NuHepMC: A universal event record format for neutrino event generators

Steven Gardiner, Joshua Isaacson, Luke Pickering

13 December 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
- Unifying the event format simplifies maintenance and eases comparisons between generators and cross-section data
- Also 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? (1)

- 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? (2)

- 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 proposed "NuHepMC" standard

- Goal: Define guidelines for representing neutrino physics within the HepMC3 format

- Example problems to address:
  - Reproducibility
  - Labeling interaction modes
  - Units to be used
  - Scaling event distributions to cross-section predictions

NuHepMC: A standardized event record format for neutrino event generators

S. Gardiner[a], J. Isaacson[a], L. Pickering[b]

[a]*Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510, USA*
[b]*STFC, Rutherford Appleton Laboratory, Harwell Oxford, United Kingdom*

arXiv:2310.13211

Generally well-received by the community so far, significant work remains for full adoption

Feedback very welcome!

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.

An example: dictionary of particle status codes

G.R.6 Particle Status Metadata:

The NuHepMC `HepMC3::GenRunInfo` instance must contain a `HepMC3::VectorIntAttribute` named `"NuHepMC.ParticleStatusIDs"` declaring any generator-specific status codes used. Including the standard HepMC3 codes in this list is optional, but they must not be reused to mean something different than in the HepMC3 specification. For each valid particle status, the `HepMC3::GenRunInfo` instance must also contain two other attributes giving a name and description of each:

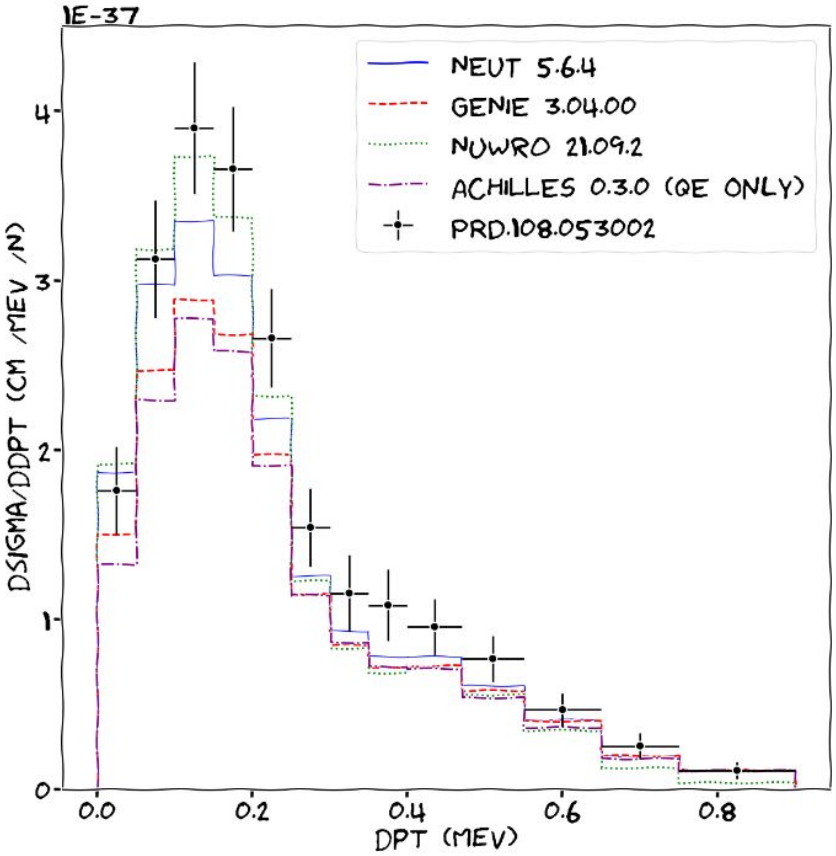
- type: `HepMC3::StringAttribute`,
  name: `"NuHepMC.ParticleStatusInfo[<ID>].Name"`
- type: `HepMC3::StringAttribute`,
  name: `"NuHepMC.ParticleStatusInfo[<ID>].Description"`

where `<ID>` enumerates all status codes present in `"NuHepMC.ParticleStatusIDs"` (see P.R.1 for more details).

An example: dictionary of particle status codes

Status Code	Description	Usage
0	Not defined	Do not use
1	Undecayed physical particle	Recommended for all cases
2	Decayed physical particle	Recommended for all cases
3	Documentation line	Used for in/out particles in the primary process
4	Incoming beam particle	Recommended for all cases
5-10	Reserved for future HepMC3 standards	Do not use
11-19	Reserved for future NuHepMC standards	Do not use
20	Target particle	Recommended for all cases
21-200	Generator-dependent	For generator usage
201-	Simulation-dependent	For simulation software usage

Use case: comparison to MicroBooNE data



- Several generator predictions compared to δp_T distribution for 1-proton knockout
  - Phys. Rev. Lett. 131, 101802 (2023)
- Produced with NUISANCE and HepMC3 outputs from each generator
  - Format conversions currently unofficial, but prototypes work well
- **No generator-specific code** used in the comparison
  - All information needed already in HepMC3 outputs

Why should you support NuHepMC?

- Provides common format for all event generators allowing for detailed comparisons
- Potential for simplified multi-generator workflows for experiments
- 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 about the existing standard to the authors
  - What do you like?
  - What needs improvement?
  - What did we miss?
- 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
- We are also exploring the possibility of having the community sign on to a revised version of the draft standard
  - If that sounds useful to you, please let us know!

Conclusions

- Standardizing neutrino generator event formats will greatly simplify their use in the neutrino community
- We leverage the established HepMC3 format used by other HEP sub-fields
- Extend framework to include information unique to neutrino experiments (NuHepMC)
- Prototype specification implemented for Achilles, GENIE, NEUT, NuWro, MARLEY, and NUISANCE
  - Planning to include GiBUU as well in the near future
- Specification posted on arXiv, considering revisions before publication
- Looking for feedback and buy-in from the community