NuHepMC: Simplifying Generator Comparisons

Steven Gardiner, Joshua Isaacson, Luke Pickering

21 June 2023

Why a common event format?

- Currently each generator, experimental simulation, and analysis framework maintains their own format
- This requires converters between formats increasing maintenance requirements
- Maintenance may become unfeasible in the future, need to unify to a common format
- Needed for any eventual interoperability between generators, (e.g. hard interaction with GENIE and cascade with Achilles or vice versa)

What is the HepMC3 format?

- Widely used format developed for the LHC and recently extended to handle heavy ions and EIC.
  - Paper: Comput. Phys. Commun. 260 (2021) 107310
  - Code: https://gitlab.cern.ch/hepmc/HepMC3
- Stores metadata for the generation of the events within the file known as run information. This is known at the start of job and can include:
  - Arbitrary runtime configuration (e.g. all information to reproduce the events)
  - The types of weights associated with each event (useful for reweighting)
- Each event contains its own metadata along with a list of particles and vertices that form a graph structure and requires:
  - No dangling particles or vertices, no cyclic relations
  - All vertices should have at least one outgoing particle and all but the root vertex should have at least one incoming particle

What is the HepMC3 format? (cont)

- Vertices connect sets of incoming and outgoing particles
  - Hold information about 4-position and a status code to describe the type of vertex
  - Used to define mother / daughter relationships between the various particles
- Particles provide information on 4-momentum, PDG PID code, and status codes
- The HepMC3 standard provides a means to supply arbitrary attributes in addition to the builtin ones to any object
  - Each attribute comes with a name to read from / write to
  - Built-in support for strings, integers, floating point numbers, and vectors
- Abstracts away the file storage type from the reading / writing of data
  - Can deduce the reader at runtime based on file
  - Currently supports ASCII, compressed ASCII, ROOT, and Protobuf I/O

The NuHepMC Proposed Standard Overview

- Goal: Define common standards for representing neutrino scattering events and reduce the maintenance burden
- How: Leverage a well-supported LHC framework like HepMC3
- Problems to address:
  - Reproducibility
  - Labeling interaction modes
  - Units to be used
  - Required metadata for analyses

Structure of the specification

- Specification defines 4 Components:
  - G: Generator Run Metadata
  - E: Event Metadata
  - V: Vertex Information
  - P: Particle Information
- Specification defines 3 categories (RCS):
  - Requirements
  - Conventions
  - Suggestions

- The options are enumerated as <Component>.<Category>.<Index>
  - Example: 6th convention for event information is E.C.6

https://github.com/NuHepMC/Spec
https://github.com/NuHepMC/ReferenceImplementation

G.C.1 Signalling Followed Conventions

To signal to a consumer that an implementation follows a named convention from this specification, a `HepMC3::VectorStringAttribute` should be added to the `HepMC3::GenRunInfo` instance named "NuHepMC.Conventions" containing the names of the conventions adhered to.

Process and Particle Status code Guidelines

Identifier	Process
100-199	Coherent Nuclear scattering
200-299	Quasielastic
300-399	Meson Exchange Current
400-499	Resonance production
500-599	Shallow inelastic scattering
600-699	Deep inelastic scattering
700-999	Other process types

Charged current (CC) processes should have identifiers in the `X00–X49` block and neutral current (NC) in the `X50–X99` block.

Status Code	Description	Usage
0	Not defined	Not meaningful
1	Undecayed physical particle	Recommended for all cases
2	Decayed physical particle	Recommended for all cases
3	Documentation line	Often used to indicate in/out particles in hard process
4	Incoming beam particle	Recommneded for all cases
5-10	Reserved for future standards	Should not be used
11	Target particle	Recommended for all cases
12-20	Reserved for future standards	Should not be used
21-200	Generator-dependent	For generator usage
201-	Simulation dependent	For simulation software usage

Cross-section information

- **E.C.5:** Cross section values should be stored in picobarns

- **E.C.2:** Event attribute ("TotXS") stores total cross section for the beam particle to interact
- **E.C.3:** Event attribute ("ProcXS") stores the total cross section for the selected ProcID

- **G.C.4:** Store the flux-averaged total cross section in the run metadata (if known at start)
  - Straightforward for simple cases (monoenergetic, flux histogram and point target)

- **E.C.4:** Store running MC estimate (and statistical uncertainty) of flux-averaged total cross section in each event
  - Likely necessary for complex fluxes and/or geometries

Credit: Steven Gardiner

Draft GENIE interface (1)

- Unofficial test branch for now, but briefly discussed with other authors
  - Blame Steven G for whatever you don't like

- Adds HepMC3 library as an optional GENIE build dependency
  - **./configure --enable-hepmc3**
  - Similar to interface with external codes (INCL++, Geant4) for new FSIs in v3.2.0

```
class HepMC3Converter {

public:

  HepMC3Converter(void);

  std::shared_ptr< HepMC3::GenEvent > ConvertToHepMC3(
    const genie::EventRecord& gevrec );

  std::shared_ptr< genie::EventRecord > RetrieveGHEP(
    const HepMC3::GenEvent& evt );
```

- **genie::HepMC3Converter**
  - Bi-directional translations between genie::EventRecord objects and NuHepMC-compliant HepMC3::GenEvent objects
  - Extra GENIE event record contents stored as attributes ("GENIE.ZZZ")

Draft GENIE interface (2)

- Output in HepMC3 text-based format provided by **genie::HepMC3NtpWriter**

- Refactored gevgen command-line program
  - **gevgen -o my_ghep_events.root,ghep,my_hepmc3_events.txt,hepmc** will write equivalent output files in both formats simultaneously

- Running estimate of flux-averaged total cross section included in output (E.C.4)

- Encountered a few surprises

- **Example:** Some mother/daughter pairs do not have the same 4-position. Considering adjustments to GENIE conventions.

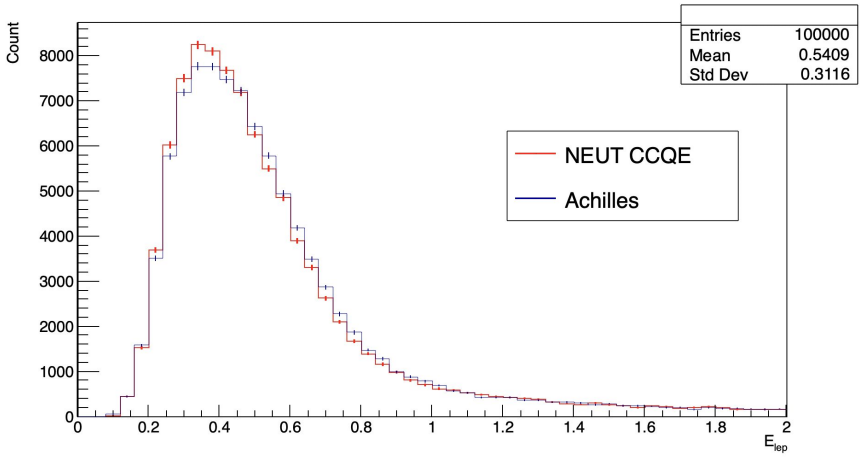
```
class HepMC3NtpWriter : public NtpWriterI {

public:

  HepMC3NtpWriter();
  virtual ~HepMC3NtpWriter();

  ///< initialize the ntuple writer
  virtual void Initialize() override;

  ///< add event
  virtual void AddEventRecord( int ievent, const EventRecord* ev_rec ) override;

  ///< save the event tree
  virtual void Save() override;
```

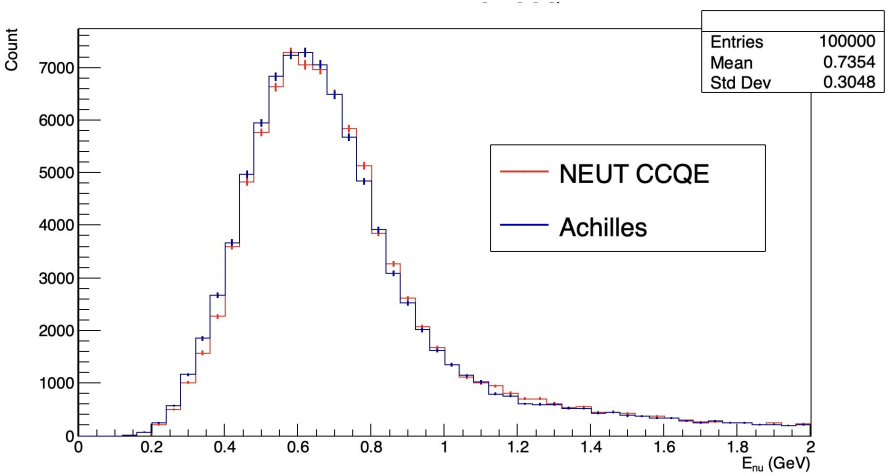
NuHepMC in Achilles

```
void NuHepMC3Writer::WriteHeader(const std::string &filename) {
    // Setup generator information
    spdlog::trace("Writing Header");
    auto run = std::make_shared<HepMC3::GenRunInfo>();
    run->add_attribute("NuHepMC.Version.Major",
                       std::make_shared<HepMC3::IntAttribute>(version[0]));
    run->add_attribute("NuHepMC.Version.Minor",
                       std::make_shared<HepMC3::IntAttribute>(version[1]));
    run->add_attribute("NuHepMC.Version.Patch",
                       std::make_shared<HepMC3::IntAttribute>(version[2]));

    struct HepMC3::GenRunInfo::ToolInfo generator={std::string("Achilles"),
                                                   std::string(ACHILLES_VERSION),
                                                   std::string("Neutrino event generator")};
    run->tools().push_back(generator);
    run->add_attribute("Achilles.RunCard",
                       std::make_shared<HepMC3::StringAttribute>(filename));
    run->set_weight_names({"CV"});

    // Add all possible processes
    // TODO Look up available processes and add bibtex info
    std::vector<int> proc_ids{101};
    run->add_attribute("NuHepMC.ProcessIDs",
                       std::make_shared<HepMC3::VectorIntAttribute>(proc_ids));
    run->add_attribute("NuHepMC.ProcessInfo[101].Name",
                       std::make_shared<HepMC3::StringAttribute>("SpectralQE"));
    run->add_attribute("NuHepMC.ProcessInfo[101].Description",
                       std::make_shared<HepMC3::StringAttribute>("Spectral function Quasielastic"));
    run->add_attribute("NuHepMC.ProcessInfo[101].Bibtex",
                       std::make_shared<HepMC3::StringAttribute>("Rocco:xxxx"));

    // List all possible vertex status codes
    // TODO Make this a conversion from enum of the EventHistory class?
    std::vector<int> vertex_ids{1};
    run->add_attribute("NuHepMC.VertexStatusIDs",
                       std::make_shared<HepMC3::VectorIntAttribute>(vertex_ids));
    run->add_attribute("NuHepMC.VertexStatusInfo[1].Name",
                       std::make_shared<HepMC3::StringAttribute>("Primary"));
    run->add_attribute("NuHepMC.VertexStatusInfo[1].Description",
                       std::make_shared<HepMC3::StringAttribute>("The main hard interaction"));
```

- Validated against NuHepMC Validator
- Many parts still hard coded
- Publically available on development branch: https://github.com/AchillesGen/Achilles/tree/dev
- Using with NUISANCE to validate a full analysis pipeline

NuHepMC in Nuisance





- Available in NUISANCE:
  - Example flattening events to root:
    - `nuisflat -i NuHepMC:t2k.numu.hepmc -o t2k.flat.root`

Why should you support NuHepMC?

- Provides common format for all event generators allowing for detailed comparisons
- Standardizes event weights to make reweighting easier
- First step towards interoperability between event generators (e.g. hard scattering in GENIE and cascade in Achilles)
- Provides easier entry point for new theory calculations to be quickly compared to data
- Reduced maintenance costs since they can be shared with the rest of HEP using HepMC3 formats

How can you support NuHepMC?

- Provide feedback to the existing standard
  - What do you like?
  - What needs improvement?
  - What did we miss?
- Sign onto community paper when it is available saying you support the format
- Develop an interface in your experimental pipeline to take in NuHepMC event files (Vincent Basque has something basic for an older style within MicroBooNE already)

Conclusions

- Desire to reduce maintenance requirements for software within neutrino event generator community
- Leverage existing HEP tools used by other parts of the community (LHC, heavy ion, EIC)
- HepMC3 is a well established and supported tool
- Extend framework to include information unique to neutrino experiments (NuHepMC)
- Prototype specification implemented in Achilles, GENIE, NEUT, and NUISANCE
- In the process of finalizing a version 1.0 specification to be posted on the arxiv
- Looking for feedback and buy-in from the community

https://github.com/NuHepMC/Spec
https://github.com/NuHepMC/ReferenceImplementation

Feedback