Prototype declarative analysis interface using uproot and awkward-array

Mason Proffitt
(University of Washington)

September 12, 2019

About me (and my group)

- **Physics PhD student at the University of Washington**
- **Advisor: Gordon Watts, postdoc: Emma Torró**
- **Currently based in Seattle, but moving (back) to CERN in 11 days**
- **Projects:**
  - IRIS-HEP Analysis Systems (this work)
  - ATLAS analysis (long-lived particle search with displaced jets)
  - MATHUSLA (proposed dedicated long-lived particle detector)

Introduction

- **The challenge that we're trying to address at UW:**
  - Accessing selections of data for analysis in a columnar format
  - In general, the data could be anywhere:
    - Local disk, cluster, web, grid, etc.
  - and in any format:
    - xAOD, flat ROOT ntuple, HDF5, Parquet, etc.
- **Main focus so far:**
  - Developing a prototype abstract interface for specifying a selection
  - Implementing some backends that actually return this selection for some popular HEP data formats

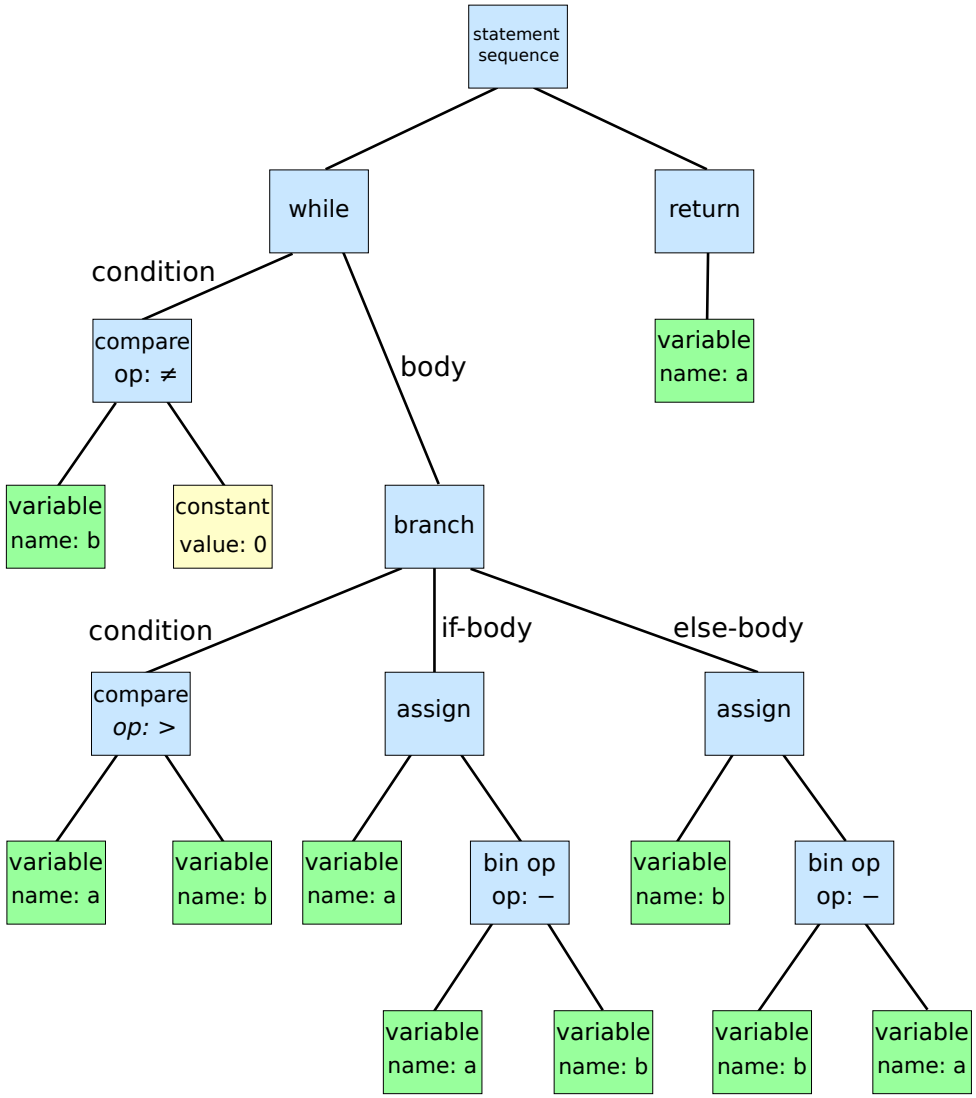
Data selection interface

- **Interface for selecting data is query-based, modeled after LINQ from C#**
- **Queries are written within Python**
- **Example from prototype:**

```
data_source.Select("lambda e: {'eventNumber': e.eventNumber,\
                               'jet_pT': e.CalibJet_pT}")\
           .Where("lambda e: (e.jet_pT > 1000).Count() > 0"))
```

Abstract syntax trees

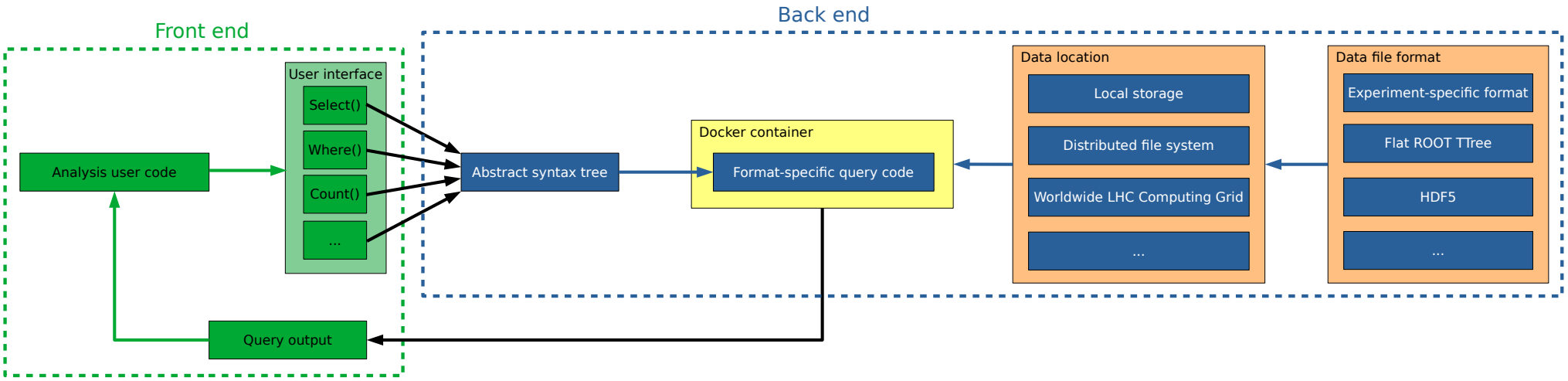
- **Query is parsed as an abstract syntax tree (AST) by the built-in Python module `ast`**
- **An AST is composed of nodes and their relationships**



https://commons.wikimedia.org/wiki/File:Abstract_syntax_tree_for_Euclidean_algorithm.svg

Backend code generation

- **Each node of the AST is translated into a representation appropriate for the data source by one of several backends**
- **This generates new code to actually extract the selection from data**



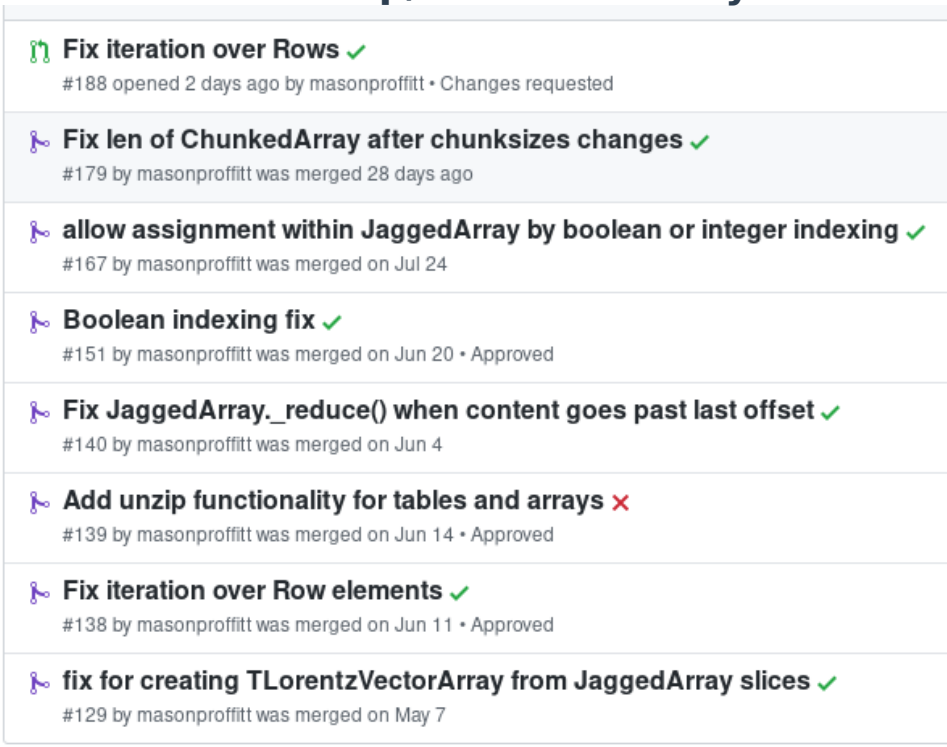
Backends

- **Current backends:**
  - xAOD (C++) (see Gordon's talk)
  - flat ROOT ntuples:
    - uproot + awkward-array (Python)
    - RDataFrame (C++)
- **I've focused on creating the uproot/awkward-array backend**
  - AST translator for this backend:
    - https://github.com/masonproffitt/BDTTrainingAnalysisLanguage/blob/master/pythonarraylib/python_array_ast_visitor.py
  - Query usage examples:
    - https://github.com/masonproffitt/BDTTrainingAnalysisLanguage/blob/master/examples/array_examples.ipynb
    - https://github.com/masonproffitt/BDTTrainingAnalysisLanguage/blob/master/examples/p4_test.ipynb

Remarks on uproot/awkward backend

- Going from a Python AST to Python code is (not surprisingly) very simple
- Columnar data extraction and selection functionality is built in to uproot/awkward-array
- Often ends up just being a stress test for awkward-array, running into spots where awkward breaks
- Lot of issues and pull requests!

Pull requests I've made on scikit-hep/awkward-array:



https://github.com/scikit-hep/awkward-array/pulls?utf8=%E2%9C%93&q=author%3Amasonproffitt

Difficulties

- **Inconsistencies between numpy and awkward**
  - Methods existing in one but missing in other
  - Both having methods with the same name but different default behavior or different parameters
  - Some inconsistencies are necessitated by different data structures, some are design choices
  - This should improve in awkward-1.0
- **Queries quickly get very ugly when you need to make detailed analysis cuts**
  - Often easier to make the selection simpler and then manipulate results manually in Python
  - It's a bit of an art to get the right balance between filtering at the query level vs. analysis code following the query execution

Conclusion

- **Basic query functionality has been implemented in the uproot/awkward-array backend, running on flat ntuples from my ATLAS analysis**
  - Doing a columnar analysis partially using this backend
  - Working on putting more of the workload into the query
- **Provides a good test of uproot/awkward-array, and shows where there are weaknesses in these**