PyDecay/GraphPhys: A Software Framework for Reducing High-Energy Physics Analysis Preparation Time

Jesse Dunietz (doonitz@mit.edu)
MIT, BaBar Collaboration (SLAC)

Purpose

Experiments such as BaBar provide high-energy physics (HEP) analysts with the data they need to make new fundamental physics discoveries. Currently, an analyst’s pre-analysis work involves a tedious simulation and data reduction phase. The purpose of PyDecay is to reduce this pre-analysis period from months to days. It is a software framework that provides various computer representations of particle decays, making it almost trivial to build tools to automate pre-analysis tasks.

The BaBar Experiment at SLAC

The BaBar experiment was designed to seek explanations for the asymmetry in the universe’s matter/antimatter balance. The basic structure of the experiment is as follows:

- The SLAC linear accelerator is used to create electron and positron beams, which are diverted to the BaBar detector, a storage ring. Eventually, the electrons and positrons collide within the BaBar detector, forming high-mass particles.

- Some high-mass particles decay into lower-mass particles, whose presence and properties are picked up by detectors.

- Analysts run software to reconstruct the chains of particle decays from the detector data.

- The task of experimental analysis is to configure and run the reconstruction software, to compare the results with theoretical predictions, and to draw conclusions from the comparison.

Difficulties of HEP Analysis

Typically, an analyst chooses a particular decay sequence to study, obtains theoretical predictions about this decay from computer simulations, then configures the ‘reconstruction’ software to search the detector data for instances of the decay.

Some of the subprocesses in this workflow and their problems:

- Choose decay of interest
- Obtain predictions from computer simulation
- Configure and run reconstruction software
- Write configuration file
- Write reconstruction configuration script
- Verify reconstruction configuration script
- Check system and/or software for correctness
- Check whether others have investigated it

Each task should take at most several hours. However, there are currently no tools at all for those in red. The tasks in yellow are doable but difficult, because:

- There is no easily searchable centralized database of previously studied decays
- The simulation and reconstruction configuration files have unwieldy, unstructured, and unrelated formats

Because of these issues, pre-analysis tasks can take up to two months.

PyDecay/GraphPhys

PyDecay was designed to provide a comprehensive solution to these problems. It provides a suite of representations for particle decay chains and the means to convert between them. Each of these decay representations is designed to target some subset of the problematic tasks mentioned above.

PyDecay is not itself a software tool; it is a framework for building software tools. The framework was designed to be as modular as possible: new representations, database implementations, and so on can easily be added in the process of building a new tool. All modules were implemented in Python.

PyDecay As a Solution: Modules for Manipulating Decay Representations

GraphPhys

GraphPhys is a language for describing particle decay chains and their attributes. It is designed for:

- Clarity for reading/writing by humans
- Ease of computer parsing
- Flexibility

GraphPhys simplifies the pre-analysis workflow by:

- Providing a common input format for various configuration files, and for use in other tools
- Making configuration files easier to write
- Making existing configuration files easier to read

The language is described by an abstract grammar. PyDecay includes a GraphPhys parser and uses GraphPhys as its primary input mechanism, but the language is independent of PyDecay in the sense that any software package could implement a parser.

Visualizations

PyDecay includes a module for producing several kinds of decay visualizations. The built-in visualization types are:

- Feynman-diagram-like images
- GraphViz Dot diagrams

Visualization simplifies the pre-analysis workflow by:

- Allowing a visual check on the correctness of configuration files
- Helping analysts consider decays more thoroughly by allowing them to think graphically

Database Module

The database module exists to store and provide generic information about particle and decay types.

The module allows for different database implementations. The default implementation is a full-fledged relational database, and also allows storing and accessing information about previously entered decay descriptions.

Core Internal Decay Representation

A common decay, represented as a decay tree:

\[ D \rightarrow K^+ K^- \pi^+ \]

At the core of all PyDecay modules is a tree-based representation of decay sequence. Each particle is a node in a decay tree, and its child nodes are the particles it decays into. Each particle and each decay can have associated properties, which are simply name/value pairs.

Each node may have multiple child sets, each representing a probabilistically possible decay.

Future Potential & Ongoing Efforts

The PyDecay package has great potential for expanded use:

- PyDecay could become a standard tool, universal decay description language.
- The database component could be the backend for a public database of the Particle Data Group’s tabulated particle data.
- The simplified analysis tools will simplify analysis in BaBar and similar projects, e.g., LHC experiments and SuperB.
- The tools can be used for education and outreach.

My mentor and I are engaged in discussions with other groups about adopting PyDecay and GraphPhys for these purposes.

We are continuing to improve and maintain PyDecay. The project is open source; the code can be freely downloaded from the PyDecay Google Code repository: http://code.google.com/p/pydecay/.

Tools Built with PyDecay

My mentor and I have built several command-line tools upon the PyDecay framework that showcase the framework’s abilities:

- Decay simulator: a minimal proof-of-concept Monte Carlo (i.e., randomized) decay simulator that uses GraphPhys as its input format
- Kinematics checker: uses database information to determine whether a proposed decay violates conservation laws (no prior software does this)
- Branching fraction calculator: computes the total probability of occurrence for an entire decay tree, using the database for probabilities of individual decays (no prior software does this)
- Configuration generators: take GraphPhys decay descriptions and output configuration files in the existing formats for simulation and reconstruction software
- Visualizer: simply outputs a visualization graphic for a given decay

Contributions

In creating PyDecay, I have:

- Defined a simple, easy-to-use decay description language on which decay descriptions can be standardized
- Implemented a framework for representations of decays that can be used to store, extract, and display decay information
- Demonstrated the usefulness of the framework by using it to implement tools that make formerly arduous tasks far easier

I believe that this framework, particularly the GraphPhys language, will prove immensely useful to high-energy physicists worldwide.

Acknowledgements

Many thanks to my mentor, Matt Beilis (mbeilis@stanford.edu), whose guidance was invaluable throughout the project.

I also thank the following institutions for their support:

Office of Science
U.S. Department of Energy
Fermi National Accelerator Laboratory
SLAC National Accelerator Laboratory

PyDecay/GraphPhys: A Software Framework for Reducing High-Energy Physics Analysis Preparation Time

Jesse Dunietz (doonitz@mit.edu)
MIT, BaBar Collaboration (SLAC)