Project Proposal: Julia Interfaces to HepMC3 Event Record Library

Divyansh Goyal
{ Guru Gobind Singh Indraprastha University **}**
Mentors :Graeme Stewart **{** CERN **}** | Mateusz Fila { CERN **}**

Abstract:

High Energy Physics (HEP) experiments and simulations critically depend on standardized event record formats like HepMC3 for data interoperability. This project proposes to develop HepMC3.jl, a native Julia package providing comprehensive interfaces to the HepMC3 library. The key goals are: (1) Implementing robust reading of HepMC3 event files, prioritizing the ASCII format, and enabling access to event data structures, particle properties, and run information. (2) Facilitating full event record navigation, including traversal between parent and child particles via vertices. (3) Supporting the creation of new HepMC3 events and the modification of existing data structures, along with their re-serialisation to file (initially ASCII). (4) Ensuring comprehensive documentation, illustrative examples, and registering HepMC3.jl in the Julia General Registry to integrate it into the JuliaHEP ecosystem. This package will empower Julia users with direct, efficient access to HepMC3 data.

Meta: Start - **05th June, 2025** | Time Commitment **- > 40hrs/week**
Parallel commitments - **Semester Examination (Recurring prior mid-terms have prepared us well)**

1. Motivation & Context:

In high-energy physics experiments, simulating physics events is essential for comparing theoretical predictions with experimental observations. The `HepMC3` event record library plays a pivotal role in this process, standardizing the output from physics event generators for use by downstream detector simulation and analysis codes. With the increasing adoption of `Julia` in HEP for its blend of interactive ease and compiled speed, there's a growing need for a robust Julia ecosystem. This project addresses a key gap by aiming to develop HepMC3.jl, providing native Julia interfaces to read, manipulate, and write HepMC3 event records, thereby enhancing Julia's capabilities for HEP software development.

2. Project Goals & Technical Deliverables:

The project will develop HepMC3.jl by wrapping the core C++ HepMC3 library, focusing on usability, performance, and integration within the JuliaHEP ecosystem.

- **A. Foundational Wrappers & ASCII File Reading (`HepMC3.jl` via `CxxWrap.jl`)**:
  - **Goal:** Establish core C++ bindings and enable robust parsing of `HepMC3 ASCII` event files.
  - **Deliverables:**
    - Functional `CxxWrap/WrapIt!` generated Julia wrappers for essential HepMC3 classes (e.g., `GenEvent, GenParticle, GenVertex, GenRunInfo, ReaderAscii`).
    - Implementation of functions to read HepMC3 ASCII files (`.hepmc3`), populating Julia-accessible HepMC3 objects.
    - Basic validation checks and error handling during the reading process.
    - Unit tests for ASCII reading functionalities.
    - Memory-efficient data handling for parsed event information.

- **B. Event Data Access & Navigation**:
  - **Goal:** Provide comprehensive and intuitive access to event structures, particle properties, and their relationships.
  - **Deliverables:**
    - Type-safe Julia interfaces for accessing particle properties (`PDG ID, four-momentum, mass, charge, status`) and `vertex` information.
    - Functions for navigating event structures:
      - Accessing production/decay vertices for particles.
      - Retrieving incoming/outgoing particles for a given vertex.
      - Implementing forward (decay products) and backward (parent particles) traversal along decay chains.
      - Tools for identifying sibling particles.
    - Comprehensive docstrings and example scripts for data access and navigation.
    - Unit tests covering all access and navigation methods.
- **C. Run Information, Event Modification & Creation**:
  - **Goal:** Enable management of run-level metadata, modification of existing events, and construction of new events from scratch in Julia.
  - **Deliverables:**
    - Interface for accessing and potentially configuring run information (e.g., `cross-sections, generator settings, event weights, units`) via `` `GenRunInfo` ``.
    - Methods for modifying particle properties (e.g., `four-momentum`, `status`) within an existing `` `GenEvent` ``.
    - Tools for event structure manipulation (e.g., adding/removing particles/vertices, modifying decay chains), with optional physics consistency checks (e.g., energy-momentum conservation).
    - An `EventBuilder`-like interface for step-by-step programmatic creation of new `GenEvent` objects in Julia, managing particle and vertex relationships.
    - Validation tools to ensure physics consistency during new event construction.
    - Unit tests for run info management, event modification, and creation.
- **D. Re-serialization, Documentation, Testing & Ecosystem Integration**:
  - **Goal:** Complete the read-write cycle, ensure the package is robust, well-documented, and accessible to the JuliaHEP community.
  - **Deliverables:**
    - Implementation of functions to serialize `GenEvent` objects back to HepMC3 ASCII format (via `WriterAscii`), maintaining numerical precision.
    - Buffered writing system for efficient handling of large-scale event processing.
    - Validation of output compatibility with standard HepMC3 tools.
    - Comprehensive `` `Documenter.jl` ``-based documentation: installation guide, API reference, tutorials, and usage examples covering all major features.
    - Registration of HepMC3.jl in the Julia General Registry.
    - Comprehensive `Test.jl` suite: High coverage (>80%) of all features, including integration tests for fitting workflows.
    - CI Integration: GitHub Actions workflow executing the full test suite

(Stretch Goal):

- Extension of serialisation to support the ROOT format..

4. Timeline (3 Months / 13 Weeks):

- **W1-2:** Project setup. Deep dive into `CxxWrap, WrapIt!`, and HepMC3 C++ API. Implement initial wrappers for core HepMC3 objects. Begin ASCII file reading.
- **W3-4:** Focus on accessing event data structures and particle properties. Develop initial unit tests for reading and basic data access.
- **W5-6:** Implement event navigation (parent/child, vertices) and access to run information. Expand unit test suite.
- **W7-8:** Develop functionalities for updating `HepMC3` data structures and creating new events. Start documentation structure with `Documenter.jl`.
- **W9-10:** Implement re-serialisation of events to ASCII format. Write core documentation content and develop usage examples.
- **W11-12:** Finalize documentation, examples, and comprehensive unit testing. Code polish and API refinement. Prepare for package registration. Set up CI.
- **W13:** Prepare release candidate. Register HepMC3.jl in Julia General Registry.

(Stretch Goal)
If time permits and prior goals are met, begin investigation/implementation of ROOT serialisation .

5. Contributor Background:

I am a sophomore at Indraprastha University, pursuing a degree in Artificial Intelligence and Machine Learning under the Computational Biology domain. As a GSoC 2024 contributor with JuliaHealth, I developed **MedEye3d.jl**, a 3D medical imaging and visualization tool, and supported the Julia ecosystem with packages for DICOM/NIFTI handling and ITK-based registration. Building on this experience and my work on **HepMC3-dev.jl**, I bring cross-domain expertise in scientific software, aiming to strengthen Julia's HEP tooling and data analysis pipelines.