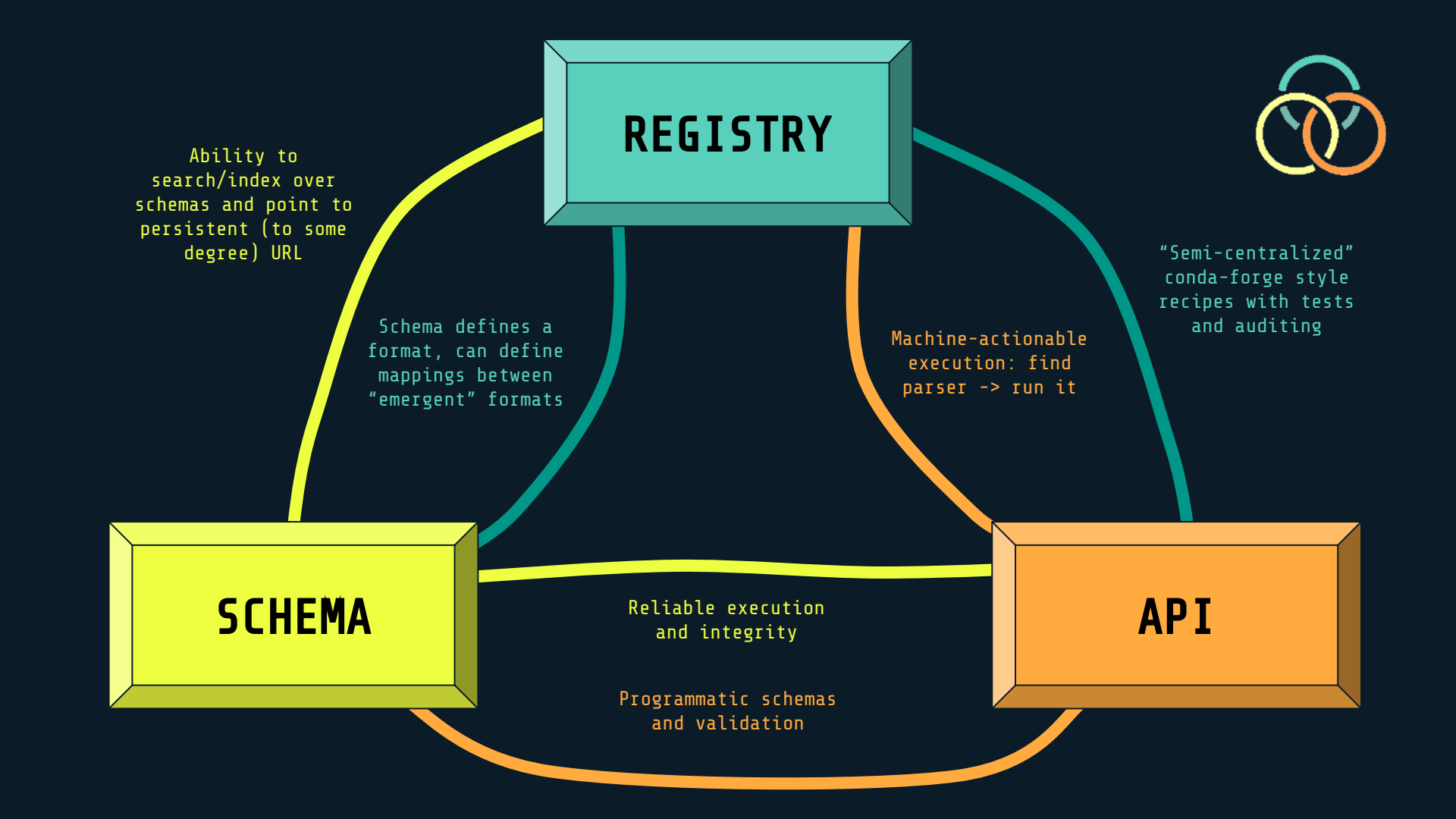
I. Enable infrastructure, archives or ELN developers robustly parse file types

II. Improve the ***quality*** and ***discoverability*** of parsers in the community with schemas and semantics

III. Indexing over relevant domain-specific data and metadata rather than using generic archives

Approach

I. A lightweight metadata schema for extractors
II. A common API specification for executing extractor code
III. A searchable registry of extractors and file types


REGISTRY
Ability to search/index over schemas and point to persistent (to some degree) URL
Schema defines a format, can define mappings between “emergent” formats
Machine-actionable execution: find parser -> run it
“Semi-centralized” conda-forge style recipes with tests and auditing
SCHEMA
Reliable execution and integrity
Programmatic schemas and validation
API

A lightweight metadata schema for extractors

`marda-alliance/metadata_extractors_schema`

https://linkml.io/

- Minimal description of inputs:
  - Instrument, vendor or software metadata, e.g., Bruker files produced by spectrometer xyz
  - Compatible versions
  - Human-readable caveats
  - Defined by examples where difficult
  - **Incrementally adoptable**
- Description of output “format”:
  - JSONSchema, RDF, JSON-LD, CSV-LD, LinkML
  - Always serialized via a second format?
  - c.f., WG on data dictionaries
  - **Agnostic to any well-described format**

- Have authored two schemas with LinkML: `FileType` and `Extractor`
- Currently only defined schemas that would be used in the registry, can then extend to the schemas that describe the outputs of extractors

Open questions

- Linking to other standards
- Online discussions around suitability of traditional schema languages for scientific data, data/metadata distinction and planning of output schema format (see later)

A common API specification for executing parser code

`marda-alliance/metadata_extractors_api`

- Unified user-facing interface for I/O through the parsers, e.g., (taking a CLI as an example)
- Well-defined entry point for invocation
- Accessible metadata (schemas, caveats etc. from WP1)
- Optional validation
- Packaging for external use:
    - multilingual, containerized, WASM?

- Still at scoping stage, lots of good discussion on GH.
- Informal agreement to leave this until the registry and schemas are in place
- Design should be driven by existing case studies and examples
- Balance between being too generic to be useful vs too specific
    - Where do we stop? Parsing is just a step in a workflow, could the same approach scale to an ML model which “parses” data and returns well defined outputs? Interacting APIs of autonomous labs?

A searchable registry of parsers

`marda-alliance/metadata_extractors_registry`

https://marda-registry.fly.dev/redoc

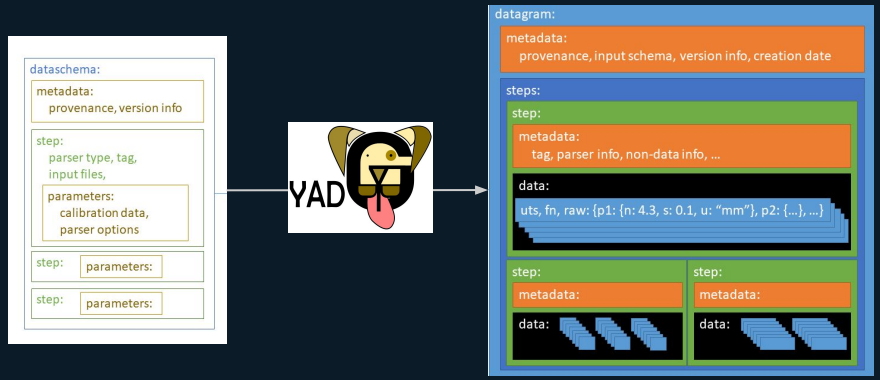
- Registry is live, using the live schemas from WP1!
- Submission currently via manual pull requests: can already add new file types that get validated in the CI
- Need to automate extractor registry process, possibilities:
  - Manually create yaml file for your code and make PRs to registry keep it up to date
  - Provide `.marda.yml` file in your code repo and get it scraped (say, once a week)
  - Write a web UI/form that generates the YAML to submit to the repo

Open questions

- Social mechanism for populating, curating and maintaining the registry
- User-facing tools built on top of registry: file type detection, copy-paste parsing
- Representing coupled collections of file types, e.g., Bruker TopSpin folder, VASP inputs/outputs (POSCAR, INCAR etc.)

Case study: yadg (Peter Kraus)

dgbowl/yadg



Main features:

- Timestamps, Units, Uncertainties
- BioLogic files (mpr, mpt)
- Agilent files (dx, ch)

Current usage pattern:

- write a DataSchema → `dataschema.yml`
- `yadg process dataschema.yml output.json`
- `output.json` is a yadg-specific structured json file...

Proposed MaRDA extractors usage pattern:

- `yadg extract biologic.mpr *.mpr`
- output: standardized FAIR format, probably NeXus...
- target: Q1/2023

Interested?

marda@ml-evs.science

- We are looking for:
  - Extractor libraries
  - Awkward file types
  - Vendors?
  - Schema/semantics experts
  - End users - library/infra developers and scientists
- Monthly meetings:
  - **March 21 at 15:00 UTC**
- Fortnightly office hours arranged on:
  - Slack (#automated-extractors)
  - GitHub Discussions