NuHepMC: A standardized event record format for neutrino event generators

Steven Gardiner[1⋆], Joshua Isaacson[1†] and Luke Pickering[2‡]

1 Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510, USA
2 STFC, Rutherford Appleton Laboratory, Harwell Oxford, United Kingdom

⋆ gardiner@fnal.gov , † isaacson@fnal.gov , ‡ luke.pickering@stfc.ac.uk

Abstract

Simulations of neutrino interactions are playing an increasingly important role in the pursuit of high-priority measurements for the field of particle physics. A significant technical barrier for efficient development of these simulations is the lack of a standard data format for representing individual neutrino scattering events. We propose and define such a universal format, named NuHepMC, as a common standard for the output of neutrino event generators. The NuHepMC format uses data structures and concepts from the HepMC3 event record library adopted by other subfields of high-energy physics. These are supplemented with an original set of conventions for generically representing neutrino interaction physics within the HepMC3 infrastructure.

Received 2024-06-27
Accepted 2025-07-02
Published 2025-08-22
doi:10.21468/SciPostPhysCodeb.57

This publication is part of a bundle: Please cite both the article and the release you used.

DOI	Type
doi:10.21468/SciPostPhysCodeb.57	Article
doi:10.21468/SciPostPhysCodeb.57-r1.0	Codebase release

Contents

1 Introduction

Worldwide experimental efforts in high-energy physics (HEP) are placing increasing emphasis on precision measurements of neutrinos. The pursuit of these measurements creates strong demands on the quality of neutrino event generators—the software tools used to simulate neutrino scattering in the context of experimental analyses [1,2].

Recent discussions about the future of neutrino research, both at workshops focused on simulations of neutrino interactions [3] and as part of the HEP-wide Snowmass 2021 community planning process [4], have emphasized the need for greater flexibility in the use of neutrino event generators and related software. A particularly problematic technical barrier is the current lack of a common data format for representing the output *events*: lists of simulated particles involved in a neutrino interaction together with information describing their properties and relationships. At present, each neutrino event generator group maintains a unique output format, substantially complicating the use of multiple generators in large-scale experimental simulation workflows [5,6]. The adoption of standard formats in the collider community has streamlined many components of the analysis pipeline. This has enabled straightforward analysis preservation in tools like Rivet [7–9], allowed theorists to reinterpret experimental limits on one exotic physics scenario in light of others [10,11], simplified comparisons between generators (see, *e.g.*, Ref. [12]), and supported interoperability between simulation tools [13–15].

The only major software product that currently provides an official interface to the event formats produced by all four of the most widely-used neutrino event generators (GENIE [16, 17], GiBUU [18], NEUT [19,20], and NuWro [21]) is NUISANCE [22], a framework for comparing simulation predictions to each other and to experimental data. Although this feature of NUISANCE has proven valuable for the field, its generic internal representation of the input events (the `FitEvent` C++ class) is too simplified for all applications, its event format conversion tools require linking to generator-specific shared libraries, and the need to support multiple evolving proprietary event formats represents a significant maintenance burden.

To facilitate further development of software products that interface with neutrino event generators, as well as to support the use of a wider variety of simulation-based neutrino interaction models in experimental analyses, we present a new event format, *NuHepMC*, as a universal standard to be adopted by consensus of the neutrino event generator community. The data structures, file formats, and basic concepts in NuHepMC are identical to the versatile and mature HepMC3 library [23] adopted by other subfields of HEP. Using HepMC3 as a foundation, we define in NuHepMC an extensible and extendable set of conventions for representing neutrino interaction physics in a tool-agnostic way. This approach provides a generic event format that can be used in all future simulation development for neutrino experiments. Because we avoid placing any limitations on the information that individual event generators may output, the NuHepMC standard enables lossless, bidirectional conversion between HepMC3 and the existing proprietary neutrino event formats.

In Sec. 2, we present the specification of the NuHepMC format.[1] The flux-averaged total cross section, a particularly important quantity for analyzing neutrino scattering simulations, is discussed in Sec. 3. Finally, Sec. 4 uses NuHepMC as an output format for GENIE, NuWro, NEUT, ACHILLES [24], and MARLEY [25] as well as an input format in NUISANCE to demonstrate a first NuHepMC-based analysis of neutrino event generator predictions.

2 Specification

The details of the NuHepMC specification are broken down into three categories that describe four components from HepMC3. Each element of the specification is labeled as a *Requirement*, a *Convention*, or a *Suggestion*. The requirements dictate a minimum level of information to be included when writing out events. The conventions are optional details that an event generator group can choose to include or omit while still conforming to the NuHepMC standard.[2] Finally, the suggestions are optional recommendations that are less strongly encouraged than the conventions.

The four HepMC3 components considered in this standard are the generator run metadata, event metadata, vertex information, and particle information. Specifications for each of these components can be found in Secs. 2.3, 2.4, 2.5, and 2.6 respectively.

2.1 Labeling scheme

The elements of the specification are enumerated below in the form <Component>.<Category>.<Index>, where the component of interest is given as G for generator run metadata, E for event metadata, V for vertex information, and P for particle information. The category is denoted by R for a requirement, C for a convention, and S for a suggestion. For example, the second convention for event metadata would be labeled as E.C.2.

If conventions or suggestions prove useful, become widely adopted, and are considered stable, they may become requirements in future versions of this specification. G.R.4 defines a requirement for event generator authors to signal whether or not specific optional elements of this specification have been followed.

[1] For the most up-to-date specification see https://github.com/NuHepMC/Spec.

[2] Helper functions and utilities for writing, reading, and interpreting NuHepMC events in C++ or Python can be found at https://github.com/NuHepMC/cpputils.

2.2 HepMC3 C++ classes

Throughout this standard, references are made to various HepMC3 C++ classes, *e.g.*, `HepMC3::GenRunInfo`. However, these are used as a convenient handle for data objects. This specification should not be considered specific to the HepMC3 C++ reference implementation.

In principle, the HepMC3 ASCII format can be written out to file without any required dependencies. However, it is important to note that the HepMC3 library does provide bindings for both C++ and Python. The implementation within each tool using the NuHepMC format is left to the developers. At the time of writing there are official or unofficial conversion tools available for all of the aforementioned neutrino event generators, see Section 4.

2.3 Generator run metadata

The generator run metadata describes the overall setup of the event generator, *i.e.*, information that is not unique to a specific event. The NuHepMC specifications for this metadata are as follows:

2.3.1 Requirements

G.R.1 VALID GENRUNINFO:
All NuHepMC vectors[3] must contain a `HepMC3::GenRunInfo` instance.

G.R.2 NUHEPMC VERSION:
A NuHepMC `HepMC3::GenRunInfo` instance must contain the following three `HepMC3::IntAttributes` that specify the version of NuHepMC that is implemented:

- “`NuHepMC.Version.Major`”
- “`NuHepMC.Version.Minor`”
- “`NuHepMC.Version.Patch`”

This document describes **version 1.0.0** of NuHepMC.[4]

G.R.3 GENERATOR IDENTIFICATION:
A NuHepMC `HepMC3::GenRunInfo` instance must contain a `HepMC3::GenRunInfo::ToolInfo` for each ‘tool’ involved in the production of the vector thus far. The `ToolInfo` instance must contain non-empty name and version fields.

G.R.4 SIGNALING FOLLOWED CONVENTIONS:
To signal to a user that an implementation follows a named convention from this specification, a `HepMC3::VectorStringAttribute` attribute must be added to the `HepMC3::GenRunInfo` instance named “`NuHepMC.Conventions`” containing the labels of the conventions adhered to, *e.g.* “`G.C.2`”. *n.b.* that G.R.5 requires the implementation of at least one convention and so this attribute must exist.

G.R.5 FLUX-AVERAGED TOTAL CROSS SECTION:
The flux-averaged total cross section, $\langle\sigma\rangle$, is a scaling factor that is needed to convert a distribution of simulated events into a cross-section prediction. Details on the

[3]The term *vector* is used herein, rather than *file*, as HepMC3 events are frequently piped or streamed between MC tools without ever being wholly persisted to memory or disk.

[4]A script is provided for converting existing NuHepMC files from an older version of the standard to the latest version of the standard at https://github.com/NuHepMC/Spec/blob/master/scripts/NuHepMC_update_standard.py.

mathematical definition of this quantity are given in Sec. 3.1. If the value of $\langle\sigma\rangle$ is known at the start of a generator run, convention G.C.2 should be used to store it in the `HepMC3::GenRunInfo`.

In general, the value of $\langle\sigma\rangle$ is not known at the start of a generator run, but it can be calculated as events are produced (see Appendix A). In this case, convention E.C.4 should be used to store the running estimate and associated statistical uncertainty in each `HepMC3::GenEvent`.

One of these two methods for communicating $\langle\sigma\rangle$ to users must be implemented and signaled via G.R.4.

G.R.6 Cross Section Units and Target Scaling:
There are a variety of units typically used to report both measured and predicted cross sections in HEP. For neutrino cross sections specifically, 10^{-38} cm^2 per nucleon is common, but not ubiquitous. Both of the following `HepMC3::StringAttributes` must be included on the `HepMC3::GenRunInfo` to indicate the cross section units used within a vector. Possible values of the attributes are not restricted by this specification, but the meanings of the following values are reserved to standardize existing conventions. It is strongly recommended that new implementations use these wherever possible.

- ‘‘`NuHepMC.Units.CrossSection.Unit`’’:
  - ‘‘`pb`’’: Picobarns or 10^{-36} cm^2. Our recommendation.
  - ‘‘`1e-38 cm2`’’: The choice of 10^{-38} cm^2 is the most frequent in the neutrino literature.
  - ‘‘`cm2`’’: Using bare cm^2 in this option, without any power-of-ten scaling, is not recommended due to numerical precision concerns. The natural scale of neutrino–nucleon cross sections is approximately 10^{-38}, which is very close to the minimum representable IEEE 754 single-precision floating point number [26].
- ‘‘`NuHepMC.Units.CrossSection.TargetScale`’’:
  - ‘‘`PerAtom`’’: Our recommendation. For this choice, the target number density ρ mentioned in Sec. 3 is expressed in atoms per unit volume.
  - ‘‘`PerNucleon`’’: A common alternative in the literature.

The chosen units should be assumed to apply to the reported value of $\langle\sigma\rangle$, which is a property of the generator run and may be stored according to either G.C.2 or E.C.4. Consistent units should also be used for the total (E.C.2) and process-specific (E.C.3) cross sections describing the primary interaction in each individual event.

It is ultimately up to the user to parse these attributes and decide whether any additional scaling is needed for their purposes, but the use of the above reserved values will facilitate automated processing. The recommended value of picobarns per atom is chosen to remain consistent with the assumptions of other tools that read and write HepMC3 files beyond the neutrino community, such as Rivet [7–9]. Consistency facilitates interoperability and reduces maintenance burdens.

G.R.7 Event Weights:
For weights that will be calculated for every event, HepMC3 provides an interface for storing the weight names only once in the `HepMC3::GenRunInfo` instance. At least one event weight named ‘‘`CV`’’ must be declared in the `HepMC3::GenRunInfo` instance and filled for every event.

This weight may be 1 or constant for every event in a generator run (in the case of an *unweighted* event vector). This weight must always be included by a user when producing cross section predictions from a NuHepMC vector and should never be assumed to be 1 for every event.

G.R.8 PROCESS METADATA:
A NuHepMC `HepMC3::GenRunInfo` instance must contain a `HepMC3::VectorIntAttribute` named "`NuHepMC.ProcessIDs`" listing all physics process IDs as integers. For each valid process ID, the `HepMC3::GenRunInfo` instance must also contain two other attributes giving a name and description of each:

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

where <ID> enumerates all process IDs present in "`NuHepMC.ProcessIDs`". (See also E.R.3).

G.R.9 VERTEX STATUS METADATA:
The NuHepMC `HepMC3::GenRunInfo` instance must contain a `HepMC3::VectorIntAttribute` named "`NuHepMC.VertexStatusIDs`" 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 declared vertex status, the `HepMC3::GenRunInfo` instance must also contain two other attributes giving a name and description of each:

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

where <ID> enumerates all status codes present in "`NuHepMC.VertexStatusIDs`" (See also V.R.1).

G.R.10 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).

G.R.11 Non-standard Particle Numbers (PDG Codes):

Essentially all event generators in HEP use a standard set of integer codes to identify particle species. This numbering scheme is maintained by the Particle Data Group (PDG) and is regularly updated in their Review of Particle Physics [27, Sec. 45, p. 733].

It is expected that neutrino event generators may need to use codes for non-standard particle species (*i.e.*, those without an existing PDG code) for a variety of applications. This could include simulating exotic physics processes involving new particles as well as implementing bookkeeping methods involving generator-specific pseudoparticles.

The NuHepMC `HepMC3::GenRunInfo` instance must contain a `HepMC3::VectorIntAttribute` named "`NuHepMC.AdditionalParticleNumbers`" declaring any particle codes used that are not defined in the current PDG numbering scheme. Including any of the standard codes in this list is permitted but not required. The standard particle codes must not be reused to mean something different than in the PDG specification.

For each additional particle code, the `HepMC3::GenRunInfo` instance must also contain an attribute giving a unique name to the represented particle species:

- type: `HepMC3::StringAttribute`,
  name: "`NuHepMC.AdditionalParticleInfo[<PDG>].Name`"

where `<PDG>` enumerates all particle numbers present in "`NuHepMC.AdditionalParticleNumbers`".

See also G.C.5 for a suggested way of storing descriptions of these special particle species.

2.3.2 Conventions

G.C.1 Vector Exposure:

Each vector should contain a description of the exposure of the generator run. When simulating with some experimental exposure, often represented for accelerator neutrino experiments in units of "protons on target" (POT), the exposure should be described. Two attributes are reserved for signaling the exposure to users. One or both can be provided.

- type: `HepMC3::DoubleAttribute`, name: "`NuHepMC.Exposure.POT`"
- type: `HepMC3::DoubleAttribute`,
  name: "`NuHepMC.Exposure.Livetime`"

G.C.2 Flux-averaged Total Cross Section:

If known at the start of a run, the value of $\langle\sigma\rangle$ should be stored as a `HepMC3::DoubleAttribute` in the `HepMC3::GenRunInfo` named "`NuHepMC.FluxAveragedTotalCrossSection`". Optionally, the uncertainty in the flux-averaged total cross section may be stored as a `HepMC3::DoubleAttribute` in the `HepMC3::GenRunInfo` named "`NuHepMC.FluxAveragedTotalCrossSectionUncertainty`".

See E.C.4 if the cross section is not known at the start. Also, recall that from G.R.5, either this convention or E.C.4 must be used.

G.C.3 Citation Metadata:

Modeling components implemented based on published work should always be fully cited. The HepMC3::GenRunInfo should contain at least one HepMC3::VectorStringAttribute for each relevant modeling component, named according to the pattern ‘‘NuHepMC.Citations.<Comp>.<Type>’’. Valid substitutions for the <Comp> and <Type> fields are not restricted by this standard beyond the requirement that they are pure mixed-case alpha-numeric. The keys <Comp>=Generator and <Comp>=Process[<ID>] are reserved for use in specifying the main citation(s) for the interaction generator and for citing theoretical motivations behind individual processes, respectively. For common reference formats in the HEP field, some reserved values for the <Type> field are:

- ‘‘InspireHep’’ might contain one or more unique InspireHep identifiers (texkeys).
- ‘‘arXiv’’ might contain one or more unique arXiv identifiers (eprint numbers).
- ‘‘DOI’’ might contain one or more unique Digital Object Identifiers.

A tool that can read this metadata and automatically produce a BibTeX file containing entries for all cited publications is briefly introduced in Appendix C.

G.C.4 Beam Energy Distribution Description:
Each vector should contain a description of the incident flux of beam particles used to simulate the events. For many MC studies and experimental simulations in which the detector is not physically close to the source, a simple univariate energy distribution is enough to describe the particle beam. The two types of energy distribution covered by this convention are mono-energetic beams and those with energy spectra that can be described by one-dimensional histograms. The type should be signaled via a HepMC3::StringAttribute named ‘‘NuHepMC.Beam.Type’’ with value ‘‘MonoEnergetic’’ or ‘‘Histogram’’ stored on the HepMC3::GenRunInfo. For both types, relevant units can be signaled via two attributes:

- ‘‘NuHepMC.Beam.EnergyUnit’’. Possible values of the attribute are not restricted, but the meanings of ‘‘MEV’’ and ‘‘GEV’’ are reserved. This attribute should always exist and be not empty.
- ‘‘NuHepMC.Beam.FluxUnit’’. Possible values of the attribute are not restricted, but the meaning of ‘‘Arbitrary’’ is reserved to signal that the normalization of the distribution was not known or used by the simulation. If this attribute is not used then the normalization will be assumed to be arbitrary.

For the case of a ‘‘MonoEnergetic’’-type distribution, all beam particles in the vector must have identical energy. The attribute ‘‘NuHepMC.Beam[<PDG>].MonoEnergetic.Energy’’ can be used to signal the beam energy in the lab frame, but the usage of this attribute is optional as the energy can be determined from the first (or any) event in the vector.

For the case of a ‘‘Histogram’’-type distribution, the histogram should be encoded into two HepMC3::VectorDoubleAttribute per beam species on the HepMC3::GenRunInfo:

- ‘‘NuHepMC.Beam[<PDG>].Histogram.BinEdges’’
- ‘‘NuHepMC.Beam[<PDG>].Histogram.BinContent’’

where <PDG> enumerates the PDG particle numbers of all beam particles present in the event vector. *n.b.* the number of entries in the ‘‘BinEdges’’ vector should always be one more than the number of entries in the ‘‘BinContent’’ vector.

The HepMC3::BoolAttribute,

- ''NuHepMC.Beam[<PDG>].Histogram.ContentIsPerWidth'',

should be set to true to indicate that the total neutrino flux in a bin of the histogram should be computed by multiplying the bin content by the bin width. A value of false, assumed by default when this attribute is not present, signals that the bin content should be used alone. While this distinction might be determined by carefully parsing the FluxUnit attribute, existing neutrino generators make different assumptions when sampling from input neutrino beam energy distributions. An explicit attribute is therefore defined here to reduce ambiguity.

For a suggestion on how to encode additional information about more realistic neutrino beam descriptions, see E.S.1.

General flux handling is a complex problem that is not currently standardized. This convention is suitable for simplified studies by theorists and phenomenologists interested in accelerator-based neutrino experiments. A standardized flux format usable for many other applications, *e.g.*, atmospheric neutrino measurements, as well as its corresponding description within the event record, is outside the scope of this specification. Development of such a standard is left to future community efforts.

G.C.5 NON-STANDARD PARTICLE NUMBER DESCRIPTIONS:
For each additional particle number <PDG> declared in the ''NuHepMC.AdditionalParticleNumbers'' attribute, according to G.R.11, the HepMC3::GenRunInfo instance may contain an attribute giving a description of the particle:

- type: HepMC3::StringAttribute,
  name: ''NuHepMC.AdditionalParticleInfo[<PDG>].Description''

For non-standard particles that should be further simulated by particle propagation simulations, such as GEANT4 [28], additional information encoded here may be enough to enable automatic propagation. In this version of NuHepMC, no attempt is made to standardize a format for such information, but it is suggested that HepMC3::GenRunInfo attributes of the form, ''NuHepMC.AdditionalParticleNumber[<PDG>].<SimName>.<AttrName>'', may be useful and could be standardized in future versions of this specification. These additional attributes could be expected to include, at minimum, the particle's mass, width, spin, and electric charge.

2.3.3 Suggestions

G.S.1 RUN CONFIGURATION:
It is suggested that a NuHepMC HepMC3::GenRunInfo instance should contain all information required to reproduce the events in the vector. This may be stored in attributes with names beginning with ''NuHepMC.Provenance''. The information required will necessarily be generator-specific, but it should be noted that storing the full initial state of any random number generators used during the generator run is crucial for exact reproduction.

G.S.2 COMPLETE STATUS METADATA:
While G.R.9 and G.R.10 explicitly do not require implementations to emit metadata for standard status codes defined in the HepMC3 standard, it is suggested

that the complete list of status codes used by an implementation are included in the ‘‘NuHepMC.VertexStatusInfo’’ and ‘‘NuHepMC.ParticleStatusInfo’’ attributes.

G.S.3 TARGET ATOMIC ABUNDANCES:
Using the flux-averaged total cross section $\langle\sigma\rangle$, it is straightforward to compute partial cross sections for specific final states, interaction modes, and components of the overall target material used in a run. An example of the last of these is a flux-averaged cross section for interactions with C when a composite CH target was simulated.

In some cases, it can be desirable to use an existing vector to create cross-section predictions for a target material with a different composition than the one initially used, *e.g.*, distributions for CH_2 from a vector generated for a CH target. Making such predictions requires a knowledge of the flux-averaged relative abundances of the nuclides present in the original target material. For situations in which these are known at the start of a run, a `HepMC3::VectorStringAttribute` named ‘‘NuHepMC.TargetMaterialRelativeAtomicAbundance’’ may be stored in the `HepMC3::GenRunInfo`. Each element of the vector should be a string of the form ‘‘100ZZZAAA0[x]’’ containing a nuclear PDG code and the corresponding relative abundance enclosed in square brackets []. For nuclides with a relative abundance of 1, the nuclear PDG code may optionally be used alone. For example, a pure CH_2 target might be represented in this format as [‘‘1000060120[0.5]’’, ‘‘1000010010’’].

For realistic experimental simulations that involve a detector and a non-uniform neutrino beam, the flux-averaged relative abundances are typically difficult to calculate analytically. While running estimates of the abundances can in principle be obtained as the generator runs (using techniques akin to those in E.C.4), alternative approaches will generally be simpler to implement. Cases in which a particular detector material is of interest can be addressed by selecting events based on their position within the lab frame (*c.f.* E.R.5). Further exploration of this topic is left to future standardization efforts focused on neutrino fluxes and detector geometries.

2.4 Event data

The event is used to store information about one primary interaction process and any relevant secondary processes. An event is described by arbitrary metadata and a graph of particles (edges) and vertices (nodes), each with their own arbitrary metadata. The NuHepMC specifications for events are as follows:

2.4.1 Requirements

E.R.1 HEPMC3 COMPATIBILITY:
The HepMC3 standard places some constraints on valid event graphs. These constraints must be respected by valid NuHepMC events as full compatibility with HepMC3 is required. More details of these constraints can be found in Ref. [23].

Existing neutrino event generators often rely on effective descriptions of the nuclear environment in a neutrino–nucleus hard scattering process. This means that four-momentum may not be explicitly conserved for the neutrino–nucleus system. Energy and momentum can be exchanged with a *nuclear remnant*, which is often not treated carefully in the simulation of a primary neutrino–nucleon collision, via initial- and final-state interactions. Implementations are of course free to thoroughly model all involved particles, conserving four-momentum over the whole system, including a fully-simulated nuclear remnant. For those implementations where such a requirement is not feasible or would delay the adoption of this standard, P.C.2 reserves a non-standard particle number that can be used to represent a nuclear remnant that is not precisely simulated.

E.R.2 EVENT NUMBER:
Each `HepMC3::GenEvent` must have a non-negative event number that is unique within a given vector.

E.R.3 PROCESS ID:
The process ID for the primary physics process that is represented in the `HepMC3::GenEvent` must be recorded in a `HepMC3::IntAttribute` named ‘‘`signal_process_id`’’. The metadata describing this process ID must be stored according to G.R.8.

E.R.4 UNITS:
Energy and position units must be explicitly set in the `HepMC3::GenEvent`.

E.R.5 LAB POSITION:
The position of the event in the lab frame must be added as a `HepMC3::VectorDoubleAttribute`, named ‘‘`lab_pos`’’, with the same units as used when implementing E.R.4. See E.C.5 for how to optionally store time in this attribute. If the simulation did not involve a macroscopic geometry, then this variable may be set to `[0, 0, 0]`.

E.R.6 VERTICES:
An event must contain at least one `HepMC3::GenVertex`, and must have one and only one with a *primary interaction vertex* status code. No `HepMC3::GenVertex` may have a *not defined* status code. (See V.R.1 for additional details).

E.R.7 BEAM AND TARGET PARTICLES:
An event must contain exactly one particle with the *incoming beam particle* status code and one particle with the *target particle* status code (see P.R.1). It is recommended that, in cases where the colliding initial-state particles are distinct, the more massive of the two should be considered the target. For neutrino scattering, the target will thus often be a complex nucleus or a free nucleon. In the case of equally massive particles, the choice to label one of them as the target is arbitrary.

n.b. A nucleon (or quark) bound within a nucleus should never be marked as the target particle; the nucleus itself should be considered the target. P.C.1 provides a convention for marking a constituent bound nucleon struck by the incoming beam particle in the event graph.

E.R.8 EVENT COMPLETENESS:
All simulated incoming and outgoing physical particles must be written to the event. The storage of intermediate particles is considered an implementation detail.

E.R.9 EVENT FRAME OF REFERENCE:
As the primary use of HepMC3 is for experiment simulation, all particle momenta specified via standard HepMC3 mechanisms must be defined in the lab frame. From the information required by a standard HepMC3 event, arbitrary frame transformations should be possible and can be performed by the `HepMC3::GenEvent::boost` method.

Table 1: Optional set of identifier ranges for various high-level process categories.

Identifier	Process
100-199	Low-energy nuclear scattering
200-299	Quasi-elastic
300-399	Multi-body scattering
400-499	Baryon resonance production
500-599	Shallow inelastic scattering
600-699	Deep inelastic scattering
700-	Other process types

If storing kinematic information in other frames of reference facilitates later calculations, this should be done via simulation-specific `HepMC3::Attributes` on the relevant `HepMC3::GenParticles`.

2.4.2 Conventions

E.C.1 PROCESS IDS:

It is not appropriate to mandate a specific set of interaction processes and assign them IDs in this standard. Different models make different choices, and it is impossible to foresee modeling developments that would require new process IDs to be defined in the future. Instead, the ranges of IDs given below are provided as a categorization of processes to facilitate high-level analysis. If a given process naturally fits in one of these categories it can be helpful to choose IDs that follow this convention, but an explicit aim of this standard is to avoid constraining or defining what processes can be stored and described. *n.b.* Even if an implementation uses the convention in Table 1, it must still adhere to G.R.8.

If following this convention, charged-current (CC) processes should have identifiers in the X00-X49 block and neutral-current (NC) processes should have them in the X50-X99 block. Negative process IDs should be used for electromagnetic interactions in event generators that include them.

In Table 1, distinct ranges for shallow inelastic scattering and deep inelastic scattering are provided because existing neutrino generators often make such a distinction in both the modeling details and simulated event metadata. It should be noted that the boundary between these two is not *a priori* well defined and choices vary significantly in current neutrino scattering simulations, usually defined with reference to some boundary value in the invariant mass of the hadronic final state. The lack of consensus and common definitions hinders progress in modeling neutrino-induced low energy hadronization. New efforts in this area are sorely needed and would facilitate collaboration with the collider community, as discussed in Section 2.4 of Ref. [4].

E.C.2 TOTAL CROSS SECTION:

The total cross section for the incoming beam particle, with its specific energy, to interact with the target particle should be stored in a `HepMC3::DoubleAttribute` on the `HepMC3::GenEvent`, named “`tot_xs`”. See G.R.6 for how to communicate cross section units.

E.C.3 Process Cross Section:
The total cross-section for the selected process ID for the incoming beam particle, with its specific energy, to interact with the target particle should be stored in a `HepMC3::DoubleAttribute` on the `HepMC3::GenEvent`, named ‘‘`proc_xs`’’. See G.R.6 for how to communicate cross section units.

E.C.4 Estimated Flux-Averaged Total Cross Section:
Some simulations build up an estimate of the flux-averaged total cross section $\langle\sigma\rangle$ as they run, which makes implementing G.C.2 impractical in many cases. As an alternative, the built-in attribute `HepMC3::GenCrossSection`, accessed via `GenEvent::cross_section`, should be used to store the current estimate of $\langle\sigma\rangle$. A user should only ever need to use the estimate provided with the last event that they read to correctly scale an event rate to a cross-section prediction. This means that statistically correct predictions can be made without reading an event vector to the end. The `HepMC3::GenCrossSection::cross_section_errors` data member can be used to decide when enough events have been read to reach some desired statistical precision on the total cross section. The best estimate from the generator run will always be provided on the final event in a vector.

For event generators that do not currently provide the value of $\langle\sigma\rangle$ in the output, Appendix A provides an algorithm for computing a running estimate, with associated Monte Carlo statistical uncertainty, as events are generated.

When implementing this convention, ensure that the `cross_sections` and `cross_section_errors` data members are the same length as the number of weights defined in the header. These should be filled with the current estimate of the total cross section for each variation based on all events generated so far, including the current event. Additionally, the `HepMC3::GenCrossSection` data members `accepted_events` and `attempted_events` should be filled with appropriate values.

E.C.5 Lab Time:
If the ‘‘`lab_pos`’’ attribute vector contains three entries then it is assumed to only contain the spatial position of the event. If it contains four entries, then the fourth entry is interpreted as the time of the event in seconds.

2.4.3 Suggestions

E.S.1 Beam Description (Beam Simulation)
For more complex beam simulations that can not adequately be described by a single energy or energy histogram (see G.C.4), it is suggested that the full parent decay history is included in the `HepMC3::GenEvent`. A full set of conventions for the description of beam particle production and parent particle decay chains (for the case of neutrino beams) is currently outside the scope of this specification, but generator implementations can signal that they adhere to this suggestion to notify users that some or all of the beam particle production information is included in the event.

2.5 Vertex information

The vertices in a HepMC3 event are used to connect groups of incoming and outgoing particles. For the vertex information, there is only one requirement and one convention in the present version of the NuHepMC standard.

Table 2: Set of vertex status codes.

Status Code	Meaning	Usage
0	Not defined	Do not use
1	Primary interaction vertex	Recommended for all cases
2	FSI Summary vertex	Recommended for all cases
3-20	Reserved for future NuHepMC standards	Do not use
21-999	Generator-dependent	For generator usage

2.5.1 Requirements

V.R.1 Vertex Status Codes:
The standard HepMC3 range of reserved values for `HepMC3::GenVertex::status` is extended to include the concept of a primary vertex, corresponding to the *primary* hard-scattering process (*i.e.*, the one labeled according to E.R.3), and a final state interaction (FSI) summary vertex. The full set of defined status codes can be found in Table 2. Implementations are free to define specific vertex status codes to refer to individual FSI (or ISI) processes and output as much information as they require. However, a single summary vertex may be useful for some purposes if the full FSI history is very detailed or not often needed by users. See Figure 4 for an example of an event with such a summary vertex.

Any secondary vertex included within a NuHepMC event may have a status between 21 and 999. G.R.9 requires that all generator-specific status codes must be fully described by attributes stored in the `HepMC3::GenRunInfo`.

2.5.2 Conventions

V.C.1 Bound Nucleon Separation Vertex
When an interaction with a nucleon bound within a nucleus with definite kinematics is simulated, a `HepMC3::GenVertex` corresponding to the separation of the struck nucleon and the nuclear remnant may be included and assigned status code 21. If this convention is signaled via G.R.4, then status code 21 need not be included in the implementation of G.R.9.

2.6 Particle information

In the current version of the NuHepMC standard, there is only a single requirement and two conventions for the particle information.

2.6.1 Requirements

P.R.1 Particle Status Codes:
The standard HepMC3 range of reserved values for `HepMC3::GenParticle::status` is slightly extended to include the concept of a target particle. For neutrino scattering, this will usually be a target nucleus. The status codes are defined in Table 3.

Note especially that any incoming real particle must have a status code of 4 or 20, and any outgoing real particle must have a status code of 1. Special care must be taken when including the effects of initial-state and final-state interactions.

Table 3: 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

Any internal particle included within a NuHepMC event may have a status in the range than 21-200. G.R.10 requires that all generator-specific status codes must be fully described by attributes stored in the `HepMC3::GenRunInfo`.

2.6.2 Conventions

P.C.1 PARTICLE STATUS CODES:

When an interaction with a bound nucleon with definite kinematics is simulated, the internal `HepMC3::GenParticle` corresponding to the bound nucleon should have status code 21. If this convention is signaled via G.R.4, then status code 21 need not be included in the implementation of G.R.10.

P.C.2 NUCLEAR REMNANT PARTICLE CODE:

HepMC3 places restrictions on all external particles in the event graph to facilitate automatic checking of four momentum conservation at the event graph level. As a result, a new particle number 2009900000 is defined to identify a nuclear remnant that is not precisely handled by the interaction simulation. If a generator does correctly simulate any nuclear remnant, such that the total mass and energy are well defined, then this convention is unnecessary and the remnant can be written out as a particle with a standard PDG nuclear number and mass and momentum set as normal. A pseudoparticle with this number is explicitly an implementation detail that should be used to abide by HepMC3 constraints to not have vertices with no outgoing particles. Any pseudoparticle with this number should not be considered for physics analyses or onward simulation. The number is chosen, according to the PDG scheme [27, Sec. 45, p. 733], to be outside the range reserved for nuclear and quark-content particles. Its status as a non-standard particle code is signaled by the 6th and 7th least significant digits being set to 9.

If the the kinematics of the nuclear remnant are not known, but the proton and neutron numbers are well-defined, then the corresponding PDG nuclear number may be added as a `HepMC3::IntAttribute` on the `HepMC3::GenParticle` named “`remnant_particle_number`”.

If this convention is signaled via the mechanism described in G.R.4, then particle number 2009900000 need not be included in the implementation of G.R.11.

3 Flux-averaged total cross section

Comparisons of event generator predictions to external model calculations and experimental data typically involve the conversion of simulated event distributions to total or differential cross sections. This conversion is usually made using a scaling factor $\langle\sigma\rangle$ called the flux-averaged total cross section. This factor is simple to use but often difficult to calculate analytically. Given the importance of $\langle\sigma\rangle$ for analyses of simulated neutrino scattering events, we provide mathematical details about its definition and calculation.

Section 3.1 derives an expression for $\langle\sigma\rangle$ that is generally applicable to most simulations of interest for neutrino experiments, including those with a time-dependent neutrino source and a full treatment of the detector geometry. Example methods for obtaining Monte Carlo estimators of $\langle\sigma\rangle$ suitable for storage via E.C.4 are described in Appendix A. Section 3.2 describes some simple cases in which $\langle\sigma\rangle$ may be calculated directly, making them good candidates for situations in which event generators may implement G.C.2.

3.1 Derivation

Let $\phi(f, E_\nu, \vartheta_\nu, \varphi_\nu, \mathbf{x}, t)$ be the differential flux of (anti)neutrinos of species f with energy E_ν, momentum direction (expressed in terms of the Cartesian unit vectors)

$$\hat{\mathbf{p}}_\nu = \sin\vartheta_\nu \cos\varphi_\nu\,\hat{\mathbf{x}} + \sin\vartheta_\nu \sin\varphi_\nu\,\hat{\mathbf{y}} + \cos\vartheta_\nu\,\hat{\mathbf{z}}\,, \tag{1}$$

and instantaneous three-position $\mathbf{x}$ at time t. This quantity is defined so that

$$\Phi(\mathbf{x}) = \sum_f \int dE_\nu\, d\cos\vartheta_\nu\, d\varphi_\nu\, dt\, \phi\,, \tag{2}$$

has units of integrated flux (*e.g.*, cm^{-2}).

Assuming that the relevant neutrino interaction cross sections are sufficiently small that beam attenuation and multiple scattering effects can be neglected, then the total number of interactions N expected in a volume of interest,

$$V = \int d^3\mathbf{x}\,, \tag{3}$$

is given by

$$N = \sum_f \sum_j \int dE_\nu\, d\cos\vartheta_\nu\, d\varphi_\nu\, d^3\mathbf{x}\, dt\, \phi\, \rho\, \sigma\,, \tag{4}$$

where $\rho = \rho(j, \mathbf{x})$ is the number density of the j-th kind of target at position $\mathbf{x}$. The symbol $\sigma = \sigma(f, j, E_\nu)$ denotes the total cross section for (anti)neutrinos of species f and energy E_ν to interact with the j-th kind of target (typically a particular nuclide).

One may rewrite the expression for N in the simple form

$$N = \langle\Phi\rangle \cdot \langle T\rangle \cdot \langle\sigma\rangle\,, \tag{5}$$

where

$$\langle\Phi\rangle = \frac{1}{V} \sum_f \int dE_\nu\, d\cos\vartheta_\nu\, d\varphi_\nu\, d^3\mathbf{x}\, dt\, \phi\,, \tag{6}$$

is the average flux in the volume V,

$$\langle T\rangle = \frac{1}{\langle\Phi\rangle} \sum_f \sum_j \int dE_\nu\, d\cos\vartheta_\nu\, d\varphi_\nu\, d^3\mathbf{x}\, dt\, \phi\, \rho\,, \tag{7}$$

is the flux-averaged number of targets illuminated by the (anti)neutrinos, and

$$\langle\sigma\rangle = \frac{N}{\langle\Phi\rangle \cdot \langle T\rangle}, \tag{8}$$

is the flux-averaged total cross section. For a discussion on how these might be calculated using Monte-Carlo methods see Appendix A.

3.2 Analytic calculation under simplifying assumptions

In certain simple cases, the flux-averaged total cross section $\langle\sigma\rangle$ is known at the start of the run and can be computed analytically to implement G.C.2. A common example is the case of a point target of type b that is exposed to uni-directional beam of (anti)neutrinos of species a with a fixed emission time. In this case, we may write the differential flux and target density as

$$\phi = A(E_\nu)\,\delta(\cos\vartheta_\nu - \cos\theta_0)\,\delta(\varphi_\nu - \phi_0)\,\delta(t - t_0)\,\delta_{fa}, \tag{9}$$

$$\rho = B\,\delta^3(\mathbf{x} - \mathbf{x}_0)\,\delta_{jb}. \tag{10}$$

Here δ_{xy} is the Kronecker delta and the constants θ_0, ϕ_0, t_0, $\mathbf{x}_0$, a, and b are arbitrary values of the relevant variables. The function $A(E_\nu)$ gives the incident (anti)neutrino energy spectrum (with unimportant normalization) and has units of flux divided by energy (*e.g.*, GeV^{-1} cm^{-2}). The constant B is dimensionless. From Eqs. 6, 7, and 8, it follows that

$$\langle\Phi\rangle = \int dE_\nu A(E_\nu), \tag{11}$$

$$\langle T\rangle = B, \tag{12}$$

$$\langle\sigma\rangle = \frac{\int dE_\nu A(E_\nu)\,\sigma(a, b, E_\nu)}{\int dE_\nu A(E_\nu)}. \tag{13}$$

If the function $A(E_\nu)$ and the total cross section $\sigma(a, b, E_\nu)$ are accessible during generator initialization, then the integrals in Eq. 13 are easily computable by standard numerical methods.

In the even simpler case where the neutrino source is also monoenergetic, then $A(E_\nu)$ contains a Dirac delta function, and the flux-averaged total cross section is just the total cross section evaluated at the fixed energy of the beam.

4 NuHepMC generator predictions

Using preliminary tools for converting proprietary neutrino event formats to the NuHepMC standard defined above, we demonstrate the utility of the common format with some cross-section predictions. See Tab. 4 for the current status of implementations of NuHepMC in the most widely-used neutrino event generators.[5] It is important to note that once built-in support for NuHepMC is adopted within a generator, the public converter from the previous generator-specific formats to NuHepMC will be deprecated.

A comparison to a recent neutrino-argon cross-section measurement from the MicroBooNE collaboration [29] is shown on the left-hand side of Figure 1. This comparison was made with

[5] An up to date table of the current status of adoption within the generators can be found in the NuHepMC Spec repository: https://github.com/NuHepMC/Spec.

Table 4: Current state of output formats and public converters available in neutrino event generators. Public converters are from default formats to NuHepMC.

Generator	Output Formats	Public Converter
Achilles	NuHepMC	N/A
GENIE	GHep (ROOT `TTree`), NuHepMC in review	No
GiBUU	LHA XML, NuHepMC in 2025 release	N/A
MARLEY	NuHepMC (in upcoming v2 release)	N/A
NEUT	`neutvect` (ROOT `TTree`)	Yes
NuWro	`event1` (ROOT `TTree`)	Yes

the NUISANCE [22] framework,[6] which before this implementation of NuHepMC would have to have been built against GENIE, NEUT, and NuWro binaries of compatible versions to be able to generate the predictions shown in the figure. In the present workflow, results can be obtained using only generator-agnostic tools for interpreting events stored in the NuHepMC format. This reduction of technical requirements will dramatically lower the barrier to making high-quality prediction–data comparisons for the neutrino community.

It is particularly worth noting that the different generators included in this comparison provide critical cross-section scaling information in different ways. Thanks to the provisions in this standard, NUISANCE is able to use G.R.4 to determine which representation of $\langle\sigma\rangle$ is available and then automatically scale the selected event rate to a differential cross-section prediction without any user input or generator-specific code or knowledge. For reference, the NEUT and NuWro event distributions are scaled via G.C.2, while GiBUU, GENIE, and Achilles are scaled via E.C.4.

We want to emphasize that the comparison to data is solely a demonstration of the prediction pipeline and not meant as a physics statement on the best or most recent predictions from each of the generators for the given measurement.

The right-hand side of Figure 1 presents MARLEY predictions of flux-averaged differential cross sections for ν_e produced from μ^+ decay at rest scattering on ^{40}Ar. The yellow histogram shows the inclusive charged-current prediction, and the other histograms show contributions arising from several distinct final states. The cross-section predictions obtained using NuHepMC without any MARLEY-specific code are very similar to those shown in a previous MARLEY publication [31].

Comparisons similar to those above can be easily produced for an arbitrary interaction channel using information available in the NuHepMC event record. A Monte Carlo estimator for the flux-averaged differential cross section in a bin of an arbitrary kinematic variable $x \in [x_k, x_{k+1})$ is given by

$$\left\langle \frac{d\sigma}{dx} \right\rangle_k \approx \frac{\langle\sigma\rangle \sum_{e=1}^{n} \delta_e^k w_e}{\Delta x_k \sum_{e=1}^{n} w_e}, \tag{14}$$

where $\langle\sigma\rangle$ is the flux-averaged total cross section (see Sec. 3), w_e is the statistical weight of the e-th event, n is the number of generated events, and $\Delta x_k = x_{k+1} - x_k$ is the width of the k-th bin. The symbol δ_e^k evaluates to unity when the value of x from the e-th event falls within the k-th bin and zero otherwise. In the case of unit-weight events, the value of $w_e = 1$ for all events, and $\sum_{e=1}^{n} w_e = n$. One thus recovers the traditional definition of these estimators for unweighted events.

[6]The NUISANCE framework can be obtained at: https://github.com/NUISANCEMC/nuisance.

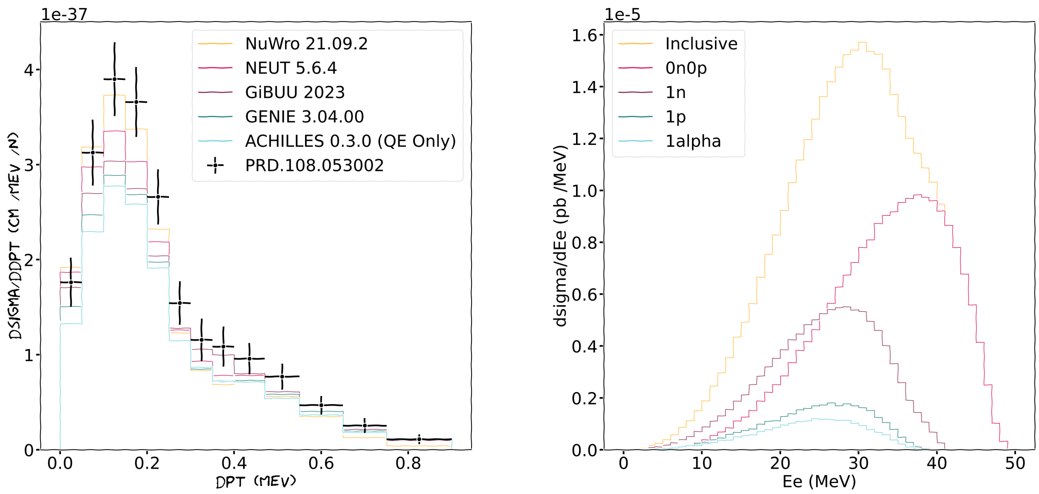

Figure 1: (*left*) Comparison of NuWro, NEUT, GiBUU, GENIE, and ACHILLES to the δp_T data from the MicroBooNE collaboration [29]. The comparison is a demonstration of the event generation to NUISANCE [30] pipeline and not a physics statement about the prediction quality of any generator. (*right*) Low-energy differential cross section predictions produced using MARLEY events in the NuHepMC format.[7]

The Monte Carlo statistical uncertainty (standard deviation) on the estimator of the differential cross section is approximately given by

$$\text{StdDev}\left(\left\langle \frac{d\sigma}{dx} \right\rangle_k\right) \approx \frac{\langle \sigma \rangle}{\Delta x_k \sum_{e=1}^{n} w_e} \sqrt{\sum_{e=1}^{n} \delta_e^k w_e^2}\,, \tag{15}$$

where we have assumed that $\langle \sigma \rangle$ is exactly known. For cases where it is estimated using Monte Carlo techniques rather than directly calculated, the statistical uncertainty discussed in Appendix A also applies.

Acknowledgments

The authors would like to thank A. Papadopoulou for sharing a NUISANCE implementation for the example comparison in Sec. 4; A. Verbytskyi for reading and commenting on this manuscript; and J. Tena Vidal for implementing an early version of this standard to test its practicality for a non-author.

Funding information This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. The work in this manuscript is supported by the Royal Society, grant number URF\R1\211661.

[7] The analysis for MicroBooNE is in the NUISANCE analysis framework. The analysis for MARLEY can be found at https://github.com/NuHepMC/cpputils/blob/main/examples/Marley.ipynb.

A Estimation via Monte Carlo sampling

In this section we provide two example methods of estimating the flux-averaged total cross-section when it is not known *a priori*. The first method is suitable for GENIE-like codes that calculate energy-dependent total cross sections prior to event generation itself. The second is suitable for ACHILLES-like codes that do not use pre-calculated total cross sections in this way.

A.1 GENIE-like approach

The (anti)neutrino interactions that occur in the volume V are drawn from the probability distribution

$$P(f, j, E_\nu, \vartheta_\nu, \varphi_\nu, \mathbf{x}, t) = \frac{1}{N}\,\phi\,\rho\,\sigma\,. \tag{A.1}$$

It follows from the definitions in Sec. 3 that

$$\langle\sigma\rangle = \left[\sum_f \sum_j \int P(f, j, E_\nu, \vartheta_\nu, \varphi_\nu, \mathbf{x}, t)\,\frac{1}{\sigma}\,dE_\nu\,d\cos\vartheta_\nu\,d\varphi_\nu\,d^3\mathbf{x}\,dt\right]^{-1}. \tag{A.2}$$

One may therefore obtain a Monte Carlo estimator $\hat{\sigma}$ for the flux-averaged total cross-section $\langle\sigma\rangle$ via the expression

$$\hat{\sigma} = \left[\frac{\sum_{e=1}^{n} \frac{w_e}{\sigma_e(f,j,E_\nu)}}{\sum_{e=1}^{n} w_e}\right]^{-1}, \tag{A.3}$$

where w_e is the statistical weight for the e-th event, n is the total number of generated events, and σ_e is the total cross section evaluated for the (anti)neutrino species (f), target (j), and incident energy (E_ν) sampled in the e-th event. An estimator for the Monte Carlo statistical standard deviation of $\hat{\sigma}$ is given by

$$\mathrm{StdDev}(\hat{\sigma}) = \sqrt{\mathrm{Var}(\hat{\sigma})} = \hat{\sigma}^2 \cdot \sqrt{\mathrm{Var}\left(\frac{1}{\hat{\sigma}}\right)} = \hat{\sigma}^2 \cdot \sqrt{\frac{n \cdot \sum_{e=1}^{n} w_e^2 \left(\frac{1}{\sigma_e} - \frac{1}{\hat{\sigma}}\right)^2}{\left(\sum_{e=1}^{n} w_e\right)^2 \cdot (n-1)}}\,. \tag{A.4}$$

Note that there is no universally accepted expression for the standard error on a weighted arithmetic mean such as the one computed in Eq. A.3 (before taking the reciprocal of the expression). The choice used in Eq. A.4 is based on the third expression for "$\mathrm{SEM_w}$" recommended in Ref. [32] based on bootstrapping studies (see also references therein). The quantity proposed as an estimator for the square of the standard error on a weighted mean

$$\bar{x}_w = \frac{\sum_{e=1}^{n} w_e\,x_e}{\sum_{e=1}^{n} w_e}\,, \tag{A.5}$$

of n values x_e is

$$\mathrm{Var}(\bar{x}_w) = \frac{n}{(n-1)(n\bar{w})^2}\left[\sum_{e=1}^{n} (w_e x_e - \bar{w}\bar{x}_w)^2 - 2\bar{x}_w(w_e - \bar{w})(w_e x_e - \bar{w}\bar{x}_w) + \bar{x}_w^2 (w_e - \bar{w})^2\right], \tag{A.6}$$

where

$$\bar{w} = \frac{1}{n}\sum_{e=1}^{n} w_e\,. \tag{A.7}$$

The expression in Eq. A.6 may be simplified to read

$$\mathrm{Var}(\bar{x}_w) = \frac{n}{(n-1)\left(\sum_{e=1}^{n} w_e\right)^2}\sum_{e=1}^{n} w_e^2 (x_e - \bar{x}_w)^2\,. \tag{A.8}$$

The result in Eq. A.4 is obtained immediately with the substitutions $x_e \to 1/\sigma_e$ and $\bar{x}_w \to 1/\hat{\sigma}$.

While the expressions in Eqs. A.3 and A.4 may be useful for estimating $\langle\sigma\rangle$ from a sample of generated events, they require access to the entire sample and are thus unsuitable for implementing the running estimate described in E.C.4. However, an adaptation of West's algorithm [33] provides a solution for this application. Let

$$n_0 = S_0 = \mu_0 = T_0 = 0\,. \tag{A.9}$$

Then, for the e-th event, let the values of these quantities be defined recursively via

$$n_e = n_{e-1} + 1 = e\,, \tag{A.10}$$
$$S_e = S_{e-1} + w_e\,, \tag{A.11}$$
$$\mu_e = \mu_{e-1} + \frac{w_e}{S_e}\left(\frac{1}{\sigma_e} - \mu_{e-1}\right), \tag{A.12}$$
$$T_e = T_{e-1} + w_e^2\left(\frac{1}{\sigma_e} - \mu_{e-1}\right)\left(\frac{1}{\sigma_e} - \mu_e\right), \tag{A.13}$$

where σ_e and w_e are assigned the same meanings as above.

The running Monte Carlo estimator $\hat{\sigma}_e$ of $\langle\sigma\rangle$ for the e-th event ($e > 0$) may then be written as

$$\hat{\sigma}_e = \frac{1}{\mu_e}\,. \tag{A.14}$$

Its estimated statistical uncertainty is given by

$$\text{StdDev}(\hat{\sigma}_e) = \frac{1}{\mu_e^2}\sqrt{\frac{n_e\,T_e}{(n_e - 1)\,S_e^2}}\,. \tag{A.15}$$

A.2 ACHILLES-like approach

When calculating Eq. (4), the equation can be expanded to include the integrals over the differential cross section giving

$$N = \sum_f \sum_j \int dE_\nu\, d\cos\vartheta_\nu\, d\varphi_\nu\, d^3\mathbf{x}\, dt\, d\Omega\, \phi_f\, \rho_j\, \frac{d\sigma_{fj}}{d\Omega}\,, \tag{A.16}$$

where Ω is the final state phase space and the cross section is broken up into individual processes as σ_{fj}. The equation can then be estimated using traditional Monte-Carlo methods giving

$$N \approx \frac{V}{n}\sum_f \sum_j \sum_i \phi_f(x_i)\,\rho_j(x_i)\,\frac{d\sigma_{fj}}{d\Omega}(x_i)\,, \tag{A.17}$$

where x_i are the selection of the variables of integration for the ith point and V is the volume of the space being integrated. The uncertainty on the integral estimate is thus given by the traditional calculation of the standard deviation. The variance of the integral estimate can be improved through the use of importance sampling, such as VEGAS [34, 35]. The integral over the neutrino fluxes and over the density of the nuclear targets can be obtained in a similar method, or by using other numerical integration techniques like quadrature. The flux-averaged cross section can thus be calculated with Monte-Carlo techniques using Eq. 8.

The results of the above Monte-Carlo calculation would produce a set of weighted events that can be used as is, or can be unweighted through the following procedure. First, the maximum values for each neutrino species and nucleus can be estimated using Monte-Carlo methods ($w_{fj}^{\max}$). Second, a neutrino species and nucleus is selected according to the probability

$$P_{fj} = \frac{w_{fj}^{\max}}{w^{\max}}, \tag{A.18}$$

where $w^{\max}$ is given by the sum of the maximum weight over all neutrino types and nuclei. Once a neutrino type and nucleus is selected, a set of initial and final state momenta is generated and the integrand for that particular neutrino type and nucleus is calculated. The event can then be unweighted by performing an accept-reject step using the ratio of weight of the event to the maximum weight calculated for the given neutrino type and nucleus. In other words, the weight of an event would be given by

$$\tilde{w}_i = w^{\max} \sum_{fj} \Theta \left(\frac{w_{fj}^{\max}}{w^{\max}} - \sum_{\substack{f'<f,\\ j'<j}} \frac{w_{f'j'}^{\max}}{w^{\max}} - R_1 \right) \Theta \left(\frac{w_i}{w_{fj}^{\max}} - R_2 \right), \tag{A.19}$$

with $R_1, R_2 \in [0,1]$ a uniformly distributed random number. This would result in a collection of events with either weights $w^{\max}$ or zero, and the average of these events would give an estimate of the total flux-averaged cross section, and only the events with non-zero weight are required to be written out as long as the number of attempted events (*i.e.* the total including the zero weight events) is also tracked. There are technical issues with directly using the true maximum sampled, since the maximum depends on the number of samples taken. There are many ways to mitigate this numerical issue, one approach is discussed in Section 4A of Ref. [36].

B Example event graphs

Figs. 2, 3, 4, 5 and 6 show some example event graphs for neutrino event generators that natively implement, or are convertible-to, the NuHepMC format. Although the details of each generator's implementation of this standard differ, the constraints that are imposed enable consistent usage for the most common tasks without any knowledge of each generator's implementation details.

C Extracting bibliography information

If the NuHepMC file signals that convention G.C.3 has been used, then it is possible to run the file through an external tool called HEPReference [37][8] to produce a BibTeX file and a short text blurb showing how to cite all the papers used to produce the events. An example use case can be found in the repository.

[8]The code can be found at: https://github.com/NuDevTools/HEPReference.

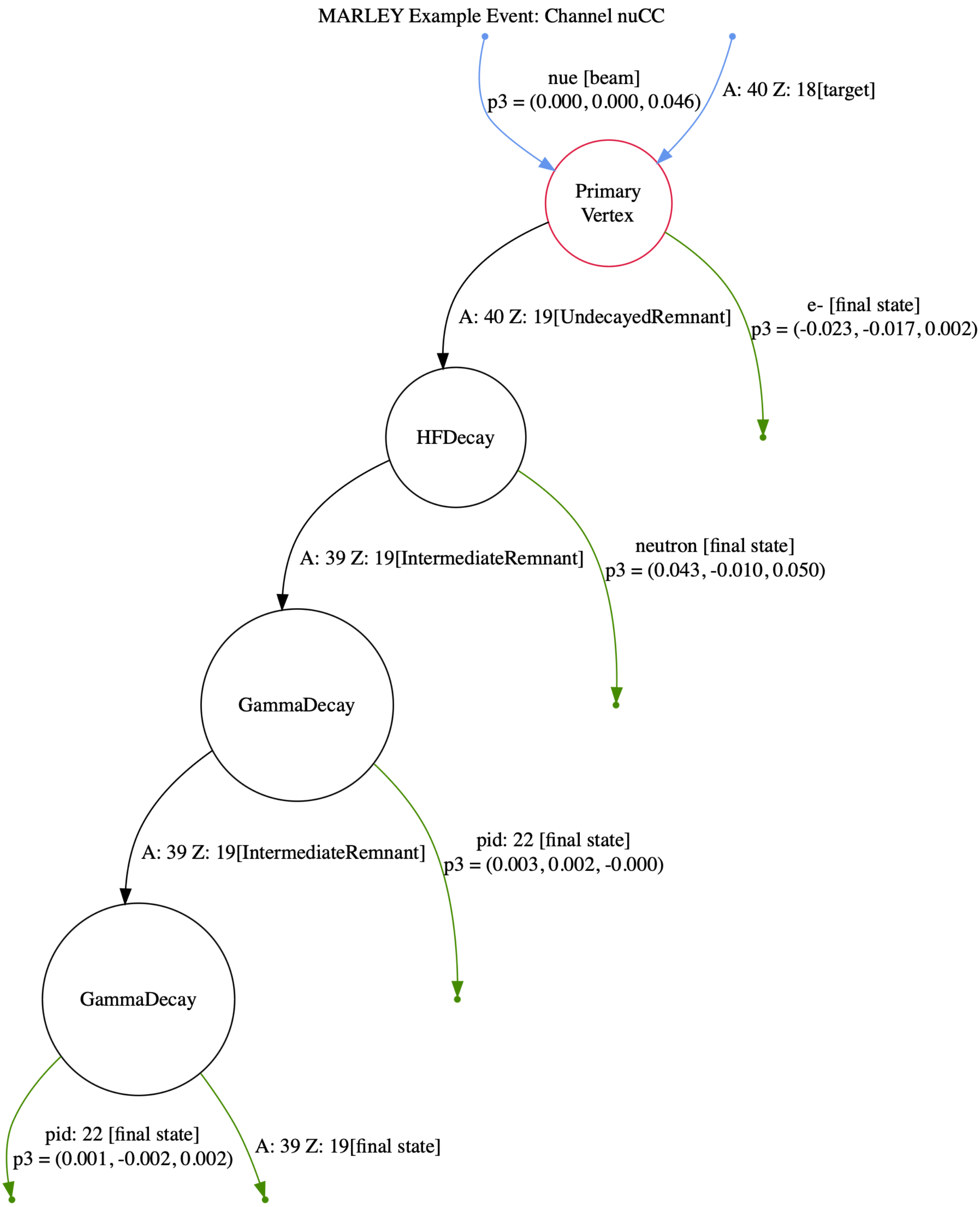

Figure 2: A MARLEY event graph in the NuHepMC format.

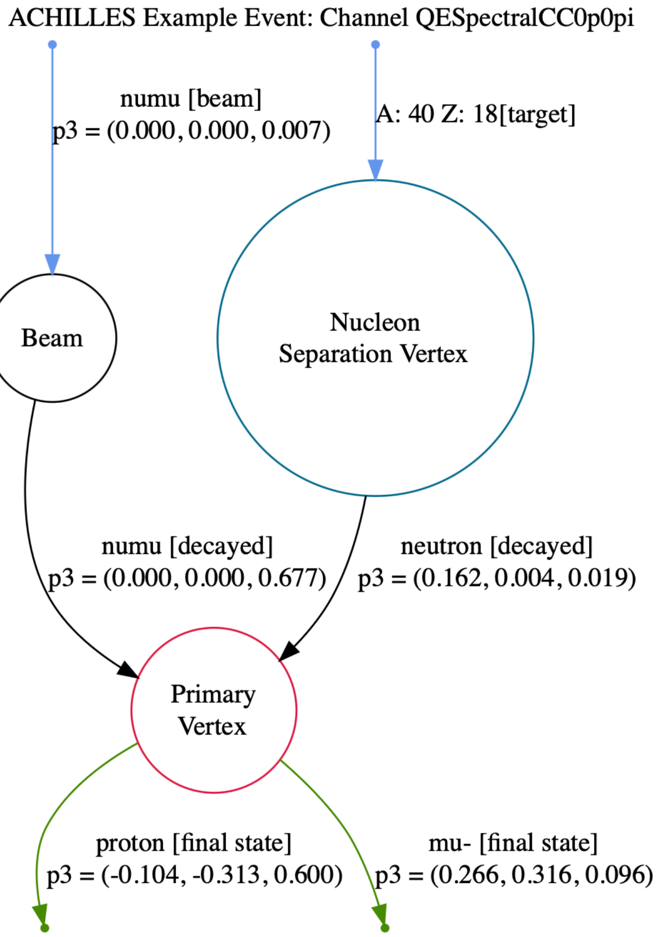

Figure 3: An ACHILLES event graph in the NuHepMC format.

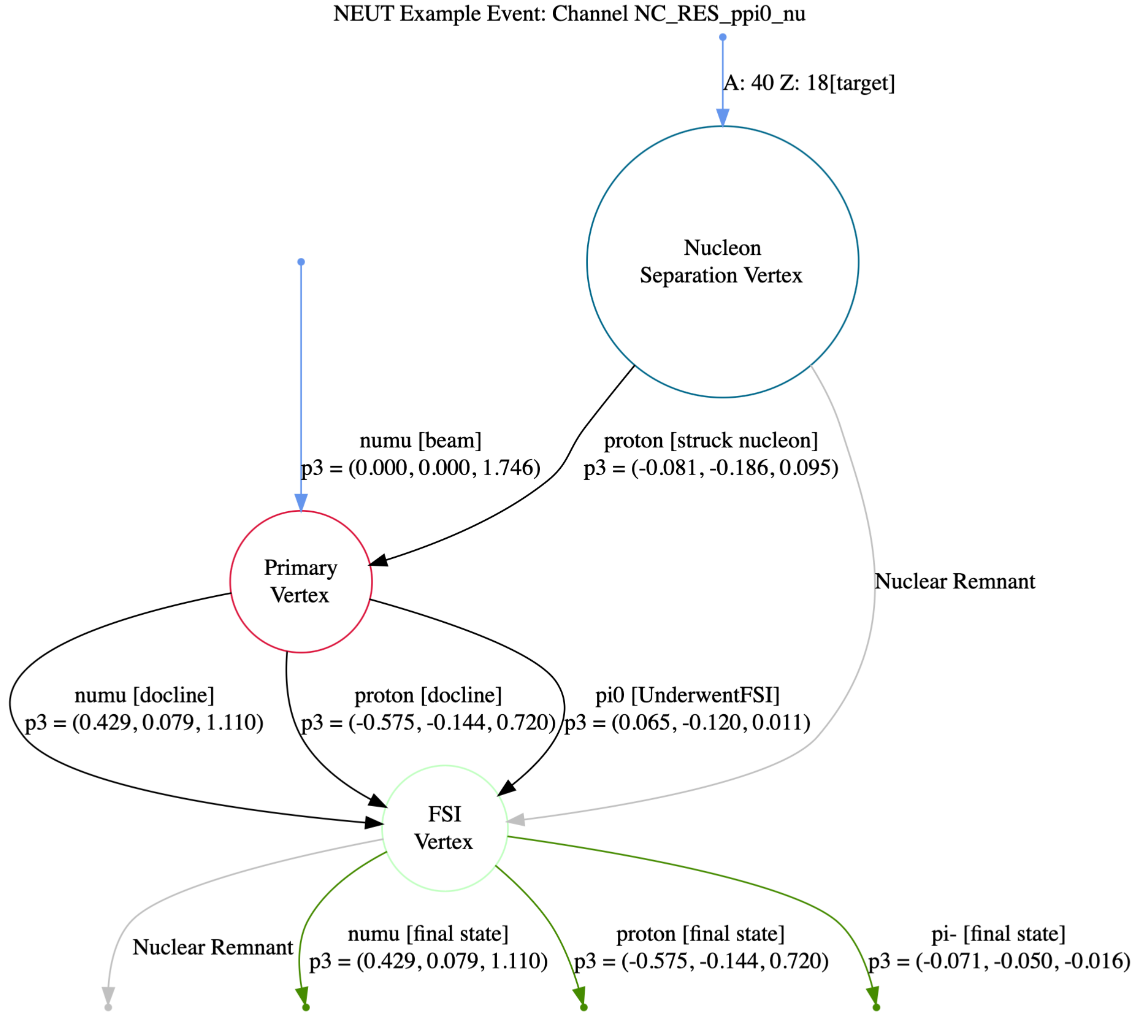

Figure 4: A NEUT event graph in the NuHepMC format.

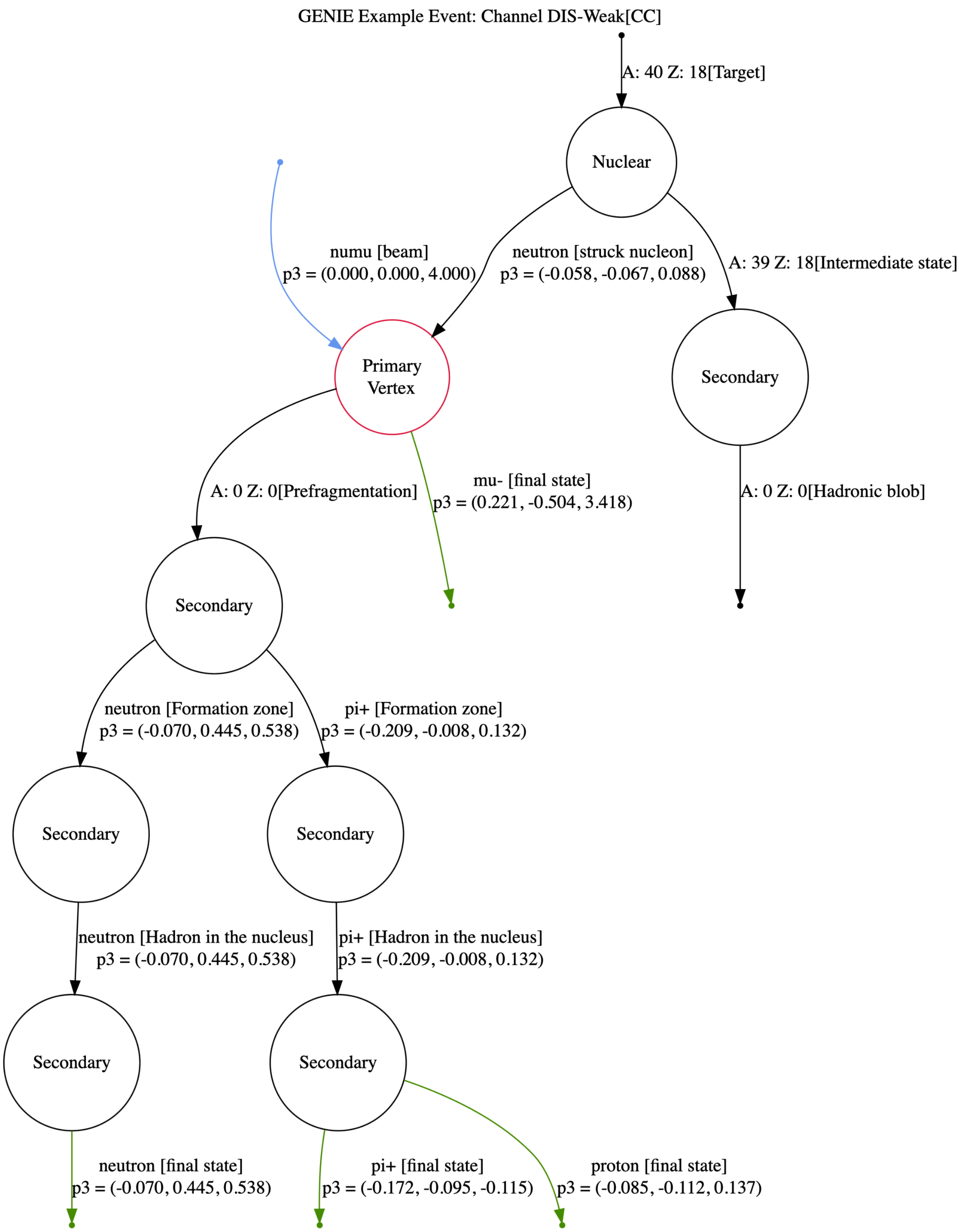

Figure 5: A GENIE event graph in the NuHepMC format.

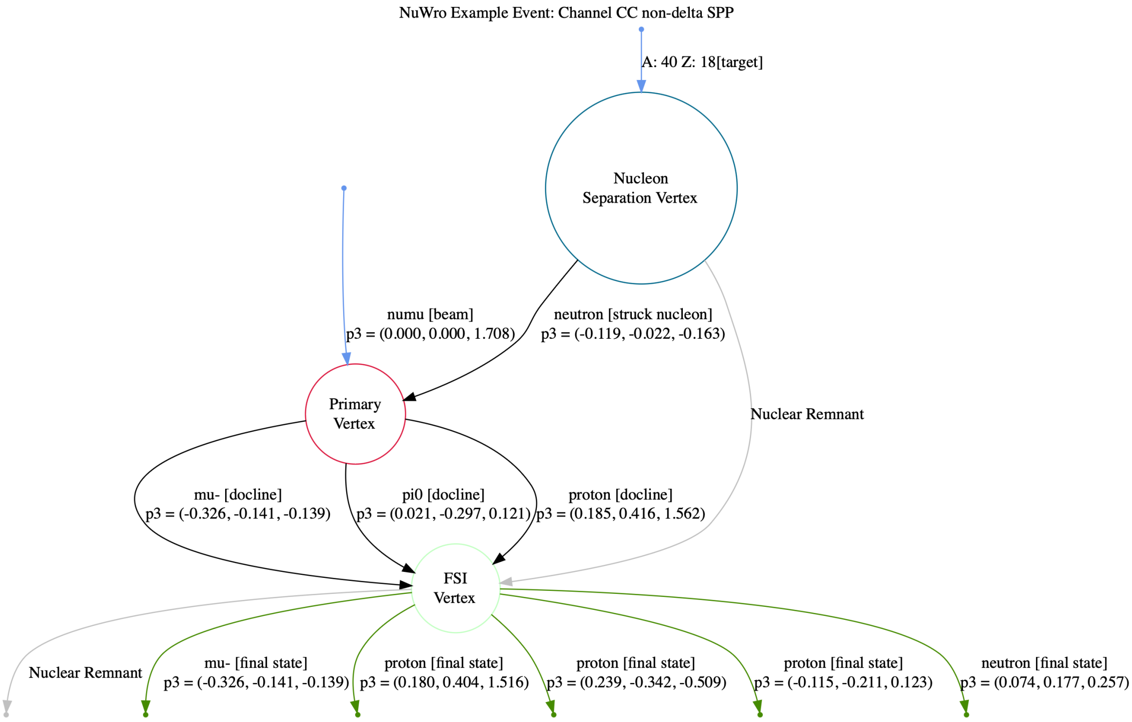

Figure 6: A NuWro event graph in the NuHepMC format.

References

[1] L. Alvarez-Ruso et al., *NuSTEC white paper: Status and challenges of neutrino-nucleus scattering*, Prog. Part. Nucl. Phys. **100**, 1 (2018), doi:10.1016/j.ppnp.2018.01.006 [preprint doi:10.48550/arXiv.1706.03621].

[2] U. Mosel, *Neutrino event generators: Foundation, status and future*, J. Phys. G: Nucl. Part. Phys. **46**, 113001 (2019), doi:10.1088/1361-6471/ab3830 [preprint doi:10.48550/arXiv.1904.11506].

[3] J. Barrow et al., *Summary of workshop on common neutrino event generator tools*, (arXiv preprint) doi:10.48550/arXiv.2008.06566.

[4] J. M. Campbell et al., *Event generators for high-energy physics experiments*, SciPost Phys. **16**, 130 (2024), doi:10.21468/SciPostPhys.16.5.130 [preprint doi:10.48550/arXiv.2203.11110].

[5] A. Aurisano, C. Backhouse, R. Hatcher, N. Mayer, J. Musser, R. Patterson, R. Schroeter and A. Sousa, *The NOvA simulation chain*, J. Phys.: Conf. Ser. **664**, 072002 (2015), doi:10.1088/1742-6596/664/7/072002.

[6] E. L. Snider and G. Petrillo, *LArSoft: Toolkit for simulation, reconstruction and analysis of liquid argon TPC neutrino detectors*, J. Phys.: Conf. Ser. **898**, 042057 (2017), doi:10.1088/1742-6596/898/4/042057.

[7] A. Buckley, J. Butterworth, D. Grellscheid, H. Hoeth, L. Lönnblad, J. Monk, H. Schulz and F. Siegert, *Rivet user manual*, Comput. Phys. Commun. **184**, 2803 (2013), doi:10.1016/j.cpc.2013.05.021 [preprint doi:10.48550/arXiv.1003.0694].

[8] C. Bierlich et al., *Robust independent validation of experiment and theory: Rivet version 3*, SciPost Phys. **8**, 026 (2020), doi:10.21468/SciPostPhys.8.2.026 [preprint doi:10.48550/arXiv.1912.05451].

[9] C. Bierlich, A. Buckley, J. M. Butterworth, C. Gütschow, L. Lönnblad, T. Procter, P. Richardson and Y. Yeh, *Robust independent validation of experiment and theory: Rivet version 4 release note*, SciPost Phys. Codebases 36 (2024), doi:10.21468/SciPostPhysCodeb.36 [preprint doi:10.48550/arXiv.2404.15984].

C. Bierlich, A. Buckley, J. M. Butterworth, C. Gütschow, L. Lönnblad, T. Procter, P. Richardson and Y. Yeh, *Codebase release 4.0 for RIVET*, SciPost Phys. Codebases 36-r4.0 (2024), doi:10.21468/SciPostPhysCodeb.36-r4.0.

[10] J. M. Butterworth, D. Grellscheid, M. Krämer, B. Sarrazin and D. Yallup, *Constraining new physics with collider measurements of Standard Model signatures*, J. High Energy Phys. **03**, 078 (2017), doi:10.1007/JHEP03(2017)078 [preprint doi:10.48550/arXiv.1606.05296].

[11] A. Buckley et al., *Testing new physics models with global comparisons to collider measurements: The Contur toolkit*, SciPost Phys. Core **4**, 013 (2021), doi:10.21468/SciPostPhysCore.4.2.013 [preprint doi:10.48550/arXiv.2102.04377].

[12] J. Bendavid et al., *Les Houches 2017: Physics at TeV colliders Standard Model working group report*, (arXiv preprint) doi:10.48550/arXiv.1803.07977.

[13] J. Alwall et al., *A standard format for Les Houches event files*, Comput. Phys. Commun. **176**, 300 (2007), doi:10.1016/j.cpc.2006.11.010 [preprint doi:10.48550/arXiv.hep-ph/0609017].

[14] E. Bothmann et al., *A standard convention for particle-level Monte Carlo event-variation weights*, SciPost Phys. Core **6**, 007 (2023), doi:10.21468/SciPostPhysCore.6.1.007 [preprint doi:10.48550/arXiv.2203.08230].

[15] E. Bothmann, T. Childers, C. Gütschow, S. Höche, P. Hovland, J. Isaacson, M. Knobbe and R. Latham, *Efficient precision simulation of processes with many-jet final states at the LHC*, Phys. Rev. D **109**, 014013 (2024), doi:10.1103/PhysRevD.109.014013 [preprint doi:10.48550/arXiv.2309.13154].

[16] L. Alvarez-Ruso et al., *Recent highlights from GENIE v3*, Eur. Phys. J. Spec. Top. **230**, 4449 (2021), doi:10.1140/epjs/s11734-021-00295-7 [preprint doi:10.48550/arXiv.2106.09381].

[17] C. Andreopoulos et al., *The GENIE neutrino Monte Carlo generator*, Nucl. Instrum. Methods Phys. Res. A: Accel. Spectrom. Detect. Assoc. Equip. **614**, 87 (2010), doi:10.1016/j.nima.2009.12.009 [preprint doi:10.48550/arXiv.0905.2517].

[18] O. Buss et al., *Transport-theoretical description of nuclear reactions*, Phys. Rep. **512**, 1 (2012), doi:10.1016/j.physrep.2011.12.001 [preprint doi:10.48550/arXiv.1106.1344].

[19] Y. Hayato and L. Pickering, *The NEUT neutrino interaction simulation program library*, Eur. Phys. J. Spec. Top. **230**, 4469 (2021), doi:10.1140/epjs/s11734-021-00287-7 [preprint doi:10.48550/arXiv.2106.15809].

[20] Y. Hayato, *A neutrino interaction simulation program library NEUT*, Acta Phys. Pol. B **40**, 2477 (2009).

[21] T. Golan, J. T. Sobczyk and J. Żmuda, *NuWro: The Wrocław Monte Carlo generator of neutrino interactions*, Nucl. Phys. B - Proc. Suppl. **229**, 499 (2012), doi:10.1016/j.nuclphysbps.2012.09.136.

[22] P. Stowell et al., *NUISANCE: A neutrino cross-section generator tuning and comparison framework*, J. Instrum. **12**, P01016 (2017), doi:10.1088/1748-0221/12/01/p01016 [preprint doi:10.48550/arXiv.1612.07393].

[23] A. Buckley, P. Ilten, D. Konstantinov, L. Lönnblad, J. Monk, W. Pokorski, T. Przedzinski and A. Verbytskyi, *The HepMC3 event record library for Monte Carlo event generators*, Comput. Phys. Commun. **260**, 107310 (2021), doi:10.1016/j.cpc.2020.107310 [preprint doi:10.48550/arXiv.1912.08005].

[24] J. Isaacson, W. I. Jay, A. Lovato, P. A. N. Machado and N. Rocco, *Introducing a novel event generator for electron-nucleus and neutrino-nucleus scattering*, Phys. Rev. D **107**, 033007 (2023), doi:10.1103/PhysRevD.107.033007 [preprint doi:10.48550/arXiv.2205.06378].

[25] S. Gardiner, *Simulating low-energy neutrino interactions with MARLEY*, Comput. Phys. Commun. **269**, 108123 (2021), doi:10.1016/j.cpc.2021.108123 [preprint doi:10.48550/arXiv.2101.11867].

[26] *IEEE standard for floating-point arithmetic*, IEEE, Piscataway, USA, ISBN 9781504459242 (2019), doi:10.1109/IEEESTD.2019.8766229.

[27] Particle Data group: R. L. Workman et al., *Review of particle physics*, Prog. Theor. Exp. Phys. **08**, 083C01 (2022), doi:10.1093/ptep/ptac097.

[28] S. Agostinelli et al., *Geant4 – A simulation toolkit*, Nucl. Instrum. Methods Phys. Res. A: Accel. Spectrom. Detect. Assoc. Equip. **506**, 250 (2003), doi:10.1016/S0168-9002(03)01368-8.

[29] MicroBooNE collaboration: P. Abratenko et al., *Multidifferential cross section measurements of ν_μ-argon quasielasticlike reactions with the MicroBooNE detector*, Phys. Rev. D **108**, 053002 (2023), doi:10.1103/PhysRevD.108.053002 [preprint doi:10.48550/arXiv.2301.03700].

[30] P. Stowell et al., *NUISANCE: A neutrino cross-section generator tuning and comparison framework*, J. Instrum. **12**, P01016 (2017), doi:10.1088/1748-0221/12/01/P01016 [preprint doi:10.48550/arXiv.1612.07393].

[31] S. Gardiner, *Nuclear de-excitations in low-energy charged-current ν_e scattering on ^{40}Ar*, Phys. Rev. C **103**, 044604 (2021), doi:10.1103/PhysRevC.103.044604 [preprint doi:10.48550/arXiv.2010.02393].

[32] D. F. Gatz and L. Smith, *The standard error of a weighted mean concentration – I. Bootstrapping vs other methods*, Atmos. Environ. **29**, 1185 (1995), doi:10.1016/1352-2310(94)00210-C.

[33] D. H. D. West, *Updating mean and variance estimates: An improved method*, Commun. ACM **22**, 532 (1979), doi:10.1145/359146.359153.

[34] G. P. Lepage, *A new algorithm for adaptive multidimensional integration*, J. Comput. Phys. **27**, 192 (1978), doi:10.1016/0021-9991(78)90004-9.

[35] G. P. Lepage, *VEGAS: An adaptive multidimensional integration program*, Cornell University, New York, USA (1980), http://cds.cern.ch/record/123074/files/clns-447.pdf?version=1.

[36] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz, *Event generation with normalizing flows*, Phys. Rev. D **101**, 076002 (2020), doi:10.1103/PhysRevD.101.076002 [preprint doi:10.48550/arXiv.2001.10028].

[37] J. Isaacson, *NuDevTools/HEPReference: v0.1*, Zenodo (2024), doi:10.5281/zenodo.12193254.