In [1]:
```python
from nomad_parser_vasp.parsers.xml_parser import VasprunXMLParser
from nomad.datamodel import EntryArchive
from nomad.normalizing.metainfo import MetainfoNormalizer
from nomad import utils
logger = utils.get_logger(__name__)
```

```
/home/jfrudzinski/miniconda3/envs/vasp-plugin-new/lib/python3.9/site-packages/tqdm/auto.
py:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See http
s://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
Schema is deprecated, use plugins. ()
```

In [2]:
```python
path = './data/'
p = VasprunXMLParser()
a = EntryArchive()
p.parse(path + 'vasprun.xml.relax', a, logger=logger)

MetainfoNormalizer().normalize(archive=a)
```

```
The used property is not defined in the FAIRmat taxonomy (https://fairmat-nfdi.github.i
o/fairmat-taxonomy/). You can contribute there if you want to extend the list of availab
le materials properties. ()
The used property is not defined in the FAIRmat taxonomy (https://fairmat-nfdi.github.i
o/fairmat-taxonomy/). You can contribute there if you want to extend the list of availab
le materials properties. ()
The used property is not defined in the FAIRmat taxonomy (https://fairmat-nfdi.github.i
o/fairmat-taxonomy/). You can contribute there if you want to extend the list of availab
le materials properties. ()
The used property is not defined in the FAIRmat taxonomy (https://fairmat-nfdi.github.i
o/fairmat-taxonomy/). You can contribute there if you want to extend the list of availab
le materials properties. ()
Length of `AtomicCell.positions` does not coincide with the length of the `AtomicCell.at
oms_state`. (normalizer=MetainfoNormalizer)
Could not extract the geometric space information from ASE Atoms object. (normalizer=Met
ainfoNormalizer)
could not normalize section (normalizer=MetainfoNormalizer, section=DFT, exc_info=max()
arg is an empty sequence)
could not normalize section (normalizer=MetainfoNormalizer, section=EntryArchive, exc_in
fo='NoneType' object has no attribute 'entry_type')
```

No energy parsing

In [4]:
```python
a.m_to_dict()
```

Out[4]:
```
{'data': {'m_def': 'nomad_simulations.schema_packages.general.Simulation',
  'program': {'name': 'VASP', 'version': '5.3.2'},
  'model_system': [{'datetime': '2024-08-15T16:03:02.786197+00:00',
    'branch_depth': 0,
    'cell': [{'m_def': 'nomad_simulations.schema_packages.model_system.AtomicCell',
      'name': 'AtomicCell',
      'positions': [[0.0, 0.0, 0.0], [0.500001, 0.500001, 0.500001]],
      'periodic_boundary_conditions': [False, False, False]}]}],
  'model_method': [{'m_def': 'nomad_simulations.schema_packages.model_method.DFT',
    'xc_functionals': [{'libxc_name': 'PE'}]}]},
 'results': {'eln': {'sections': ['ModelSystem']}}}
```

Parse double counting energies

Case 1: Don't include UnknownEnergy in parsing

Expected Results: UnknownEnergy is added to contribution list by the normalizer

In [3]:
```
total_energy = a.data.outputs[0].total_energy[0]
print(total_energy.name)
print(total_energy.value)
print(total_energy.contributions)
```

```
TotalEnergy
-1.1442321664199474e-18 joule
[HartreeDCEnergy:HartreeDCEnergy(name, type, is_derived, variables, value), XCdcEnergy:X
CdcEnergy(name, type, is_derived, variables, value), UnknownEnergy:UnknownEnergy(name, i
s_derived, value)]
```

In [4]:
```
hartreedc = total_energy.contributions[0]
print(hartreedc.name)
print(hartreedc.type)
print(hartreedc.value)
```

```
HartreeDCEnergy
double_counting
-6.432015131607956e-17 joule
```

In [5]:
```
xcdc = total_energy.contributions[1]
print(xcdc.name)
print(xcdc.type)
print(xcdc.value)
```

```
XCdcEnergy
double_counting
-7.660195079365588e-18 joule
```

In [6]:
```
unknown = total_energy.contributions[2]
print(unknown.name)
print(unknown.value)
```

```
UnknownEnergy
7.08361142290252e-17 joule
```

In [7]:
```
total_energy.value - hartreedc.value - xcdc.value
```

Out[7]: $7.08361142290252 \times 10^{-17}$ joule

Case 2: Add UnknownEnergy to contribution list in the parser but without a value

Expected Results: UnknownEnergy value is calculated by the normalizer and placed into this section (same result as Case 1).

In [3]:
```
total_energy = a.data.outputs[0].total_energy[0]
print(total_energy.name)
print(total_energy.value)
print(total_energy.contributions)
```

```
TotalEnergy
-1.1442321664199474e-18 joule
[HartreeDCEnergy:HartreeDCEnergy(name, type, is_derived, variables, value), XCdcEnergy:X
CdcEnergy(name, type, is_derived, variables, value), UnknownEnergy:UnknownEnergy(name, i
s_derived, variables, value)]
```

In [4]:
```
hartreedc = total_energy.contributions[0]
print(hartreedc.name)
print(hartreedc.type)
print(hartreedc.value)
```

```
HartreeDCEnergy
double_counting
-6.432015131607956e-17 joule
```

In [5]:
```
xcdc = total_energy.contributions[1]
print(xcdc.name)
print(xcdc.type)
print(xcdc.value)
```

```
XCdcEnergy
double_counting
-7.660195079365588e-18 joule
```

In [6]:
```
unknown = total_energy.contributions[2]
print(unknown.name)
print(unknown.value)
```

```
UnknownEnergy
7.08361142290252e-17 joule
```

In [7]:
```
total_energy.value - hartreedc.value - xcdc.value
```

Out[7]: $7.08361142290252 \times 10^{-17}$ joule

Case 3: Add UnknownEnergy to contribution list in the parser with a value

Expected Results: normalizer does not change the value of UnknownEnergy (for testing purposes we subtract double the hartreedc value).

In [3]:
```
total_energy = a.data.outputs[0].total_energy[0]
print(total_energy.name)
print(total_energy.value)
print(total_energy.contributions)
```

```
TotalEnergy
-1.1442321664199474e-18 joule
[HartreeDCEnergy:HartreeDCEnergy(name, type, is_derived, variables, value), XCdcEnergy:X
CdcEnergy(name, type, is_derived, variables, value), UnknownEnergy:UnknownEnergy(name, i
s_derived, variables, value)]
```

In [4]:
```
hartreedc = total_energy.contributions[0]
print(hartreedc.name)
print(hartreedc.type)
print(hartreedc.value)
```

```
HartreeDCEnergy
double_counting
-6.432015131607956e-17 joule
```

In [5]:
```
xcdc = total_energy.contributions[1]
print(xcdc.name)
print(xcdc.type)
print(xcdc.value)
```

```
XCdcEnergy
double_counting
-7.660195079365588e-18 joule
```

In [6]:
```
unknown = total_energy.contributions[2]
print(unknown.name)
print(unknown.value)
```

```
UnknownEnergy
1.3515626554510475e-16 joule
```

In [ ]:
```

```