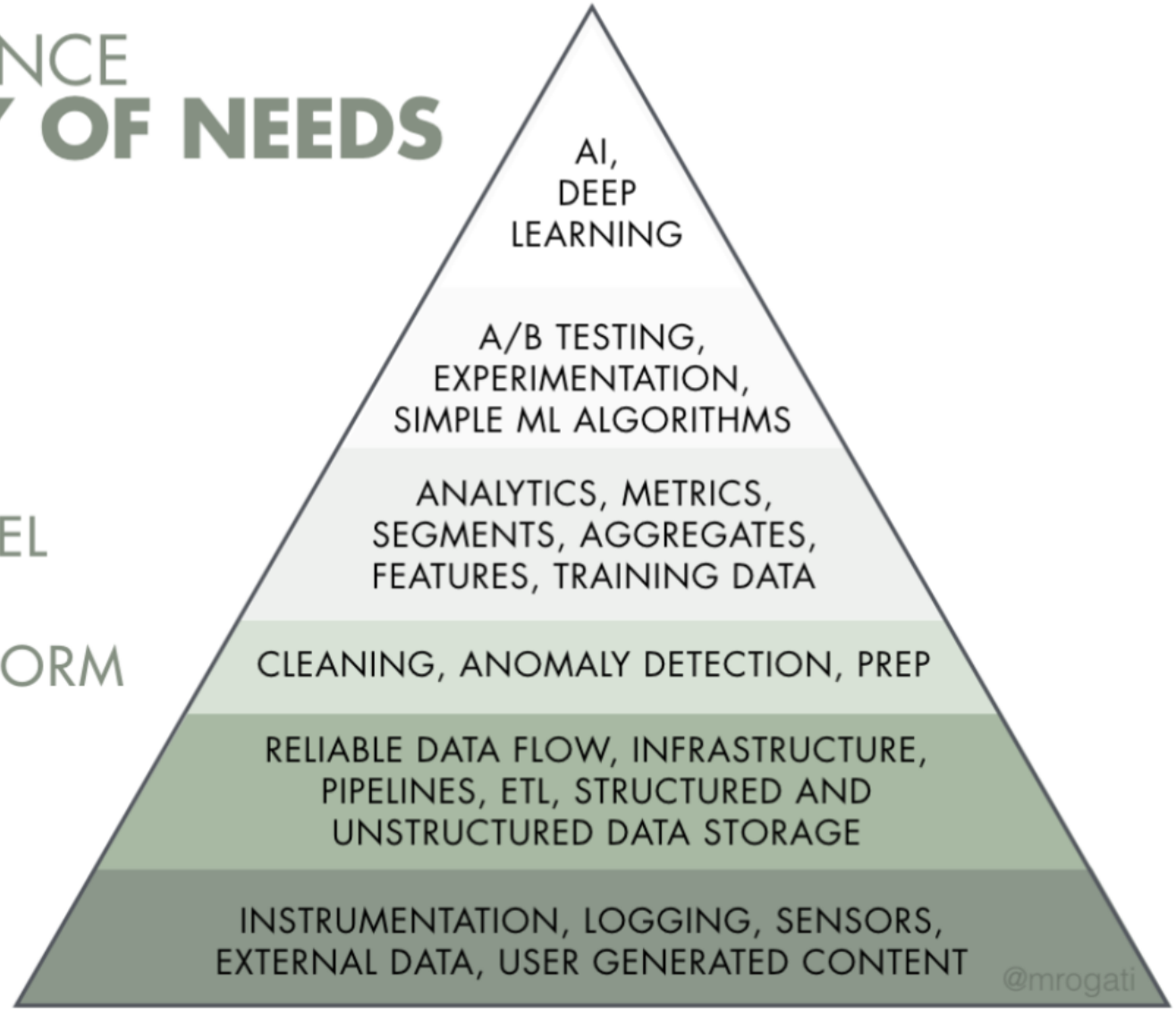
Implementing FAIR Principals Inside and Outside the Lab with Atomate and MongoDB

Kat Nykiel | Panos Manganaris

Overview

- Data Science Hierarchy of needs
- Features of Atomate
- Findable
  - Application to Historical Datasets
- Accessible
  - Application to Panos's Perovskite Property Modeling
- Interoperable
  - Application to Kat's High Throughput MXene Experiment
- Reusable
  - Application to Publication

THE DATA SCIENCE HIERARCHY OF NEEDS



LEARN/OPTIMIZE

AGGREGATE/LABEL

EXPLORE/TRANSFORM

MOVE/STORE

COLLECT

"The AI Hierarchy of Needs" by Monica Rogati on Medium

Features of Atomate

Workflow management plus analytical tools currently suited to VASP simulation

Why Atomate2 and Jobflow?

Simplified Workflow Definition

- lots of standard workflows
- @Job Decorator turns arbitrary user defined function into jobs

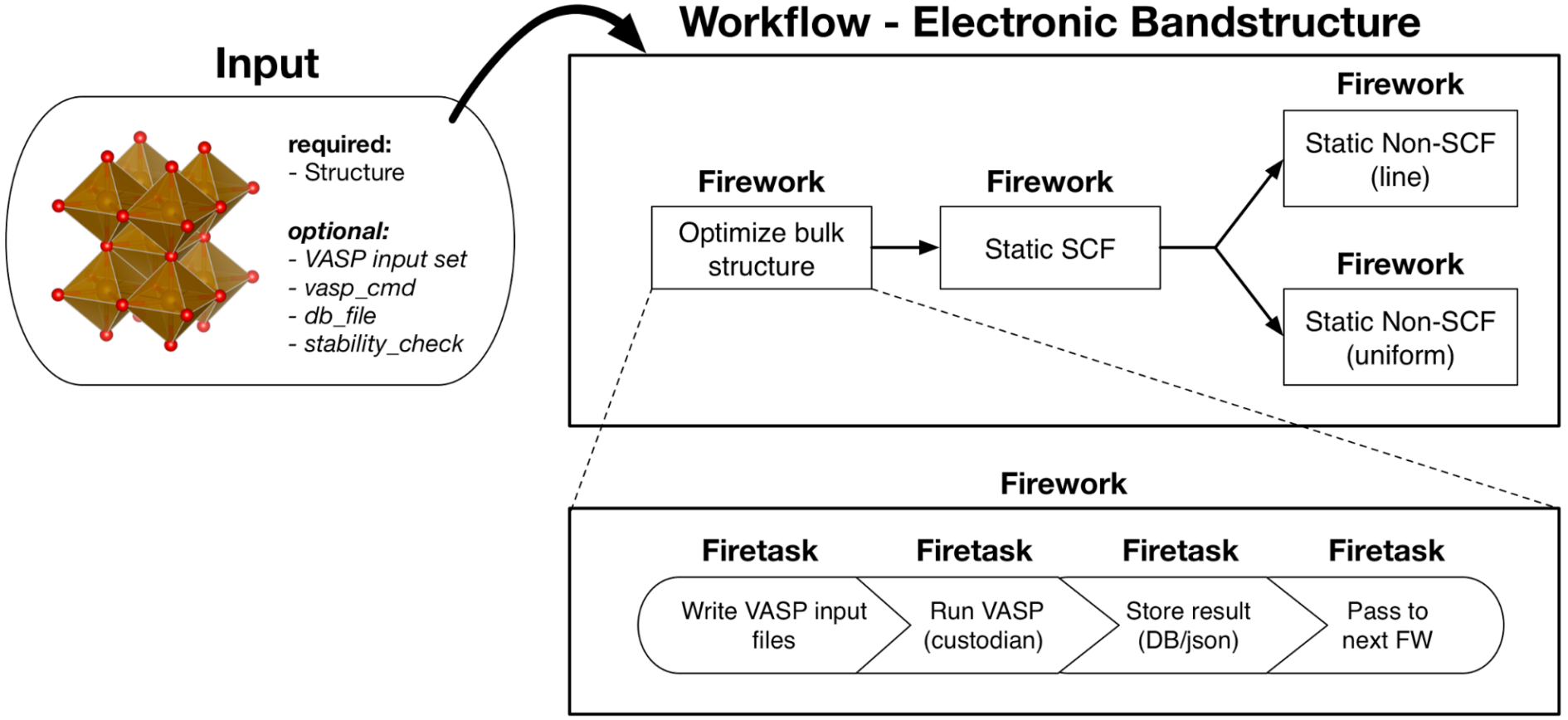
Standard Workflows

Atomate Defines VASP Simulations as python commands called workflows

- Supports FAIR data principles
- Supports high-throughput (HT) experiments
- Extendable to alternative simulators

Name	Type	Maker
Static	job	StaticMaker
Relax	job	RelaxMaker
Tight Relax	job	TightRelaxMaker
Dielectric	job	DielectricMaker
Transmuter	job	TransmuterMaker
HSE06 Static	job	HSEStaticMaker
HSE06 Relax	job	HSERelaxMaker
HSE06 Tight Relax	job	HSETightRelaxMaker
Double Relax	flow	DoubleRelaxMaker
Band Structure	flow	BandStructureMaker
Uniform Band Structure	flow	UniformBandStructureMaker
Line-Mode Band Structure	flow	LineModeBandStructureMaker
HSE06 Band Structure	flow	HSEBandStructureMaker
HSE06 Uniform Band Structure	flow	HSEUniformBandStructureMaker
HSE06 Line-Mode Band Structure	flow	HSELineModeBandStructureMaker
Relax and Band Structure	flow	RelaxBandStructureMaker
Elastic Constant	flow	ElasticMaker
Optics	flow	OpticsMaker
HSE06 Optics	flow	HSEOpticsMaker

Band-Structure Workflow with Atomate Jobs and FireWorks Flows



https://atomate.org/index.html

Flexible Database Integration

Execution and Analysis Options

- Integrate with state-of-the-art libraries
  - Pymatgen for setup and analysis
  - FireWorks for distributing workflows to supercomputers
- Store workflow definitions and outputs in database
- More efficient use of cluster queueing systems

Host with Purdue Geddes Cluster

- Enterprise Grade Continuity
- Containerized Workloads Ensure Portability
  - Move database across server commissioning/decommissioning
- Orders of Magnitude more Affordable than External Service Providers at $70/TB/Year
- Purdue Support

Findable

How Atomate Improves relevance of Legacy Work

Atomate Finds VASP Data Quickly

Pipelines are usually built in tandem with data generation.

Atomate API connects pipelines to data that predates the pipeline.

```
drone = VaspDrone()
vaspaths = list(chain.from_iterable(
  [drone.get_valid_paths(x) for x in
 os.walk('.') if drone.get_valid_paths(x)]
))
docs = []
for vaspath in vaspaths:
    with monty.os.cd(vaspath):
        docs.append(drone.assimilate())
```

- Improve communication between analysists and simulators
- Seamlessly integrate novel experiments into existing analysis
- Save Time!

Accessible

How Atomate Enables all Group Members to Access Available Data

Extracting intermediate structures from relaxations

Pseudocode for extracting unstable perovskite structures to files for Graph Network Training

```
for doc in docs:
    for k,struct in doc.calcs_reversed.structure.items():
        write_struct_file(name=k, content=struct)
```

Interoperable

How Atomate Enables Standardized Data Extraction and Snapshot Reporting

Interoperability

- Store experiments
  - machine readable documents >>> inscrutable directory trees
  - adopt legacy data schemas and evolve over time
- Ensure Project Continuity across graduations
- Enable export and import of documents and/or workflows
  - Define experiments abstractly
  - Send specifications to reviewers

generating data with atomate

1

```
# Query for a Ti3AlC2 MAX structure
with MPRester(key) as m:
    data = m.summary.search(material_ids=["mp-3747"])

# Verify structure object
struct = data[0].structure
```

2

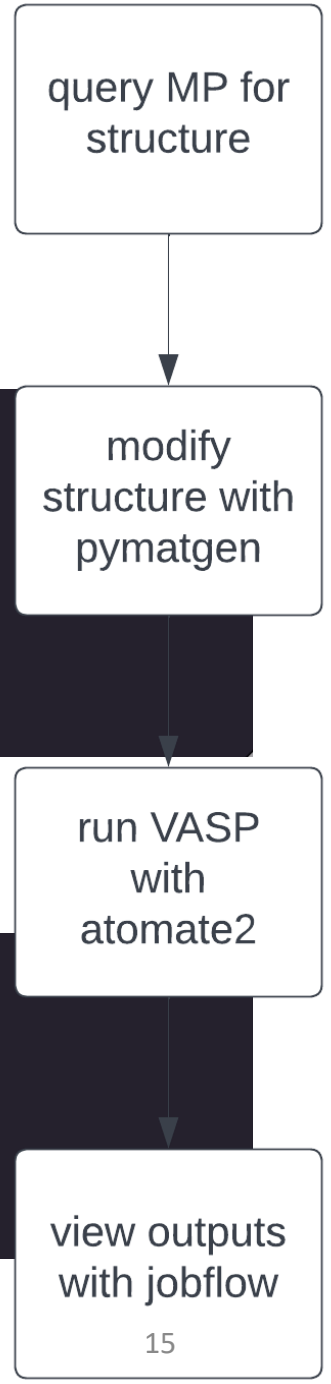
```
# Modify pymatgen structure object
map = {'C':'N', 'Ti':'Mo'}
struct.replace_species(map)
```

3

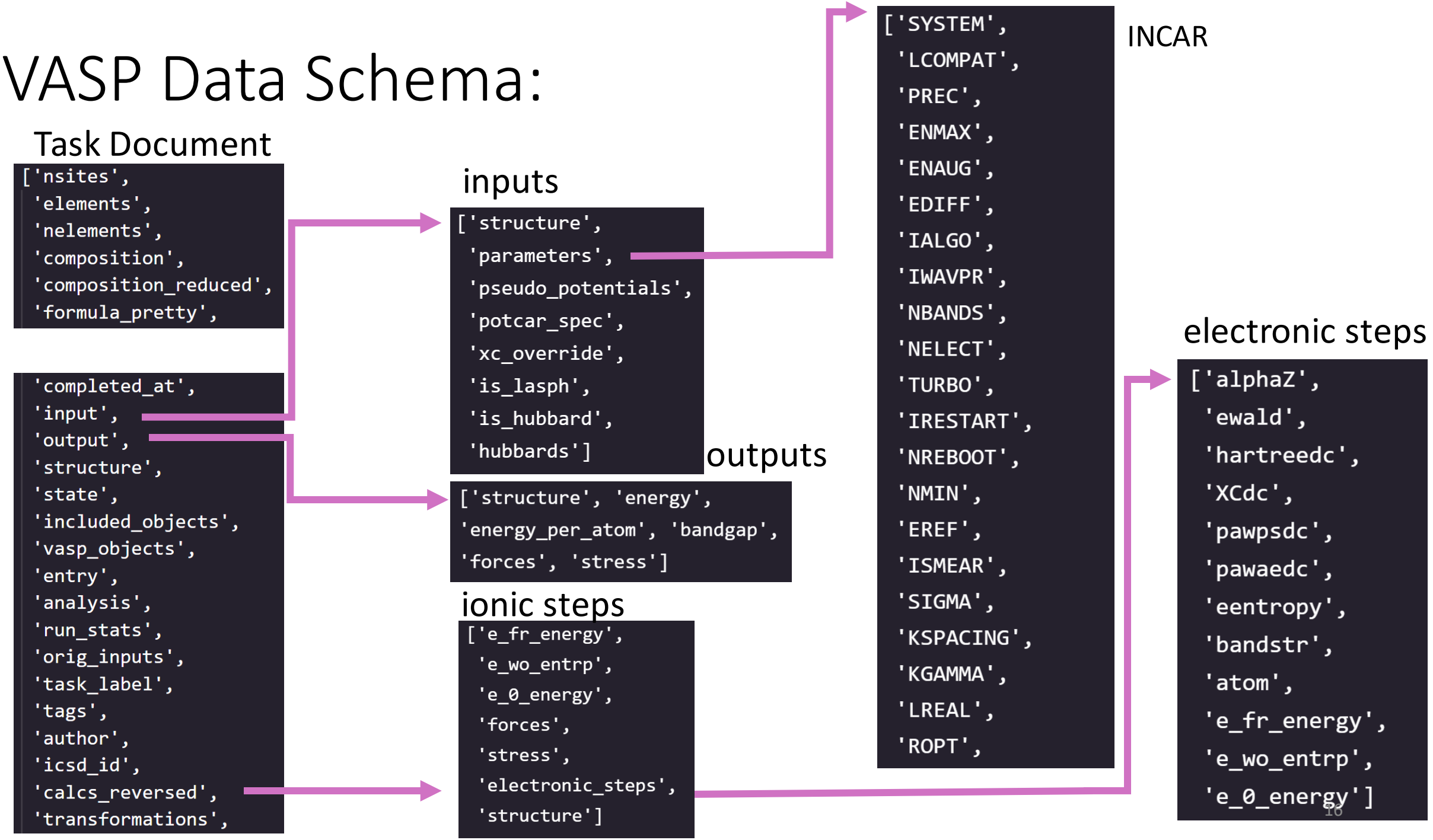
```
# Create atomate2 double relax workflow
relax_job = DoubleRelaxMaker().make(struct)

# Submit the workflow with fireworks
wf = flow_to_workflow(relax_job)
lpad = LaunchPad.auto_load()
lpad.add_wf(wf)
```

4

```
# View output for test MAX phase
store.connect()
result = store.query_one(
    {"output.formula_pretty":"AlMo3N2"},
    properties=["output"]
)
```

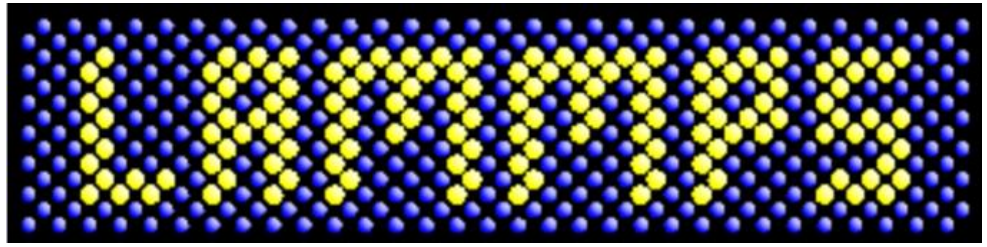


VASP Data Schema:



Extending Implementation

- The Task Document is a subclassed of a python MSONable object
- The Task Document is populated using a VASP Drone
- Both MSONable schemas and Drones can be written for any Simulation Code with minimal effort



Reusable

How Atomate Necessitates Metadata Collection.

Everything Is Stored!

- Never overwrite data
- Never perform redundant operations
- Never forget the context of an experiment
- Embed metadata in database documents programmatically
- Issue queries in a consistent, understandable, imperative way
- Flexibly publish snapshots of data corresponding to state of database at time of publications