A Common Neutrino Event Format: NuHepMC

Luke Pickering, S. Gardiner, J. Isaacson

2024/01/18

NuSTEC CEWG

UKRI Science and Technology Facilities Council

Supported by URF\R1\211661

The Royal Society

State of the NuEvGen…tion

- A number of commonly used event generators:
  - GENIE, ACHILLES, NEUT, NuWro, MARLEY, GiBUU, …
  - Rich physics differences are a strength!

State of the NuEvGen...tion

- A number of commonly used event generators:
  - GENIE, ACHILLES, NEUT, NuWro, MARLEY, GiBUU, ...
  - Rich physics differences are a strength!
- Because we haven't agreed on common interfaces:
  - Comparing a cross-section measurement to a range of event generator predictions is an expert-level task.
    - NUISANCE helps, but maintenance is a 100% unnecessary time sink.

State of the NuEvGen…tion

- A number of commonly used event generators:
  - GENIE, ACHILLES, NEUT, NuWro, MARLEY, GiBUU, …
  - Rich physics differences are a strength!
- Because we haven't agreed on common interfaces:
  - Comparing a cross-section measurement to a range of event generator predictions is an expert-level task.
    - NUISANCE helps, but maintenance is a 100% unnecessary time sink.
  - Entire experiment simulation software stacks are precariously balanced on top of generator-specific APIs.
    - Adding a new bit of metadata to a GENIE event requires a new LArSoft release even if nothing uses that metadata. ROOT object I/O is horrifically inextensible.

State of the NuEvGen…tion

- A number of commonly used event generators:
  - GENIE, ACHILLES, NEUT, NuWro, MARLEY, GiBUU, …
  - Rich physics differences are a strength!
- Because we haven't agreed on common interfaces:
  - Comparing a cross-section measurement to a range of event generator predictions is an expert-level task.
    - NUISANCE helps, but maintenance is a 100% unnecessary time sink.
  - Entire experiment simulation software stacks are precariously balanced on top of generator-specific APIs.
    - Adding a new bit of metadata to a GENIE event requires a new LArSoft release even if nothing uses that metadata. ROOT object I/O is horrifically inextensible.
- Neutrino interaction modelling and event generator development are very difficult problems.

State of the NuEvGen…tion

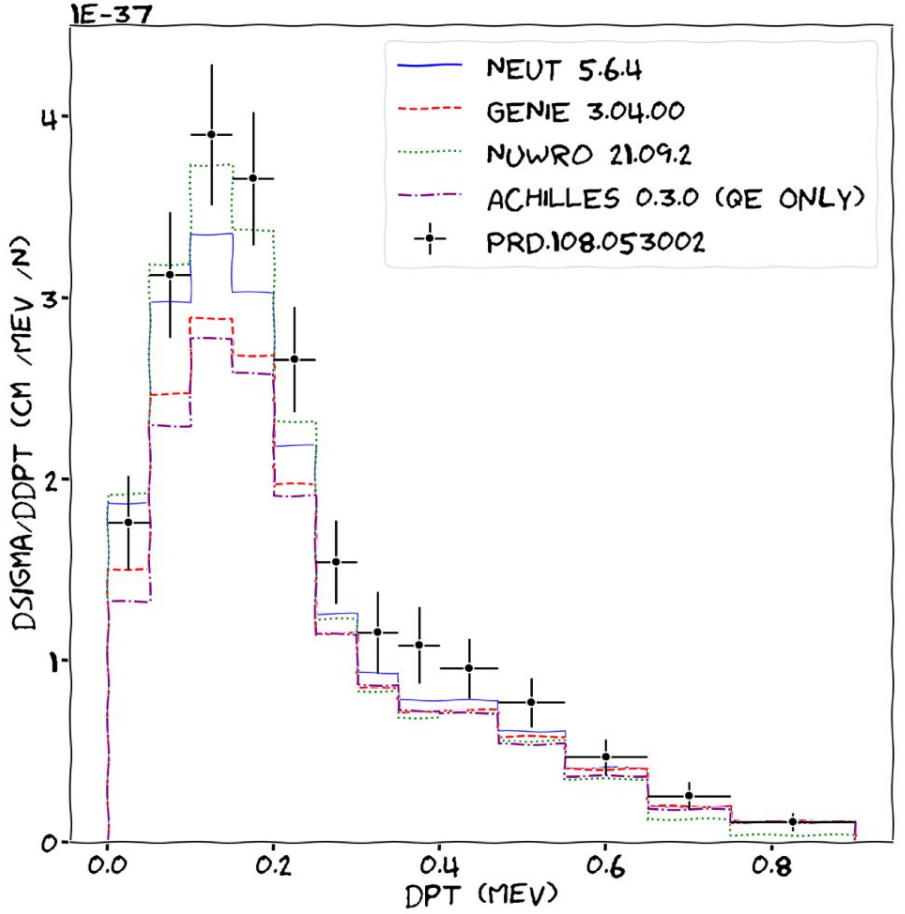
- A number of commonly used event generators:
  - GENIE, ACHILLES, NEUT, NuWro, MARLEY, GiBUU, …
  - Rich physics differences are a strength!
- Because we haven't agreed on common interfaces:
  - Comparing a cross-section measurement to a range of event generator predictions is an expert-level task.
    - NUISANCE helps, but maintenance is a 100% unnecessary time sink.
  - Entire experiment simulation software stacks are precariously balanced on top of generator-specific APIs.
    - Adding a new bit of metadata to a GENIE event requires a new LArSoft release even if nothing uses that metadata. ROOT object I/O is horrifically inextensible.
- Neutrino interaction modelling and event generator development are very difficult problems.
  - **Turning an event vector into a measurement prediction shouldn't be.**

Goal #1

Perform the following steps:

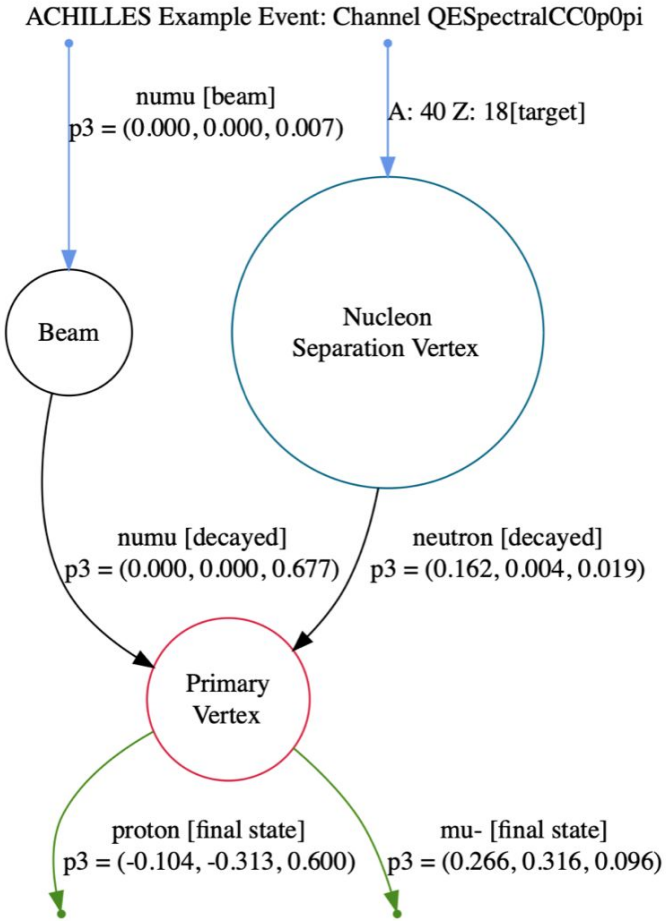
- Make topological event selections
- Make kinematic event projections
- Scale the selected and projected events to a cross section prediction with specified units

on an event vector file without knowing *anything* about how that file was produced.



Goal #2

Be able take a pre-simulated event vector and pass it through an experimental simulation without using any generator-specific code!

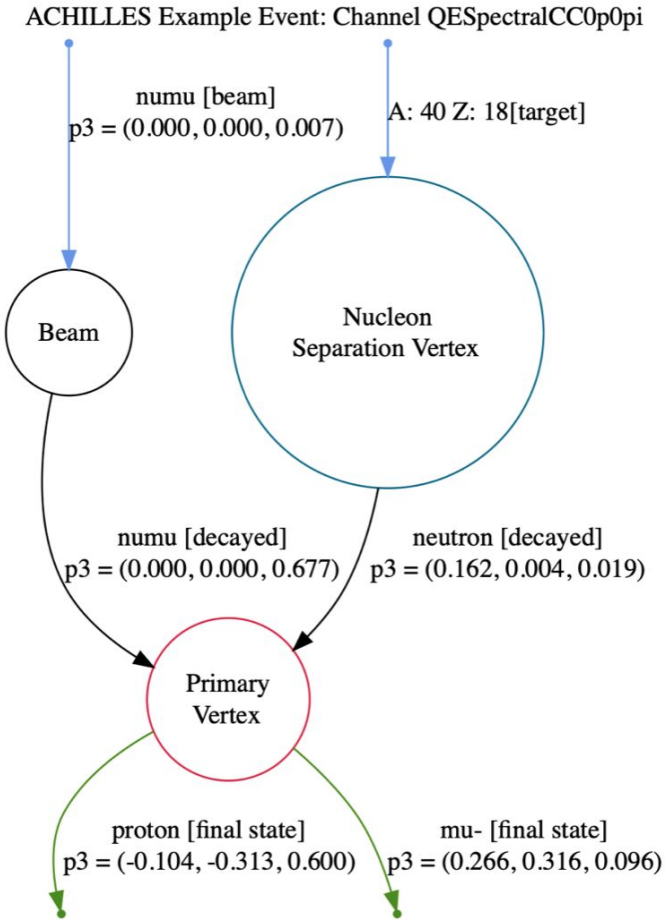


Goal #2

Be able take a pre-simulated event vector and pass it through an experimental simulation without using any generator-specific code!

(caveat 1: At the analysis stage, systematic tools may need to link back to the generator)

(caveat 2: Here we are not stipulating how those events were generated: experiment flux/interaction sim is currently necessarily generator specific)



Some Axioms

- We can make multi-generator event processing simpler and more flexible by agreeing on common interfaces without losing any functionality

Some Axioms

- We can make multi-generator event processing simpler and more flexible by agreeing on common interfaces without losing any functionality
- Such a common interface must not constrain any potential modelling choices

Some Axioms

- We can make multi-generator event processing simpler and more flexible by agreeing on common interfaces without losing any functionality
- Such a common interface must not constrain any potential modelling choices
- The simplest solution that needs the least maintenance is the best

Some Axioms

- We can make multi-generator event processing simpler and more flexible by agreeing on common interfaces without losing any functionality
- Such a common interface must not constrain any potential modelling choices
- The simplest solution that needs the least maintenance is the best
- Event **parsing** is not the computational bottleneck in any compute-intensive tasks (e.g. experiment simulation).
    - If a boatload of extensibility makes event reading off disk a little slower, this isn't a problem

NuHepMC In a Nutshell

First: HepMC3 In a Nutshell

- Our proposal is based on the HepMC3 event format: https://arxiv.org/pdf/1912.08005.pdf
- HepMC3:
  - Generic, extensible event graph library written and maintained by the collider community
  - An active user community
  - C++ & Python language bindings
  - Events can be read/written to ASCII, ROOT, Protobuf files on disk. User-extensible to other on-disk formats
  - Authors are very open to contributions from active users:
    - *e.g.* I contributed the Protobuf format because I wanted a non-ROOT binary event format that could be streamed between event generator and event processor

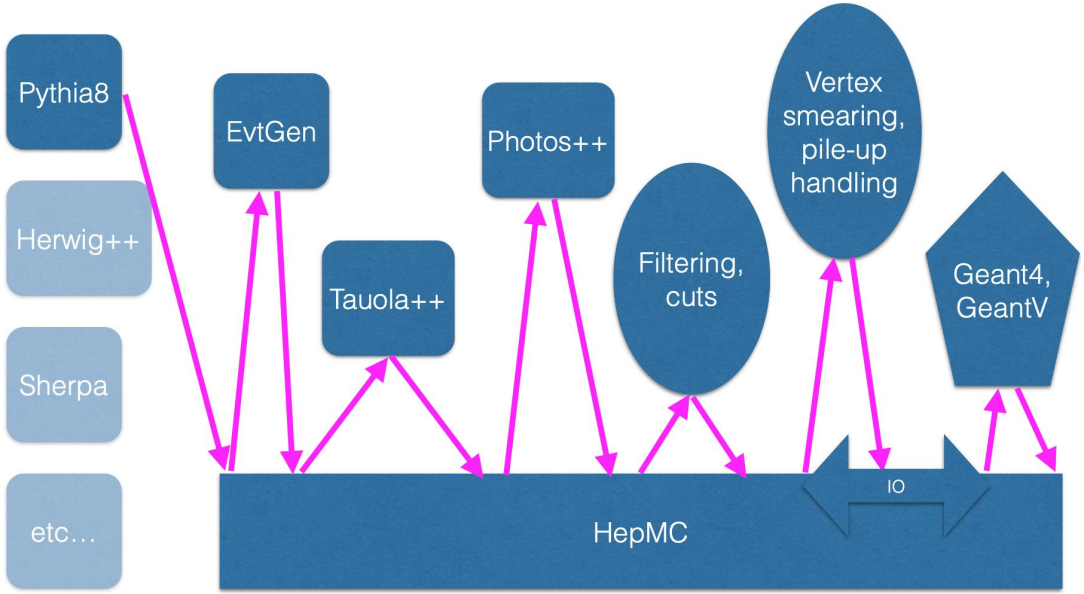
First: HepMC3 In a Nutshell

- Our proposal is based on the HepMC3 event format: https://arxiv.org/pdf/1912.08005.pdf
- HepMC3:
  - Generic, extensible event graph library written and maintained by the collider community
  - An active user community
  - C++ & Python language bindings
  - Events can be read/written to ASCII, ROOT, Protobuf files on disk. User-extensible to other on-disk formats
  - Authors are very open to contributions from active users:
    - *e.g.* I contributed the Protobuf format because I wanted a non-ROOT binary event format that could be streamed between event generator and event processor
- **NuHepMC is a set of conventions for event graphs and associated metadata, it does not require any new code beyond HepMC3.**

First: HepMC3 In a Nutshell

https://indico.cern.ch/event/351898/contributions/827536/attachments/696475/956357/HepMC3Status.pdf

Typical use case



HepMC3 Vectors

- HepMC3 Primitives:
  - GenRunInfo
  - Attribute
  - Event
  - Vertex
  - Particle

```
HepMC::Version 3.02.07
HepMC::Asciiv3-START_EVENT_LISTING
W CV
T GENIE\|3.04.00\|74d03698c18de4e6ae6d7d884f24c69df5986747
A GENIE.XSecTune AR23_20i_00_000
A NuHepMC.AdditionalParticleNumber[-2000010000].Description A dark matter antiparticle use
d in GENIE's boosted dark matter mode
A NuHepMC.AdditionalParticleNumber[-2000010000].Name anti-DM
A NuHepMC.AdditionalParticleNumber[-2000020000].Description Heavy neutral antilepton
A NuHepMC.AdditionalParticleNumber[-2000020000].Name anti-HNL
A NuHepMC.AdditionalParticleNumber[-2000030000].Description Dark matter antiparticle used
in GENIE's Dark Neutrino mode
A NuHepMC.AdditionalParticleNumber[-2000030000].Name Dark-anti-nu
A NuHepMC.AdditionalParticleNumber[2000000001].Description DIS hadronic system before hadr
onization
A NuHepMC.AdditionalParticleNumber[2000000001].Name HadronicSystem
A NuHepMC.AdditionalParticleNumber[2000000002].Description Unmodeled portion of the hadron
ic system
A NuHepMC.AdditionalParticleNumber[2000000002].Name HadronicBlob
A NuHepMC.AdditionalParticleNumber[2000000101].Description Pseudo-particle representing bi
nding energy subtracted from final-state nucleons
A NuHepMC.AdditionalParticleNumber[2000000101].Name Bindino
A NuHepMC.AdditionalParticleNumber[2000000102].Description Pseudo-particle representing Co
ulomb energy subtracted from final-state leptons
A NuHepMC.AdditionalParticleNumber[2000000102].Name Coulobtron
A NuHepMC.AdditionalParticleNumber[2000000200].Description A two-neutron cluster within a
nucleus
A NuHepMC.AdditionalParticleNumber[2000000200].Name NNCluster
A NuHepMC.AdditionalParticleNumber[2000000201].Description A neutron + proton cluster with
in a nucleus
A NuHepMC.AdditionalParticleNumber[2000000201].Name NPCluster
A NuHepMC.AdditionalParticleNumber[2000000202].Description A two-proton cluster within a n
ucleus
A NuHepMC.AdditionalParticleNumber[2000000202].Name PPCluster
A NuHepMC.AdditionalParticleNumber[2000000300].Description An undecayed cluster of nucleon
s
A NuHepMC.AdditionalParticleNumber[2000000300].Name CompNuclCluster
A NuHepMC.AdditionalParticleNumber[2000010000].Description A dark matter particle used in
GENIE's boosted dark matter mode
A NuHepMC.AdditionalParticleNumber[2000010000].Name DM
A NuHepMC.AdditionalParticleNumber[2000010001].Description Mediator particle used in GENIE
's Boosted Dark Matter mode
A NuHepMC.AdditionalParticleNumber[2000010001].Name DM-Mediator
A NuHepMC.AdditionalParticleNumber[2000020000].Description Heavy neutral lepton
A NuHepMC.AdditionalParticleNumber[2000020000].Name HNL
A NuHepMC.AdditionalParticleNumber[2000030000].Description Dark matter particle used in GE
NIE's Dark Neutrino mode
A NuHepMC.AdditionalParticleNumber[2000030000].Name Dark-nu
A NuHepMC.AdditionalParticleNumber[2000030001].Description Mediator particle used in GENIE
's Dark Neutrino mode
A NuHepMC.AdditionalParticleNumber[2000030001].Name Dark-Mediator
A NuHepMC.AdditionalParticleNumber[91].Description PYTHIA cluster pseudo-particle
A NuHepMC.AdditionalParticleNumber[91].Name PCluster
```

HepMC3 Vectors

- HepMC3 Primitives:
  - GenRunInfo
  - Attribute
  - Event
  - Vertex
  - Particle
- **GenRunInfo**: once-per-vector metadata about the generator run that produced the contained events.

```
HepMC::Version 3.02.07
HepMC::Asciiv3-START_EVENT_LISTING
W CV
T GENIE\|3.04.00\|74d03698c18de4e6ae6d7d884f24c69df5986747
A GENIE.XSecTune AR23_20i_00_000
A NuHepMC.AdditionalParticleNumber[-2000010000].Description A dark matter antiparticle use
d in GENIE's boosted dark matter mode
A NuHepMC.AdditionalParticleNumber[-2000010000].Name anti-DM
A NuHepMC.AdditionalParticleNumber[-2000020000].Description Heavy neutral antilepton
A NuHepMC.AdditionalParticleNumber[-2000020000].Name anti-HNL
A NuHepMC.AdditionalParticleNumber[-2000030000].Description Dark matter antiparticle used
in GENIE's Dark Neutrino mode
A NuHepMC.AdditionalParticleNumber[-2000030000].Name Dark-anti-nu
A NuHepMC.AdditionalParticleNumber[2000000001].Description DIS hadronic system before hadr
onization
A NuHepMC.AdditionalParticleNumber[2000000001].Name HadronicSystem
A NuHepMC.AdditionalParticleNumber[2000000002].Description Unmodeled portion of the hadron
ic system
A NuHepMC.AdditionalParticleNumber[2000000002].Name HadronicBlob
A NuHepMC.AdditionalParticleNumber[2000000101].Description Pseudo-particle representing bi
nding energy subtracted from final-state nucleons
A NuHepMC.AdditionalParticleNumber[2000000101].Name Bindino
A NuHepMC.AdditionalParticleNumber[2000000102].Description Pseudo-particle representing Co
ulomb energy subtracted from final-state leptons
A NuHepMC.AdditionalParticleNumber[2000000102].Name Coulobtron
A NuHepMC.AdditionalParticleNumber[2000000200].Description A two-neutron cluster within a
nucleus
A NuHepMC.AdditionalParticleNumber[2000000200].Name NNCluster
A NuHepMC.AdditionalParticleNumber[2000000201].Description A neutron + proton cluster with
in a nucleus
A NuHepMC.AdditionalParticleNumber[2000000201].Name NPCluster
A NuHepMC.AdditionalParticleNumber[2000000202].Description A two-proton cluster within a n
ucleus
A NuHepMC.AdditionalParticleNumber[2000000202].Name PPCluster
A NuHepMC.AdditionalParticleNumber[2000000300].Description An undecayed cluster of nucleon
s
A NuHepMC.AdditionalParticleNumber[2000000300].Name CompNuclCluster
A NuHepMC.AdditionalParticleNumber[2000010000].Description A dark matter particle used in
GENIE's boosted dark matter mode
A NuHepMC.AdditionalParticleNumber[2000010000].Name DM
A NuHepMC.AdditionalParticleNumber[2000010001].Description Mediator particle used in GENIE
's Boosted Dark Matter mode
A NuHepMC.AdditionalParticleNumber[2000010001].Name DM-Mediator
A NuHepMC.AdditionalParticleNumber[2000020000].Description Heavy neutral lepton
A NuHepMC.AdditionalParticleNumber[2000020000].Name HNL
A NuHepMC.AdditionalParticleNumber[2000030000].Description Dark matter particle used in GE
NIE's Dark Neutrino mode
A NuHepMC.AdditionalParticleNumber[2000030000].Name Dark-nu
A NuHepMC.AdditionalParticleNumber[2000030001].Description Mediator particle used in GENIE
's Dark Neutrino mode
A NuHepMC.AdditionalParticleNumber[2000030001].Name Dark-Mediator
A NuHepMC.AdditionalParticleNumber[91].Description PYTHIA cluster pseudo-particle
A NuHepMC.AdditionalParticleNumber[91].Name PCluster
```

HepMC3 Vectors

- HepMC3 Primitives:
  - GenRunInfo
  - Attribute
  - Event
  - Vertex
  - Particle
- **GenRunInfo**: once-per-vector metadata about the generator run that produced the contained events.
- **Attribute**: Extensible Key-Value metadata that can be attached to all other primitive types:
  - Facilities for typed attributes: int/double/string, vectors thereof etc...

```
HepMC::Version 3.02.07
HepMC::Asciiv3-START_EVENT_LISTING
W CV
T GENIE\|3.04.00\|74d03698c18de4e6ae6d7d884f24c69df5986747
A GENIE.XSecTune AR23_20i_00_000
A NuHepMC.AdditionalParticleNumber[-2000010000].Description A dark matter antiparticle use
d in GENIE's boosted dark matter mode
A NuHepMC.AdditionalParticleNumber[-2000010000].Name anti-DM
A NuHepMC.AdditionalParticleNumber[-2000020000].Description Heavy neutral antilepton
A NuHepMC.AdditionalParticleNumber[-2000020000].Name anti-HNL
A NuHepMC.AdditionalParticleNumber[-2000030000].Description Dark matter antiparticle used
in GENIE's Dark Neutrino mode
A NuHepMC.AdditionalParticleNumber[-2000030000].Name Dark-anti-nu
A NuHepMC.AdditionalParticleNumber[2000000001].Description DIS hadronic system before hadr
onization
A NuHepMC.AdditionalParticleNumber[2000000001].Name HadronicSystem
A NuHepMC.AdditionalParticleNumber[2000000002].Description Unmodeled portion of the hadron
ic system
A NuHepMC.AdditionalParticleNumber[2000000002].Name HadronicBlob
A NuHepMC.AdditionalParticleNumber[2000000101].Description Pseudo-particle representing bi
nding energy subtracted from final-state nucleons
A NuHepMC.AdditionalParticleNumber[2000000101].Name Bindino
A NuHepMC.AdditionalParticleNumber[2000000102].Description Pseudo-particle representing Co
ulomb energy subtracted from final-state leptons
A NuHepMC.AdditionalParticleNumber[2000000102].Name Coulobtron
A NuHepMC.AdditionalParticleNumber[2000000200].Description A two-neutron cluster within a
nucleus
A NuHepMC.AdditionalParticleNumber[2000000200].Name NNCluster
A NuHepMC.AdditionalParticleNumber[2000000201].Description A neutron + proton cluster with
in a nucleus
A NuHepMC.AdditionalParticleNumber[2000000201].Name NPCluster
A NuHepMC.AdditionalParticleNumber[2000000202].Description A two-proton cluster within a n
ucleus
A NuHepMC.AdditionalParticleNumber[2000000202].Name PPCluster
A NuHepMC.AdditionalParticleNumber[2000000300].Description An undecayed cluster of nucleon
s
A NuHepMC.AdditionalParticleNumber[2000000300].Name CompNuclCluster
A NuHepMC.AdditionalParticleNumber[2000010000].Description A dark matter particle used in
GENIE's boosted dark matter mode
A NuHepMC.AdditionalParticleNumber[2000010000].Name DM
A NuHepMC.AdditionalParticleNumber[2000010001].Description Mediator particle used in GENIE
's Boosted Dark Matter mode
A NuHepMC.AdditionalParticleNumber[2000010001].Name DM-Mediator
A NuHepMC.AdditionalParticleNumber[2000020000].Description Heavy neutral lepton
A NuHepMC.AdditionalParticleNumber[2000020000].Name HNL
A NuHepMC.AdditionalParticleNumber[2000030000].Description Dark matter particle used in GE
NIE's Dark Neutrino mode
A NuHepMC.AdditionalParticleNumber[2000030000].Name Dark-nu
A NuHepMC.AdditionalParticleNumber[2000030001].Description Mediator particle used in GENIE
's Dark Neutrino mode
A NuHepMC.AdditionalParticleNumber[2000030001].Name Dark-Mediator
A NuHepMC.AdditionalParticleNumber[91].Description PYTHIA cluster pseudo-particle
A NuHepMC.AdditionalParticleNumber[91].Name PCluster
```

Hep

- Hep
  -
  -
  -
  -
  -
- **Ge**
  abc
  the
- **Att**
  me
  typ
  -

Critical: Adding/removing attributes or even new types of attribute doesn't change the binary readability of an event vector!

> *e.g.* Reading in event vectors produced by different versions of NEUT, doesn't require any changes to user code.

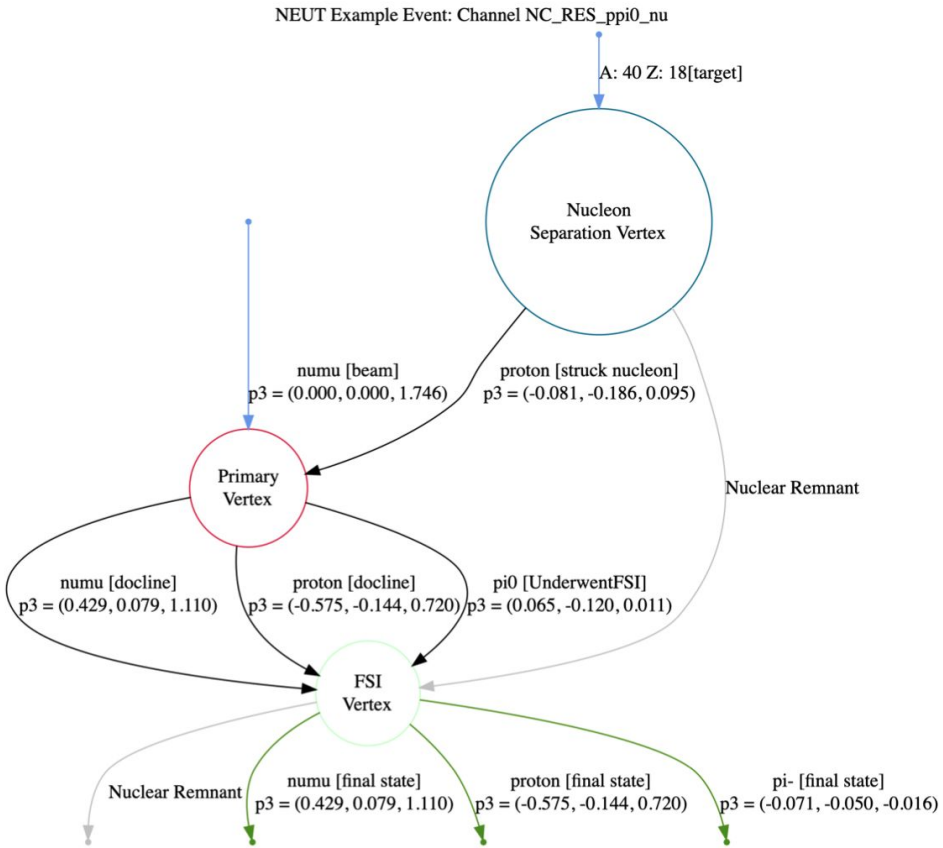
This might seem like a minor detail, but it **dramatically** lowers the technical barrier and maintenance load to producing and using event vectors.

1. I know that I can read a vector produced 10 years ago and one produced today with the same code.
2. I know that I can read a vector produced by Josh without checking the exact version of ACHILLES and building the same one
3. I can reasonably expect that my analysis will run on output from newer versions of a generator without modification

The coupling between user code and exact generator versions becomes very weak and this is surprisingly powerful.

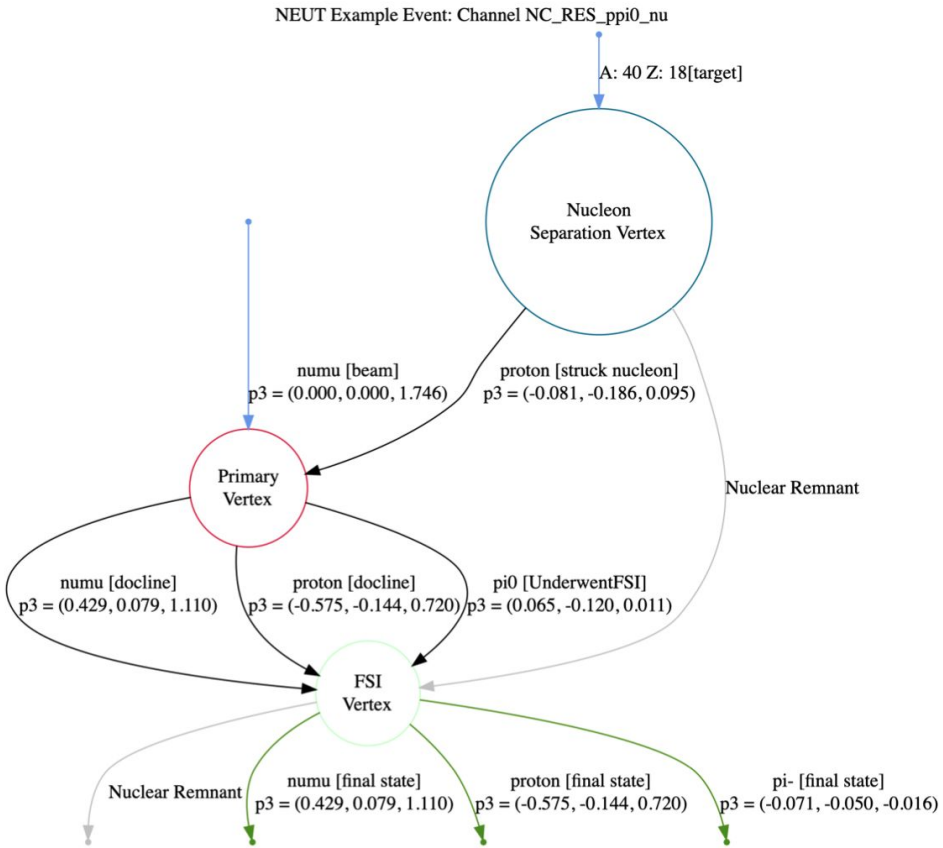
HepMC3 Event Graphs

- **Events** contain metadata **attributes** and an event graph of vertices and particles



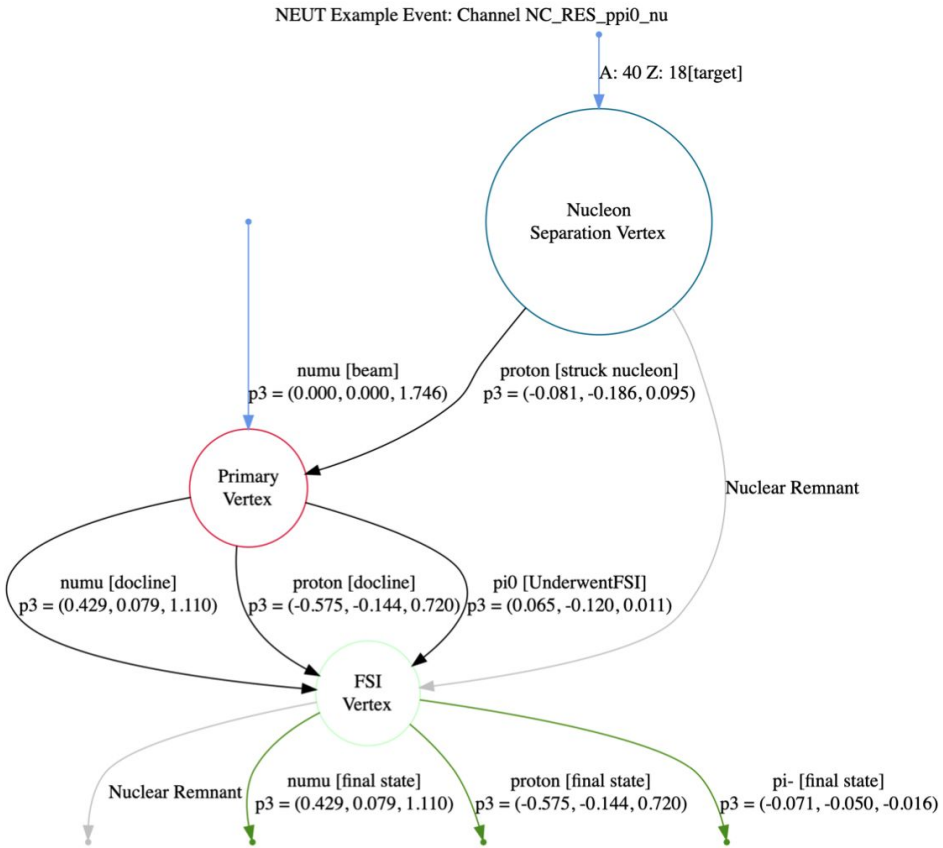
HepMC3 Event Graphs

- **Events** contain metadata **attributes** and an event graph of vertices and particles
- **Vertices** have a position, a status code and incoming and outgoing particles (and optional metadata)



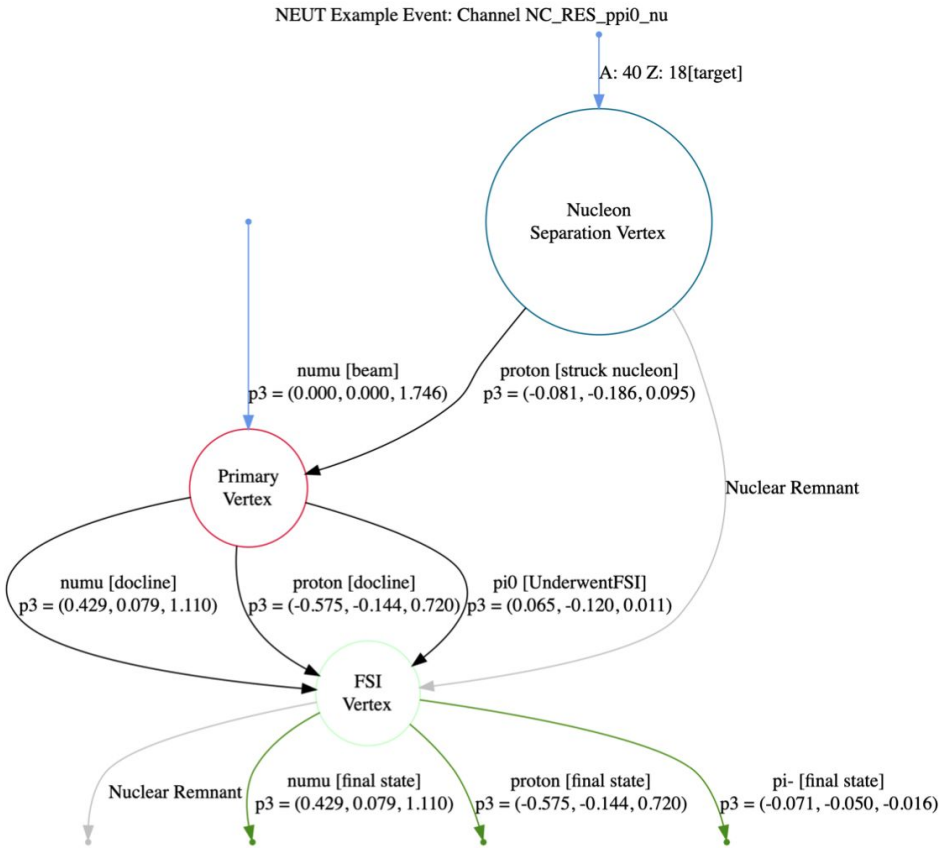
HepMC3 Event Graphs

- **Events** contain metadata **attributes** and an event graph of vertices and particles
- **Vertices** have a position, a status code and incoming and outgoing particles (and optional metadata)
- **Particles** have a status code, a PID, and a four-momenta (and optional metadata)



HepMC3 Event Graphs

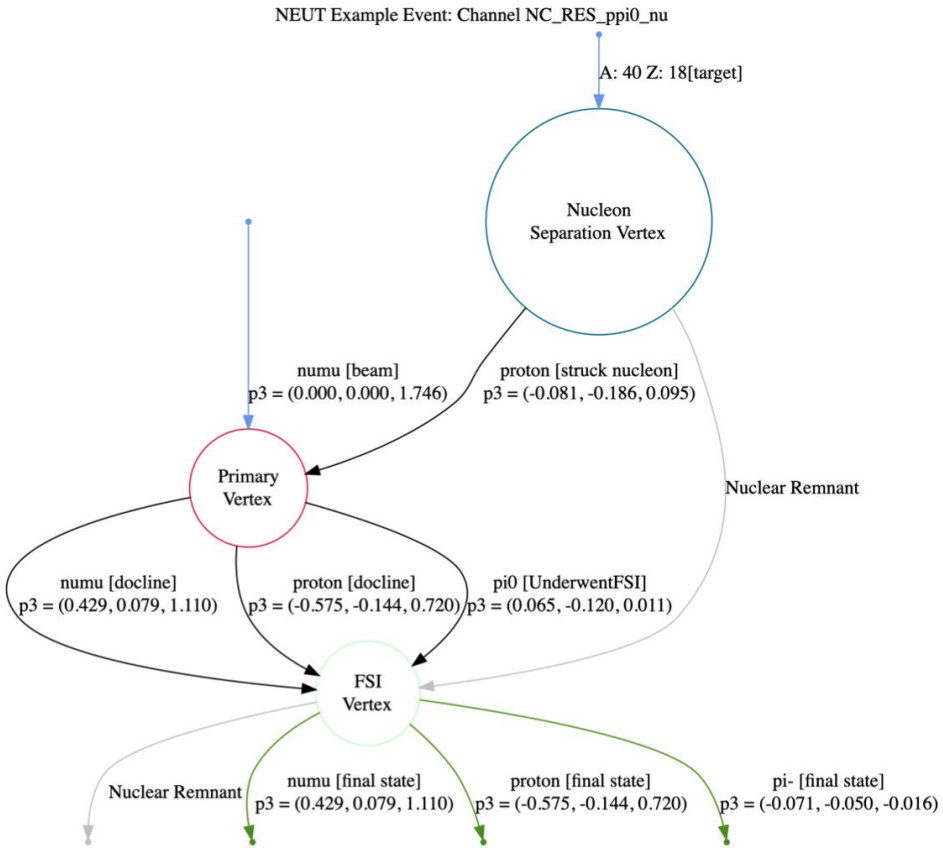
- **Events** contain metadata **attributes** and an event graph of vertices and particles
- **Vertices** have a position, a status code and incoming and outgoing particles (and optional metadata)
- **Particles** have a status code, a PID, and a four-momenta (and optional metadata)



HepMC3 Event Graphs

- **Events** contain metadata **attributes** and an event graph of vertices and particles
- **Vertices** have a position, a status code and incoming and outgoing particles (and optional metadata)
- **Particles** have a status code, a PID, and a four-momenta (and optional metadata)

HepMC3 itself places minimal structural constraints on any of these.



NuHepMC in a Nutshell

- Full working spec on Arxiv: https://arxiv.org/pdf/2310.13211.pdf
- More 'living' version on Github: https://github.com/NuHepMC/Spec
- Individual specifications take the form:

  <Component>.<Category>.<Index>

  - <Component>:
    - **G:** GenRunInfo
    - **E:** Event
    - **V:** Vertex
    - **P:** Particle
  - <Category>:
    - **R:** Requirement – Must be implemented by all
    - **C:** Convention – Recommend specifications
    - **S:** Suggestion – Optional details

NuHepMC in a Nutshell

- Full working spec on Arxiv: https://arxiv.org/pdf/2310.13211.pdf
- Individual specifications take the form:

  <Component>.<Category>.<Index>

- Where:
  - <Component>:
    - <Category>:
      - **R:** Requirement – Must be implemented by all
      - **C:** Convention – Recommend specifications
      - **S:** Suggestion – Optional details

G.R.4 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:

Some Important Technical Details

Conventions

G.C.1 Signaling Followed Conventions:

To signal to a user 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.

Conventions

G.C.1 Signaling Followed Conventions:

To signal to a user 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.

```
A NuHepMC.Citations.Generator.DOI 10.1016/j.nima.2009.12.009 10.1140/epjs/s1
95-7
A NuHepMC.Citations.Process[100].DOI 10.1103/PhysRevD.79.053003
A NuHepMC.Citations.Process[200].DOI 10.1103/PhysRevC.70.055503 10.1103/Phys
02
A NuHepMC.Citations.Process[300].DOI 10.1103/PhysRevD.91.073004
A NuHepMC.Citations.Process[400].DOI 10.1103/PhysRevD.76.113004
A NuHepMC.Citations.Process[600].DOI 10.1103/PhysRevD.50.3085 10.1103/PhysRe
A NuHepMC.Conventions E.C.1 E.C.2 E.C.4 E.C.5 G.C.1 G.C.4 G.C.6 G.S.2 P.C.1
A NuHepMC.ParticleStatusIDs 0 1 2 4 11 21 22 23 24 25 26 27 28 29
A NuHepMC.ParticleStatusInfo[0].Description Not meaningful
A NuHepMC.ParticleStatusInfo[0].Name Not defined
```

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 recommended for high-level categorization of processes. Even if an implementation uses the convention in Table 1, it must still adhere to G.R.4.

Conventions

C++ uses https://github.com/NuHepMC/cpputils

```cpp
#include "NuHepMC/HepMC3Features.hxx"

#include "HepMC3/GenEvent.h"
#include "HepMC3/ReaderFactory.h"
#include "NuHepMC/ReaderUtils.hxx"
#include <iostream>

int main(int argc, char const *argv[]) {
  auto rdr = HepMC3::deduce_reader(argv[1]);
  HepMC3::GenEvent evt;
  rdr->read_event(evt);
  for (auto c : NuHepMC::GC1::ReadConventions(evt.run_info())) {
    std::cout << c << std::endl;
  }
}
```

```
[genbox:nustectalk $ ./gc1 ../achilles.dune.5E6.pb
E.C.1
E.C.4
E.C.5
G.C.2
G.C.4
G.C.6
V.C.1
```

Python

```python
from pyHepMC3 import HepMC3 as hm
import sys

file = hm.deduce_reader(sys.argv[1])
evt = hm.GenEvent()
while not file.failed():
  file.read_event(evt)
  convs = hm.VectorStringAttribute()
  convs.from_string(evt.run_info()
    .attribute_as_string("NuHepMC.Conventions"))

  for i in convs.value():
    print(i)
  break
```

```
[genbox:nustectalk $ python3 ./gc1.py ../achilles.dune.5E6.pb
G.C.2
G.C.4
G.C.6
E.C.1
E.C.4
E.C.5
V.C.1
```

Conventions

C++ uses https://github.com/NuHepMC/cpputils

```cpp
#include "NuHepMC/HepMC3Features.hxx"

#include "HepMC3/GenEvent.h"
#include "HepMC3/ReaderFactory.h"
#include "NuHepMC/ReaderUtils.hxx"
#include <iostream>

int main(int argc, char const *argv[]) {
  auto rdr = HepMC3::deduce_reader(argv[1]);
  HepMC3::GenEvent evt;
  rdr->read_event(evt);
  for (auto c : NuHepMC::GC1::ReadConventions(evt.run_info())) {
    std::cout << c << std::endl;
  }
}
```

Python

```python
from pyHepMC3 import HepMC3 as hm
import sys

file = hm.deduce_reader(sys.argv[1])
evt = hm.GenEvent()
while not file.failed():
  file.read_event(evt)
  convs = hm.VectorStringAttribute()
  convs.from_string(evt.run_info()
    .attribute_as_string("NuHepMC.Conventions"))

  for i in convs.value():
    print(i)
  break
```

```
[genbox:nustectalk $ ./gc1 ../achilles.dune.5E6.pb
E.C.1
E.C.4
E.C.5
G.C.2
G.C.4
G.C.6
V.C.1
```

Example code is just an advert for how you can use minimally dependent tools to dump information out of a NuHepMC vector without knowing anything about the generator that produced it! It might not be the best way to achieve the results in an actual analysis.

pyHepMC3 can be installed with pip!

Process Metadata

G.R.4 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.C.1).

Pr

This is both an important specific detail and an example of our general approach to not violating our axiom:

- **Such a common interface must not constrain any potential modelling choices**

We cannot specify hard scatter channels in the spec without constraining the physics describable by the format. Instead we require that implementations communicate their choices in a pre-specified and programmatically accessible manner.

This is both an important specific detail and an example of our general approach to not violating our axiom:

- **Such a common interface must not constrain any potential modelling choices**

We cannot specify hard scatter channels in the spec without constraining the physics describable by the format. Instead we require that implementations communicate their choices in a pre-specified and programmatically accessible manner.

- Current implementations usually hard code definitions in the source code, which are then not extractable from a given vector file.
- **What does NEUT mode 41 mean? Read the source code!**
- Trivial to define new process ids for BSM models or new processes or sub-processes.

Where appropriate, we also provide conventions that can be adopted that do provide additional standardisation.

Process Metadata

```python
from pyHepMC3 import HepMC3 as hm
import sys

file = hm.deduce_reader(sys.argv[1])
evt = hm.GenEvent()
procid = hm.IntAttribute()
convn = hm.StringAttribute()
convd = hm.StringAttribute()

evti = 1

while not file.failed():
  file.read_event(evt)
  procid.from_string(evt.attribute_as_string("signal_process_id"))
  convn.from_string(evt.run_info()
    .attribute_as_string("NuHepMC.ProcessInfo[%s].Name" % procid.value()))
  convd.from_string(evt.run_info()
    .attribute_as_string("NuHepMC.ProcessInfo[%s].Description" % procid.value()))

  print("Event: %s, ProcId: %s = %s,\n  %s" % (evti,procid.value(),
    convn.value(),convd.value()))

  if evti >= 10:
    break
  evti = evti + 1

```

```
genbox:nustectalk $ python3 ./gr4.py ../nustec_dune_numu_cc_AR23_20i_00_000-fixed.hepmc3
Event: 1, ProcId: 400 = RES-Weak[CC],
  genie::BergerSehgalRESPXSec2014/NoPauliBlock
Event: 2, ProcId: 200 = QES-Weak[CC],
  genie::KovalenkoQELCharmPXSec/Default genie::NievesQELCCPXSec/ZExp
Event: 3, ProcId: 600 = DIS-Weak[CC],
  genie::AivazisCharmPXSecLO/CC-Default genie::KNOTunedQPMDISPXSec/Default
Event: 4, ProcId: 400 = RES-Weak[CC],
  genie::BergerSehgalRESPXSec2014/NoPauliBlock
Event: 5, ProcId: 600 = DIS-Weak[CC],
  genie::AivazisCharmPXSecLO/CC-Default genie::KNOTunedQPMDISPXSec/Default
Event: 6, ProcId: 600 = DIS-Weak[CC],
  genie::AivazisCharmPXSecLO/CC-Default genie::KNOTunedQPMDISPXSec/Default
Event: 7, ProcId: 600 = DIS-Weak[CC],
  genie::AivazisCharmPXSecLO/CC-Default genie::KNOTunedQPMDISPXSec/Default
Event: 8, ProcId: 300 = MEC-Weak[CC],
  genie::SuSAv2MECPXSec/Default
Event: 9, ProcId: 200 = QES-Weak[CC],
  genie::KovalenkoQELCharmPXSec/Default genie::NievesQELCCPXSec/ZExp
Event: 10, ProcId: 400 = RES-Weak[CC],
  genie::BergerSehgalRESPXSec2014/NoPauliBlock
```

- The python interface can be a little clunky. But users can and will improve it!
- There is also a scikit-hep interface that has some different pros/cons

Process Metadata

Interpreting process IDs generally and correctly will *always* need generator-specific code:

- But with NuHepMC, we don't need to *link* to generator binaries, it can all be implemented in the analysis code in a compartmentalised way and can access all the information about a process ID from the GenRunInfo metadata

Process Metadata

Interpreting process IDs generally and correctly will *always* need generator-specific code:

- But with NuHepMC, we don't need to *link* to generator binaries, it can all be implemented in the analysis code in a compartmentalised way and can access all the information about a process ID from the GenRunInfo metadata

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 recommended for high-level categorization of processes. Even if an implementation uses the convention in Table 1, it must still adhere to G.R.4.

Identifier	Process
100-199	Low-Energy 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-	Other process types

Table 1: Process ID ranges for various process categories.

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 may be reserved for electromagnetic interactions in neutrino event generators that include them.

Process Metadata

Interpreting process IDs generally and correctly will *always* need generator-specific code:

- But with NuHepMC, we don't need to *link* to generator binaries, it can all be implemented in the analysis code in a compartmentalised way and can access all the information about a process ID from the GenRunInfo metadata

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 recommended for high-level categorization of processes. Even if an implementation uses the convention in Table 1, it must still adhere to G.R.4.

Identifier	Process
100-199	Low-Energy 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-	Other process types

Table 1: Process ID ranges for various process categories.

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 may be reserved for electromagnetic interactions in neutrino event generators that include them.

Process Metadata

Interpreting process IDs generally and correctly will *always* need generator-specific code:

- But with NuHepMC, we don't need to *link* to generator binaries, it can all be implemented in the analysis code in a compartmentalised way and can access all the information about a process ID from the GenRunInfo metadata

If implementations signal the use of E.C.1 and this breakdown in granular enough for a given analysis, then it can be fully generator agnostic again!

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 recommended for high-level categorization of processes. Even if an implementation uses the convention in Table 1, it must still adhere to G.R.4.

Identifier	Process
100-199	Low-Energy 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-	Other process types

Table 1: Process ID ranges for various process categories.

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 may be reserved for electromagnetic interactions in neutrino event generators that include them.

Status Codes

- The general approach of requiring that implementations signal the definitions of their hard scatter process codes in a specified metadata format is repeated for **Vertex** status codes and **Particle** status codes.

V.R.1 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

Status Codes

- The general approach of requiring that implementations signal the definitions of their hard scatter process codes in a specified metadata format is repeated for **Vertex** status codes and **Particle** status codes.

V.R.1 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

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 signalled via the mechanism described in G.C.1, then status code 21 need not be included in the implementation of G.R.5.

Status Codes

- The general approach of requiring that implementations signal the definitions of their hard scatter process codes in a specified metadata format is repeated for **Vertex** status codes and **Particle** status codes.

P.R.1 PARTICLE STATUS CODES:
We extend the HepMC3 definition of `HepMC3::GenParticle::status` slightly 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.

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

Citations

G.C.6 Citation Metadata:

Modelling components implemented based on published work should always be fully cited. The `HepMC3::GenRunInfo` should contain at least one `HepMC3::VectorStringAttribute` for each relevant modelling 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. We suggest using `<Comp>=Generator` for specifying the main citation for the interaction generator and `<Comp>=Process[<ID>]` for individual processes. For common reference formats in the HEP field, we suggest some common values for the `<Type>` field:

- `"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.

We hope that automatic bibliography generation tools using this metadata will be built.

Citations

```python
from pyHepMC3 import HepMC3 as hm
import sys

file = hm.deduce_reader(sys.argv[1])
evt = hm.GenEvent()
dois = hm.VectorStringAttribute()

while not file.failed():
  file.read_event(evt)
  for a in evt.run_info().attribute_names():
    if a.startswith("NuHepMC.Citations"):
      if a.endswith("DOI"):
        dois.from_string(evt.run_info().attribute_as_string(a))
        print(a)
        for doi in dois.value():
          print("  https://www.doi.org/%s" % doi)

  break
```

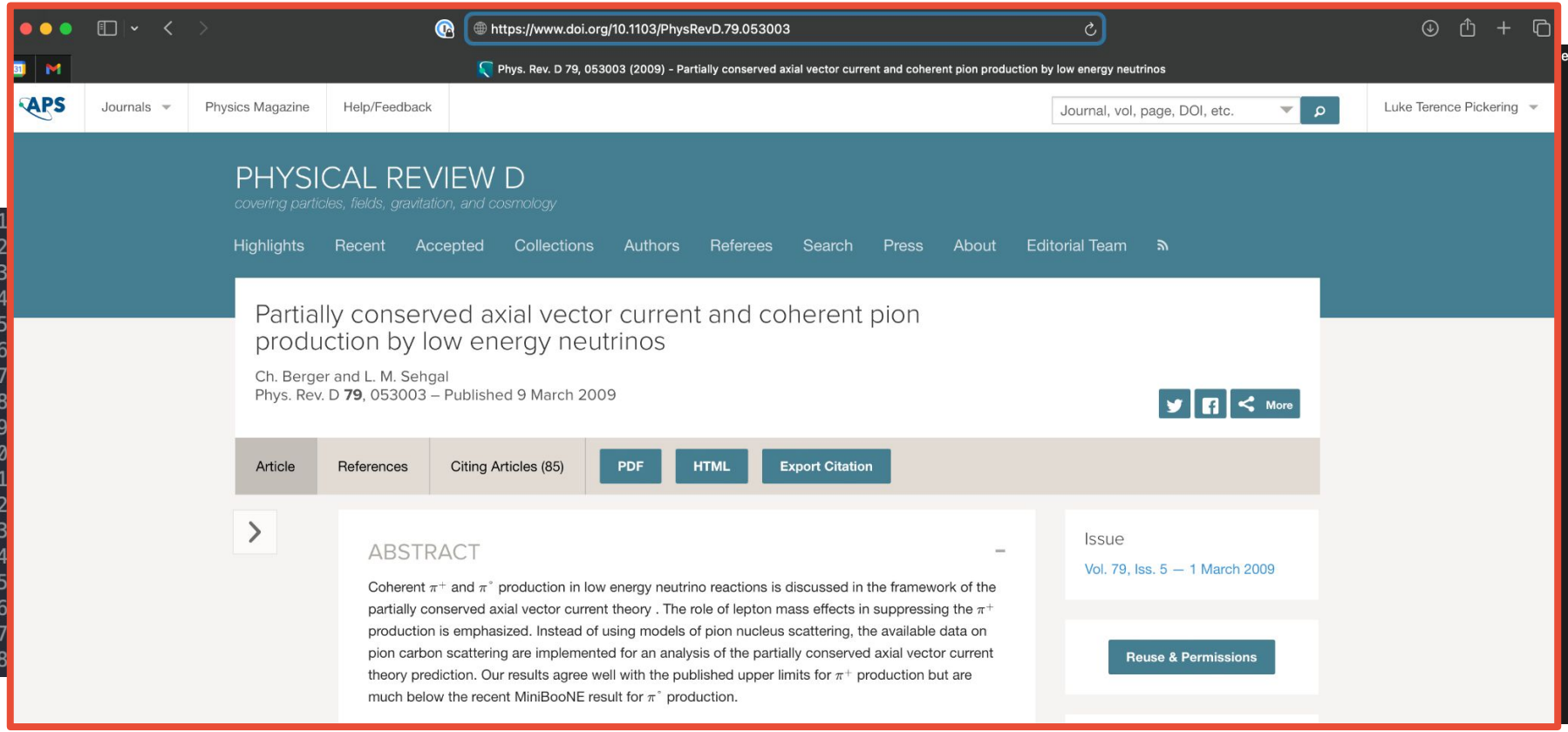
```
genbox:nustectalk $ python3 ./gc6.py ../nustec_dune_numu_cc_AR23_20i_00_000-fixed.hepmc3
NuHepMC.Citations.Generator.DOI
  https://www.doi.org/10.1016/j.nima.2009.12.009
  https://www.doi.org/10.1140/epjs/s11734-021-00295-7
NuHepMC.Citations.Process[100].DOI
  https://www.doi.org/10.1016/j.nima.2009.12.009
  https://www.doi.org/10.1140/epjs/s11734-021-00295-7
  https://www.doi.org/10.1103/PhysRevD.79.053003
NuHepMC.Citations.Process[200].DOI
  https://www.doi.org/10.1016/j.nima.2009.12.009
  https://www.doi.org/10.1140/epjs/s11734-021-00295-7
  https://www.doi.org/10.1103/PhysRevD.79.053003
  https://www.doi.org/10.1103/PhysRevC.70.055503
  https://www.doi.org/10.1103/PhysRevC.72.019902
NuHepMC.Citations.Process[300].DOI
  https://www.doi.org/10.1016/j.nima.2009.12.009
  https://www.doi.org/10.1140/epjs/s11734-021-00295-7
  https://www.doi.org/10.1103/PhysRevD.79.053003
  https://www.doi.org/10.1103/PhysRevC.70.055503
  https://www.doi.org/10.1103/PhysRevC.72.019902
  https://www.doi.org/10.1103/PhysRevD.91.073004
NuHepMC.Citations.Process[400].DOI
  https://www.doi.org/10.1016/j.nima.2009.12.009
  https://www.doi.org/10.1140/epjs/s11734-021-00295-7
  https://www.doi.org/10.1103/PhysRevD.79.053003
  https://www.doi.org/10.1103/PhysRevC.70.055503
  https://www.doi.org/10.1103/PhysRevC.72.019902
  https://www.doi.org/10.1103/PhysRevD.91.073004
  https://www.doi.org/10.1103/PhysRevD.76.113004
NuHepMC.Citations.Process[600].DOI
  https://www.doi.org/10.1016/j.nima.2009.12.009
  https://www.doi.org/10.1140/epjs/s11734-021-00295-7
  https://www.doi.org/10.1103/PhysRevD.79.053003
  https://www.doi.org/10.1103/PhysRevC.70.055503
  https://www.doi.org/10.1103/PhysRevC.72.019902
  https://www.doi.org/10.1103/PhysRevD.91.073004
  https://www.doi.org/10.1103/PhysRevD.76.113004
  https://www.doi.org/10.1103/PhysRevD.50.3085
  https://www.doi.org/10.1103/PhysRevD.65.033002
```

G.C.6 Citation Metadata:

Modelling components implemented h

- "DOI" might contain one or more u

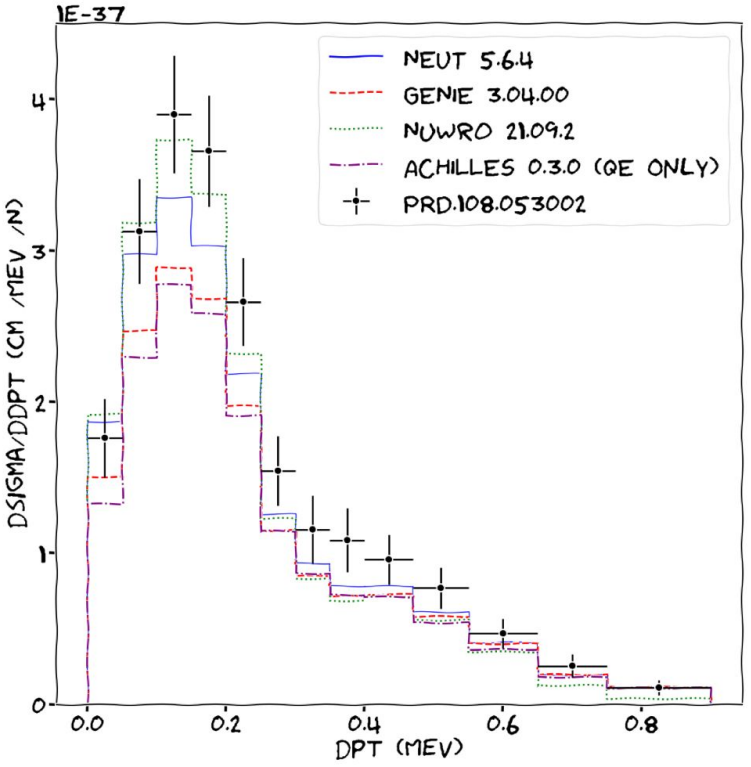
We hope that automatic bibliography generation tools using this metadata will be built.



We hope that automatic bibliography generation tools using this metadata will be built.

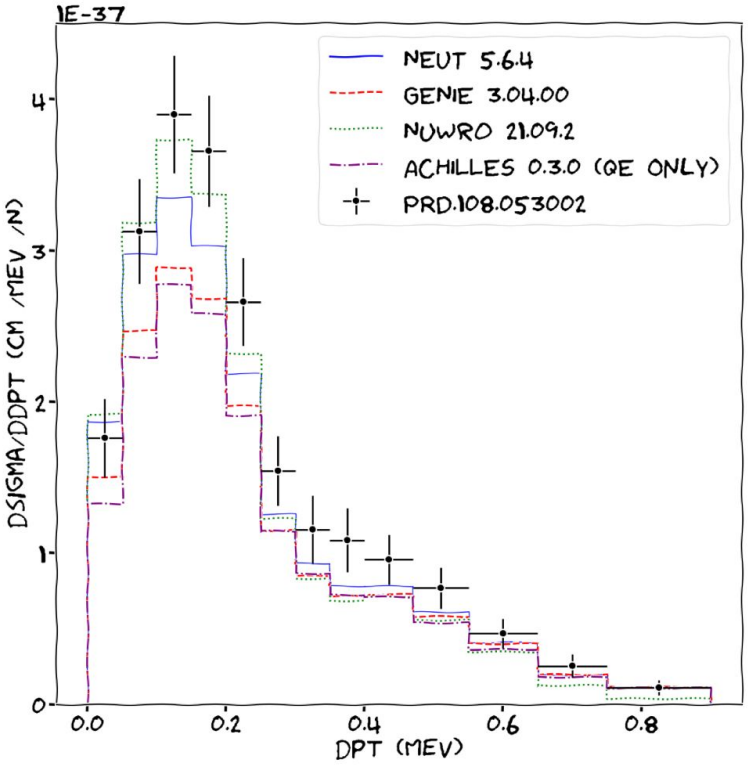
Scaling to a Cross Section

- When you run a generator for a comparison to an xs measurement, you generally ask for a number of events from a given model configuration, with a given flux *shape*.
- Measurements are usually published as a flux-averaged cross-section
- You can think of this as the expected distribution of the measurement per neutrino per target*.



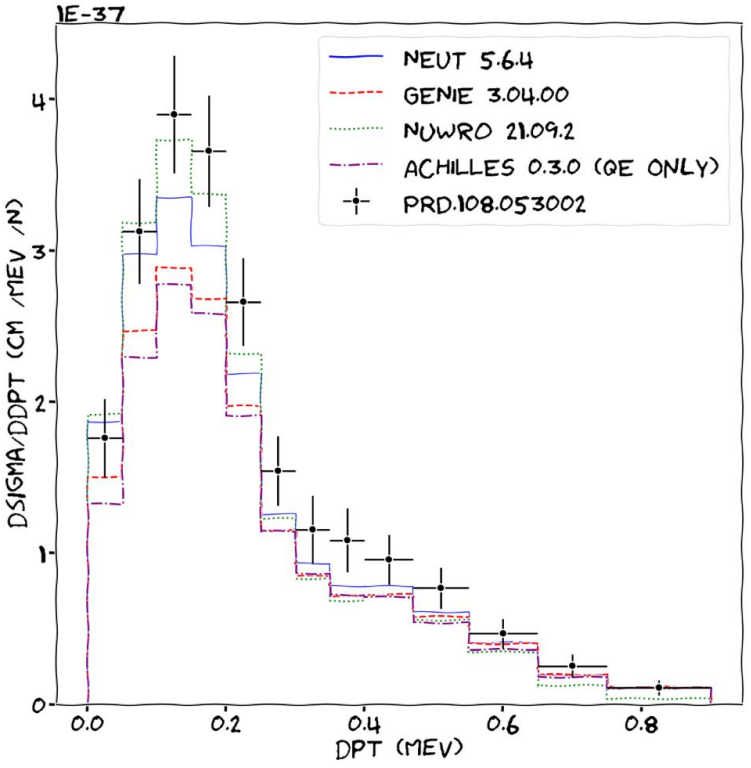
Scaling to a Cross Section

- When you run a generator for a comparison to an xs measurement, you generally ask for a number of events from a given model configuration, with a given flux *shape*.
- Measurements are usually published as a flux-averaged cross-section
- You can think of this as the expected distribution of the measurement per neutrino per target*.
    - * Often per target Nucleon†



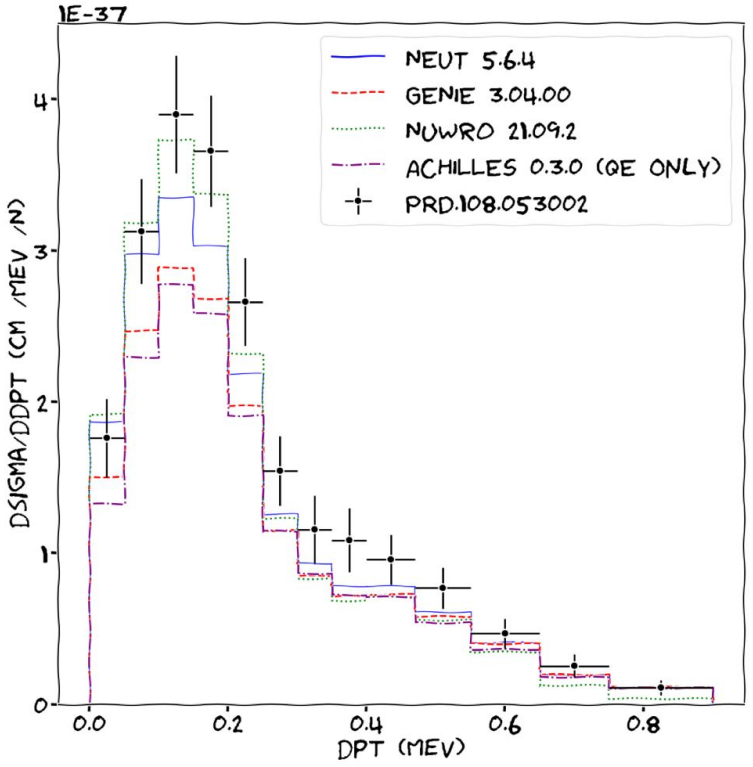
Scaling to a Cross Section

- When you run a generator for a comparison to an xs measurement, you generally ask for a number of events from a given model configuration, with a given flux *shape*.
- Measurements are usually published as a flux-averaged cross-section
- You can think of this as the expected distribution of the measurement per neutrino per target*.
  - * Often per target Nucleon†
  - † there are also a number of different conventions within this convention…



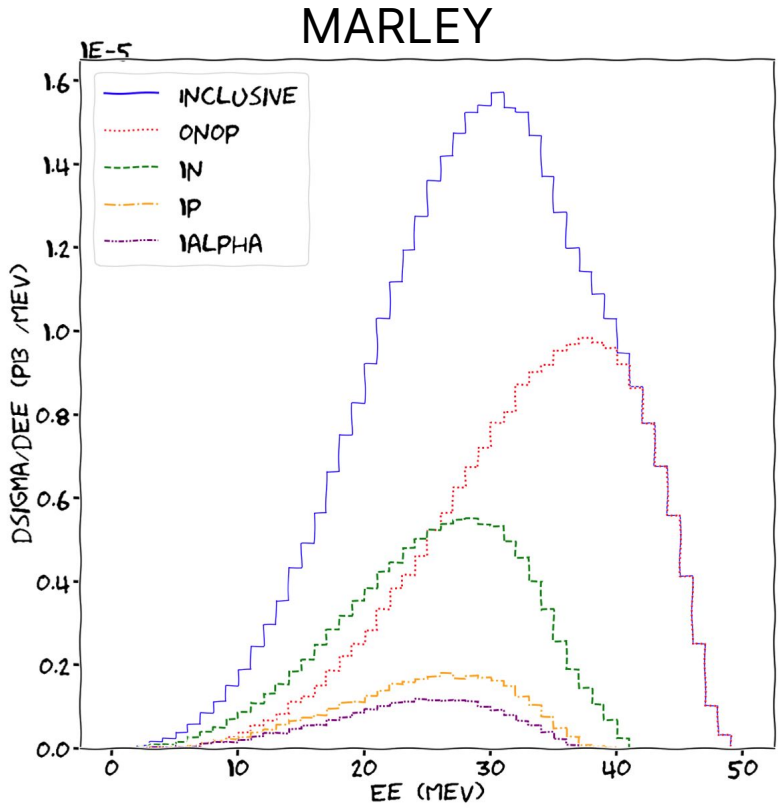
Scaling to a Cross Section

- Historically each generator have provided the scale factor to go from a requested event rate to the ***Flux-Averaged Total Cross Section*** prediction differently:
  - One of the *main* uses of NUISANCE within the community is handling these differences 'transparently' for users
  - A significant amount of NUISANCE maintenance time (and bugs) have been in this process.
  - Not all generators know this scale factor at the start of a run, some calculate it as they go.



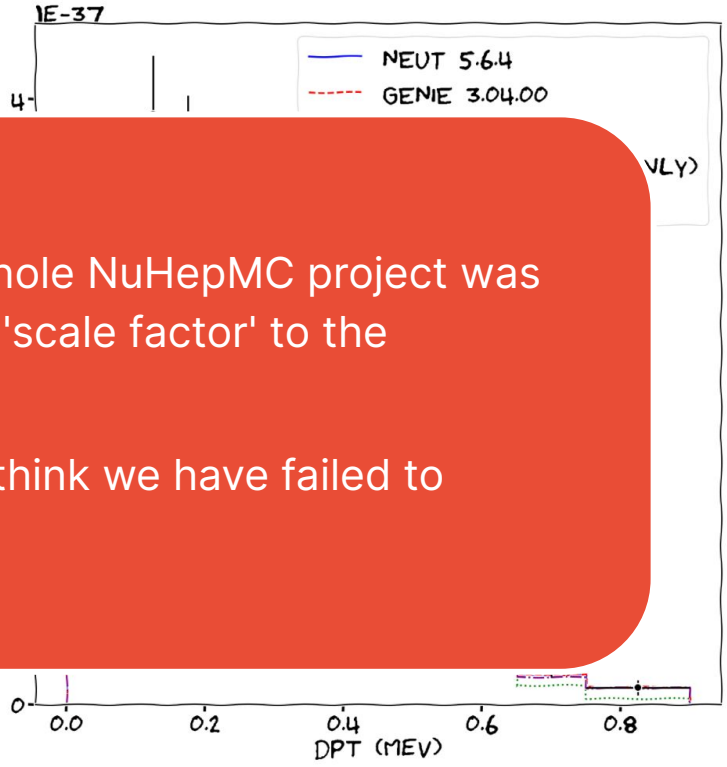
Scaling to a Cross Section

- Historically each generator have provided the scale factor to go from a requested event rate to the ***Flux-Averaged Total Cross Section*** prediction differently:
  - One of the *main* uses of NUISANCE within the community is handling these differences 'transparently' for users
  - A significant amount of NUISANCE maintenance time (and bugs) have been in this process.
  - Not all generators know this scale factor at the start of a run, some calculate it as they go.



Scaling to a Cross Section

- Historically each generator have provided the scale factor to go from a requested event rate to the ***Flux***
  predictio
  - One
    the
    'trar
  - A si
    mai
    proc
  - Not
    star



- One of my main goals with this whole NuHepMC project was the offload the production of this 'scale factor' to the generator/event format.
- If we cannot get this right, then I think we have failed to provide a useful common format.

Scaling to a Cross Section

- This is so central, that we devote an entire section to the mathematical demonstration of the importance of the ***FATX*** and a general derivation of it for an arbitrary flux and target complexity

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 B. 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.5.

Scaling to a Cross Section

G.C.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 prediction of the flux-averaged cross-section—an experimentally-accessible quantity. Details on the definition of this quantity are given in Sec. 3.1.

- Some generators know the FATX for a given configuration before they start spitting out events (GENIE, NEUT, NuWro*).
  - It is calculated for simple single-target, single-species measurements like:

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

Where A(Eν) is the flux distribution, and σ is the total cross-section as a function of the neutrino energy for the given model configuration.

*NuWro uses a weighted test run before writing out events to pre-estimate the FATX

Scaling to a Cross Section

- Other generators (ACHILLES) calculate it as they go, this obviates the need for 'splines' (GENIE lingo for total cross section tables) as an explicit input but produces weighted output events that require a little more care at analysis time.

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. This makes implementing G.C.5 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 can then use the best estimate provided with the last generated event to correctly scale an event rate to a cross-section prediction.

Scaling to a Cross Section

- Other generators (ACHI
  'splines' (GENIE lingo for
  weighted, or non-equipr
  time.

E.C.4 Estimated Flux-A
Some simulations bu
$\langle\sigma\rangle$ as they run. Thi
an alternative, the
`GenEvent::cross_se`
A user can then use
correctly scale an ev

```
[genbox:nustectalk $ ./gc1 ../achilles.dune.5E6.pb
E.C.1
E.C.4
E.C.5
G.C.2
G.C.4
G.C.6
V.C.1
[genbox:nustectalk $ ./gc1 ../neut_DUNE_numu_BS1pi_LFG_1.21.pb.gz
E.C.1
E.C.2
G.C.1
G.C.2
G.C.4
G.C.5
G.C.7
P.C.1
P.C.2
V.C.1
```

Cross Section Units

G.C.4 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, $10^{-38}\,\mathrm{cm}^2$ per nucleon is common, but not ubiquitous. We want to provide a sensible recommended default while preserving the flexibility for an implementation to signal a different choice. One or both of the following `HepMC3::StringAttributes` may be included on the `HepMC3::GenRunInfo` to indicate the cross section units used within a vector.

- `"NuHepMC.Units.CrossSection.Unit"`. Possible values of the attribute are not restricted, but we reserve the meanings of the following:
  - `"pb"`: Picobarns or $10^{-36}\,\mathrm{cm}^2$. This is our recommended default.
  - `"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].
  - `"1e-38 cm2"`: The choice of $10^{-38}\,\mathrm{cm}^2$ in this option is the most frequent in the neutrino literature.
- `"NuHepMC.Units.CrossSection.TargetScale"`. Possible values of the attribute are not restricted, but we reserve the meanings of the following for ease of compatibility with existing conventions:
  - `"PerTargetAtom"`: Our recommendation. Choosing "atom" rather than "nucleus" in this context removes ambiguity when considering neutrino interactions with atomic electrons.
  - `"PerTargetMolecule"`: Sometimes used for hydrocarbon- and water-target measurements.
  - `"PerTargetNucleon"`: A common choice in the literature.
  - `"PerTargetMolecularNucleon"`: Another common choice in the literature and in existing neutrino event generators. We recommend against implementations using this scheme.

Cross Section Units

G.C.4 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, $10^{-38}\,\mathrm{cm}^2$ per nucleon is common, but not ubiquitous. We want to provide a sensible recommended default while preserving the flexibility for an implementation to signal a different choice. One or both of the following `HepMC3::StringAttributes` may be included on the `HepMC3::GenRunInfo` to indicate the cross section units used within a vector.

- `"NuHepMC.Units.CrossSection.Unit"`. Possible values of the attribute are not restricted, but we reserve the meanings of the following:
  - `"pb"`: Picobarns or $10^{-36}\,\mathrm{cm}^2$. This is our recommended default.
  - `"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].
  - `"1e-38 cm2"`: The choice of $10^{-38}\,\mathrm{cm}^2$ in this option is the most frequent in the neutrino literature.

- `"NuHepMC.Units.CrossSection.TargetScale"`. Possible values of the attribute are not restricted, but we reserve the meanings of the following for ease of compatibility with existing conventions:
  - `"PerTargetAtom"`: Our recommendation. Choosing "atom" rather than "nucleus" in this context removes ambiguity when considering neutrino interactions with atomic electrons.
  - `"PerTargetMolecule"`: Sometimes used for hydrocarbon- and water-target measurements.
  - `"PerTargetNucleon"`: A common choice in the literature.
  - `"PerTargetMolecularNucleon"`: Another common choice in the literature and in existing neutrino event generators. We recommend against implementations using this scheme.

Cross Section Units

This is another good example of our general approach to standardising existing conventions.

1. Specify free-form string attributes that generators can put whatever they want in. **Fully extensible.**
2. Reserve the meaning for some values that we expect to cover most current uses so that automatic processing of the vast majority of generator outputs can be achieved without restricting the evolution of community conventions and best practices

→ If a generator needs to output something non-standard, that's allowed, but analyses will necessarily have to adapt to those differences.

→ Such unforeseen cases can be implemented in community tools rapidly, without requiring a new release of this specification.

Scaling to a Cross Section

- The details here are involved and I'm sure I've gone into them too much already...
- The upshot is:

Scaling to a Cross Section

- The details here are involved and I'm sure I've gone into them too much already...

- The upshot is:
  - We think that it is critically important that we require enough metadata to go from a NuHepMC file of unknown provenance, to a total cross-section prediction without any generator specific code

Scaling to a Cross Section

- The details here are involved and I'm sure I've gone into them too much already...

- The upshot is:
  - We think that it is critically important that we require enough metadata to go from a NuHepMC file of unknown provenance, to a total cross-section prediction without any generator specific code
  - We think we have done that generally enough, but want eyes on 👀!

Scaling to a Cross Section

- The details here are involved and I'm sure I've gone into them too much already...

- The upshot is:
  - We think that it is critically important that we require enough metadata to go from a NuHepMC file of unknown provenance, to a total cross-section prediction without any generator specific code
  - We think we have done that generally enough, but want eyes on 👀!

We will provide tools that deal with the details of how this overall scale factor is calculated

- much like NUISANCE currently does, but we shouldn't require users to use a monolithic framework to calculate such a fundamental quantity for interaction cross-section predictions.

Scaling to a CrossSection

```cpp
#include "NuHepMC/CrossSectionUtils.hxx"
#include "NuHepMC/ReaderUtils.hxx"

int main(int argc, char const *argv[]) {
  NuHepMC::CrossSection::GetFATX(
   argv[1],
   NuHepMC::CrossSection::Units::XSUnits::pb,
   NuHepMC::CrossSection::Units::XSTargetScale::PerTargetAtom);
}
```

```
[genbox:nustectalk $ NuHepMCXSTool ../dune_argon_sf_10mega.nuwro.pb.gz
 -- FATX from GC5 = 1.05628(0.026407*40, u: [pb,PerTargetNucleon] -> [pb, PerTargetAtom])
[genbox:nustectalk $ NuHepMCXSTool ../neut_DUNE_numu_BS1pi_LFG_1.21.pb.gz
 -- FATX from GC5 = 1.22629(0.0306573*40, u: [pb,PerTargetMolecularNucleon] -> [pb, PerTargetAtom])
[genbox:nustectalk $ NuHepMCXSTool ../achilles.dune.5E6.pb
 -- FATX from EC4 = 0.117866(0.117866*1, u: [pb,PerTargetAtom] -> [pb, PerTargetAtom])
[genbox:nustectalk $ NuHepMCXSTool ../nustec_dune_numu_cc_AR23_20i_00_000-fixed.hepmc3
 -- FATX from EC2 = 0.877573(0.877573*1, u: [pb,PerTargetAtom] -> [pb, PerTargetAtom])
```

Scaling to a CrossSection

```cpp
#include "NuHepMC/CrossSectionUtils.hxx"
#include "NuHepMC/ReaderUtils.hxx"

int main(int argc, char const *argv[]) {
  NuHepMC::CrossSection::GetFATX(
   argv[1],
   NuHepMC::CrossSection::Units::XSUnits::pb,
   NuHepMC::CrossSection::Units::XSTargetScale::PerTargetAtom);
}
```

```
[genbox:nustectalk $ NuHepMCXSTool ../dune_argon_sf_10mega.nuwro.pb.gz
 -- FATX from GC5 = 1.05628(0.026407*40, u: [pb,PerTargetNucleon] -> [pb, PerTargetAtom])
[genbox:nustectalk $ NuHepMCXSTool ../neut_DUNE_numu_BS1pi_LFG_1.21.pb.gz
 -- FATX from GC5 = 1.22629(0.0306573*40, u: [pb,PerTargetMolecularNucleon] -> [pb, PerTargetAtom])
[genbox:nustectalk $ NuHepMCXSTool ../achilles.dune.5E6.pb
 -- FATX from EC4 = 0.117866(0.117866*1, u: [pb,PerTargetAtom] -> [pb, PerTargetAtom])
[genbox:nustectalk $ NuHepMCXSTool ../nustec_dune_numu_cc_AR23_20i_00_000-fixed.hepmc3
 -- FATX from EC2 = 0.877573(0.877573*1, u: [pb,PerTargetAtom] -> [pb, PerTargetAtom])
```

Scaling to a CrossSection

```cpp
#include "NuHepMC/CrossSectionUtils.hxx"
#include "NuHepMC/ReaderUtils.hxx"

int main(int argc, char const *argv[]) {
  NuHepMC::CrossSection::GetFATX(
    argv[1],
    NuHepMC::CrossSection::Units::XSUnits::pb,
    NuHepMC::CrossSection::Units::XSTargetScale::PerTargetAtom);
}
```

```
[genbox:nustectalk $ NuHepMCXSTool ../dune_argon_sf_10mega.nuwro.pb.gz
 -- FATX from GC5 = 1.05628(0.026407*40, u: [pb,PerTargetNucleon] -> [pb, PerTargetAtom])
[genbox:nustectalk $ NuHepMCXSTool ../neut_DUNE_numu_BS1pi_LFG_1.21.pb.gz
 -- FATX from GC5 = 1.22629(0.0306573*40, u: [pb,PerTargetMolecularNucleon] -> [pb, PerTargetAtom])
[genbox:nustectalk $ NuHepMCXSTool ../achilles.dune.5E6.pb
 -- FATX from EC4 = 0.117866(0.117866*1, u: [pb,PerTargetAtom] -> [pb, PerTargetAtom])
[genbox:nustectalk $ NuHepMCXSTool ../nustec_dune_numu_cc_AR23_20i_00_000-fixed.hepmc3
 -- FATX from EC2 = 0.877573(0.877573*1, u: [pb,PerTargetAtom] -> [pb, PerTargetAtom])
```

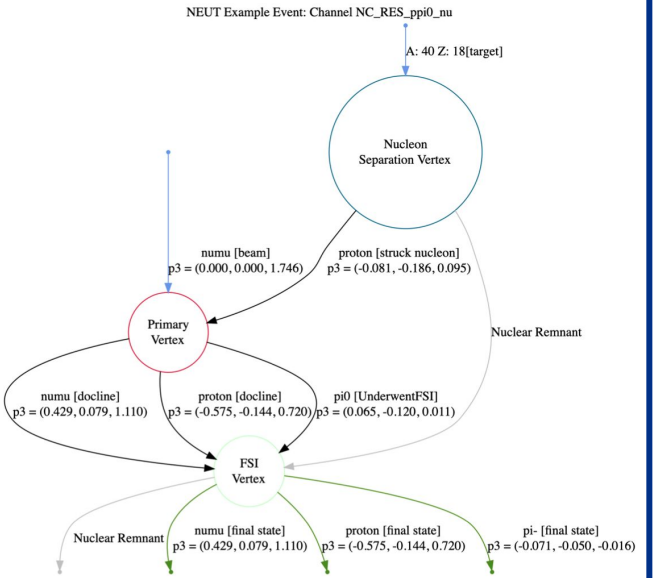
Other Bits and Bobs…

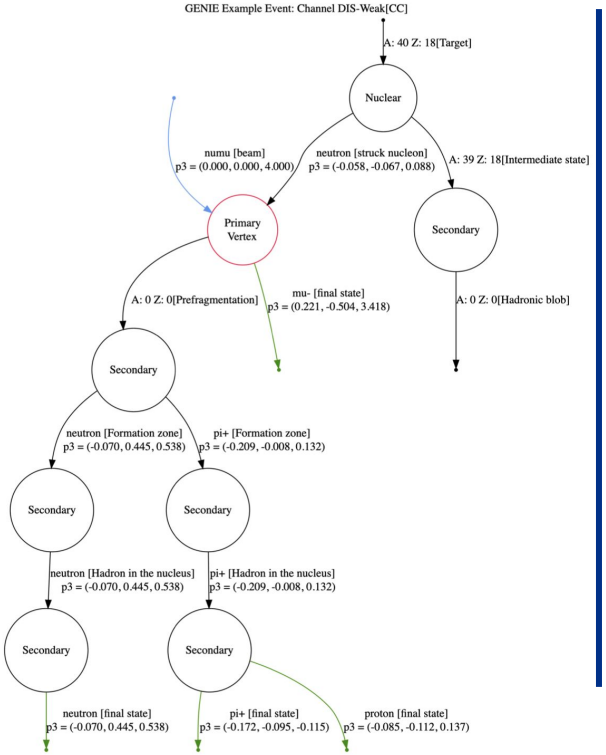
We have tried to think of, implement, and test many bits of important metadata that I will not go into the details of here.
Some important things that I haven't gone through:

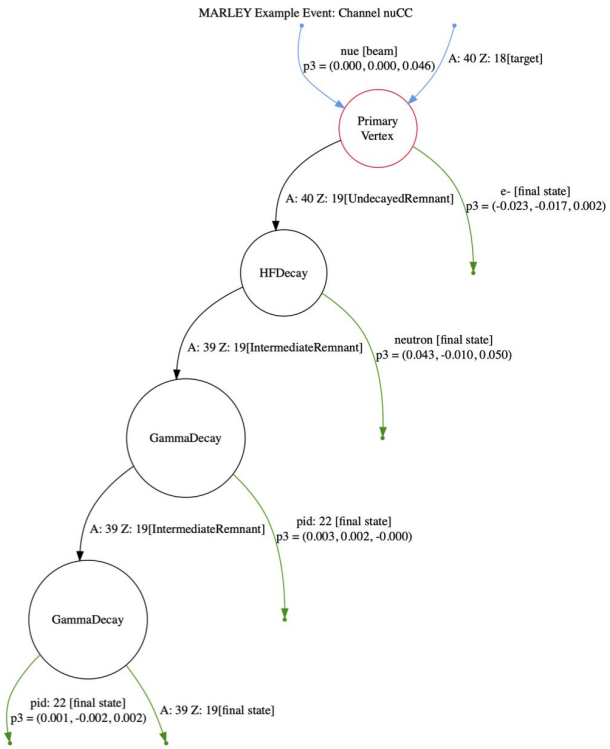
- ASCII/Protobuf formats are message based → Can be streamed
- Specification is probe-agnostic: We hope that it will also be useful for e-, nucleon-, and pion-scattering analyses
- Non-standard PDGs can be defined, with space reserved for specifying information required by GEANT4 for propagation and decay.
- Beam energy distribution information:
  - Simple flux histograms
  - Ideas for including the full beam simulation provenance directly in the HepMC3 event.
- Exposure information (NEvents, or POT)
- Lab position and time

Palate Cleanser

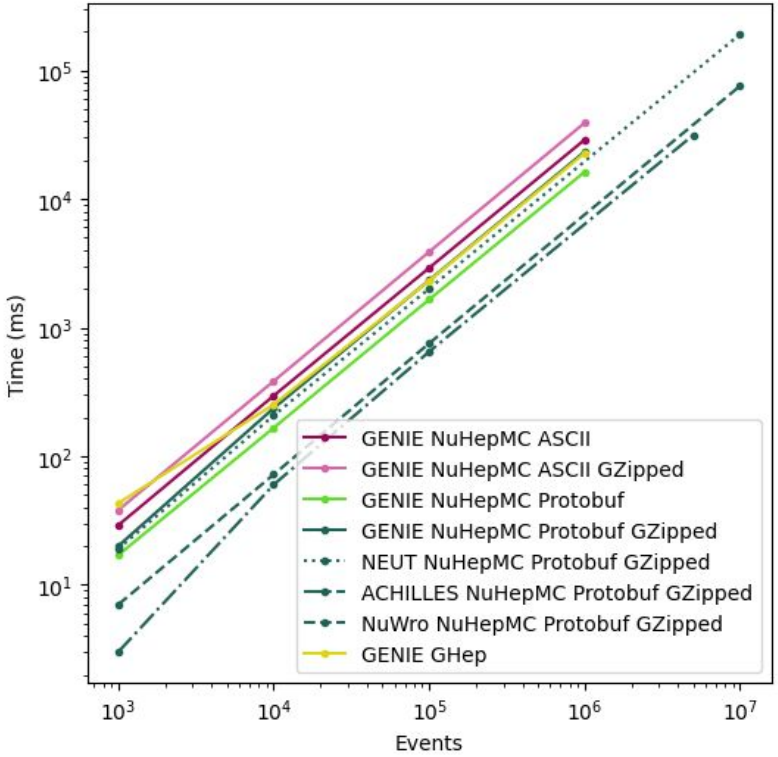
Example Event Graphs

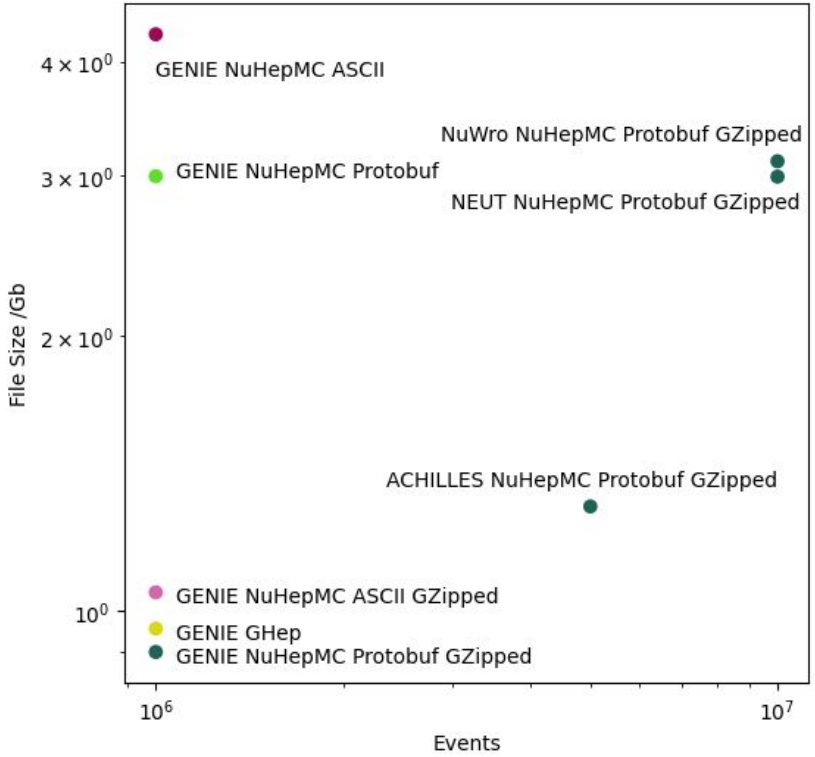






Non-Comprehensive I/O Stats





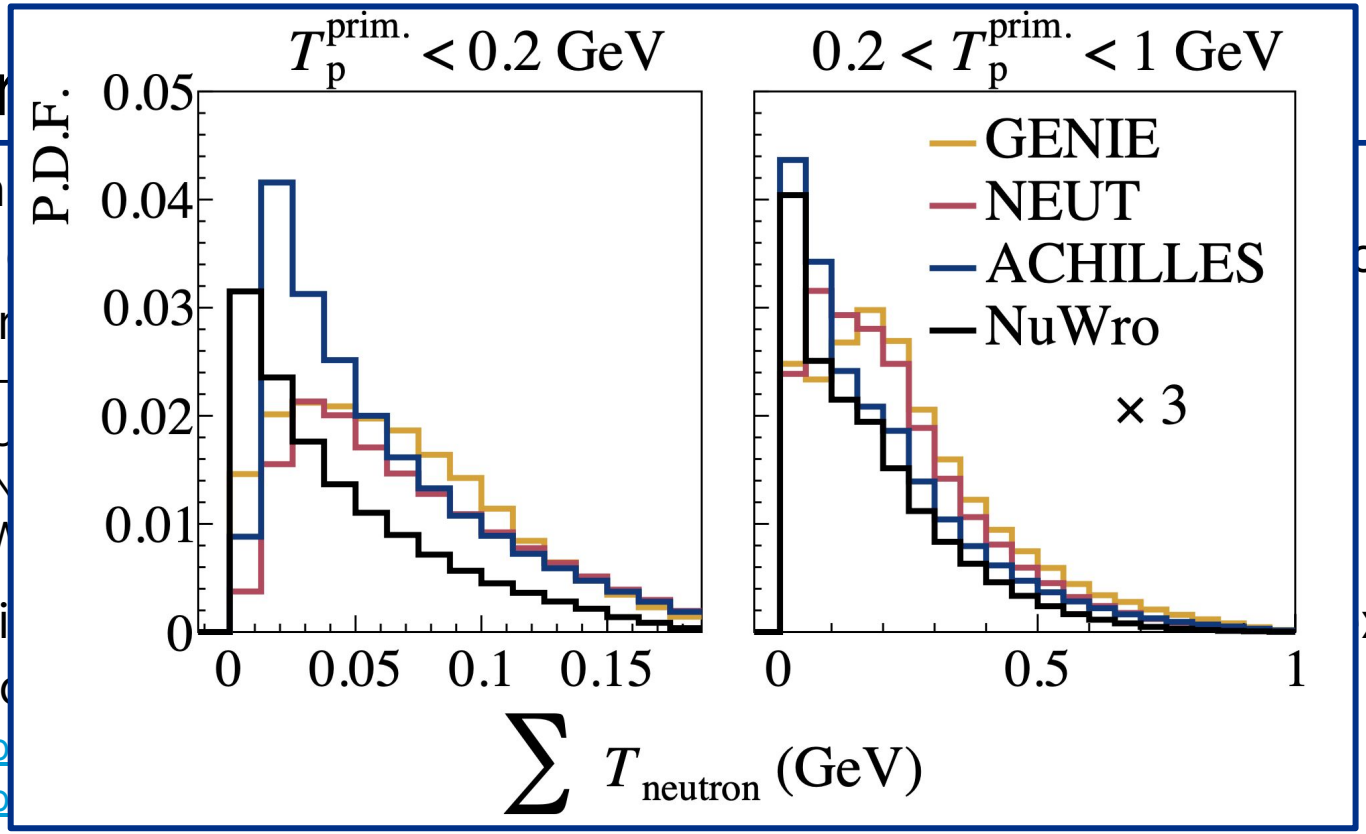
Closing Thoughts

Accomplished

- Written a working specification
- Added compliant native output modules or vector converter tools for a number of in-use generators:
  - ACHILLES: native
  - NEUT: converter https://github.com/neut-devel/neutvect-converter
  - GENIE: native, unofficial https://github.com/sjgardiner/Generator/tree/hepmc3
  - NuWro: converter https://github.com/NuHepMC/nuwro2hepmc3
- Beginning development of community C++ tools and C++/python examples repository:
  - https://github.com/NuHepMC/cpputils
  - https://github.com/NuHepMC/Examples
- Used for first 'end to end' analysis for the NuSTEC FSI white paper
  - Analysis can be run with the only link time dependency being a compiler (+ROOT for plots)

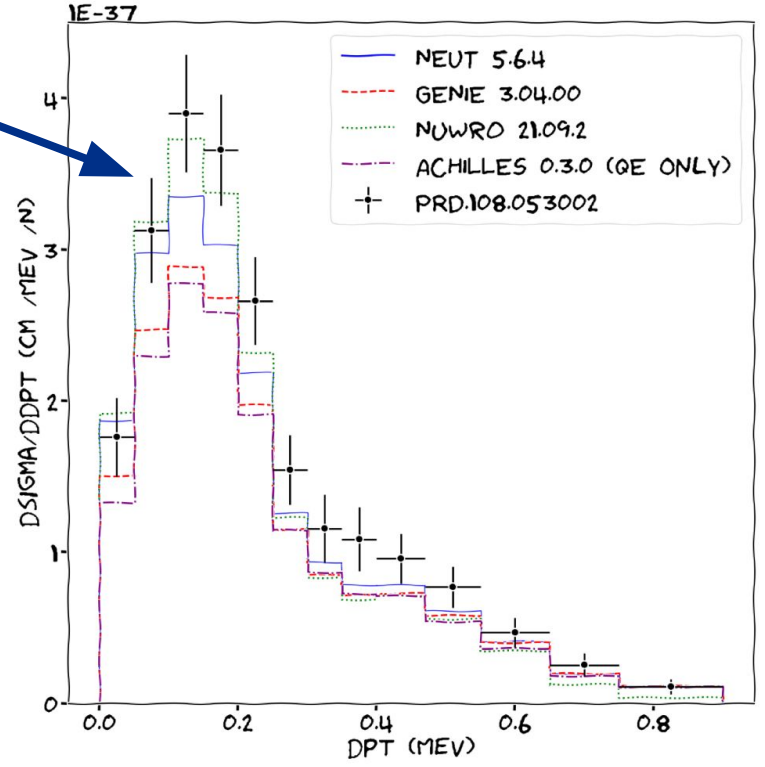
Accor

- Written
- Added
  - ACH
  - NEU
  - GEN
  - NuW
- Beginni
  - http
  - http
- Used for first 'end to end' analysis for the NuSTEC FSI white paper
  - Analysis can be run with the only link time dependency being a compiler (+ROOT for plots)



Other Related Projects

- NUISANCE can already read in NuHepMC events for data comparison



Watch this space for:

- Fully automatic measurement comparison tools based on NuHepMC
  - Leveraging NUISANCE & HEPData
- Standalone 'theorist' Geometry/Neutrino Ray Tracer tools with a standardised API
- Bibliography generation tools

Why!?

Why a Common interface?

- Reduce maintenance → More time for writing community tooling
- Lower technical barrier → More people, more productivity, less bugs
- Improved interoperability → More flexible generator predictions

Why!?

Why a Common interface?

- Reduce maintenance → More time for writing community tooling
- Lower technical barrier → More people, more productivity, less bugs
- Improved interoperability → More flexible generator predictions

Why NuHepMC?

- Based on Collider tools → Spread the maintenance, existing community
- Active core development team
- Flexible on-disk formats, with flexibility for additional ones
- Designed with extensibility and data preservation in mind
- New opportunities for interoperability with Collider tools!

What We Need

- We need people **reading, using and critiquing this specification**:
  - Which use cases doesn't it simplify. Can they be further simplified?
  - Seeing what tools need to be written for non-expert users

What We Need

- We need people **reading, using and critiquing this specification**:
  - Which use cases doesn't it simplify. Can they be further simplified?
  - Seeing what tools need to be written for non-expert users
- We need **political will** to put in the effort to build next-generation experiment stacks on common extensible tools:
  - **It doesn't have to be this one**, but a relatively small amount of work now will make multiple generations of students and PDRAs lives easier.
  - This is the experience communicated from the collider folks

What We Need

- We need people **reading, using and critiquing this specification**:
  - Which use cases doesn't it simplify. Can they be further simplified?
  - Seeing what tools need to be written for non-expert users
- We need **political will** to put in the effort to build next-generation experiment stacks on common extensible tools:
  - **It doesn't have to be this one**, but a relatively small amount of work now will make multiple generations of students and PDRAs lives easier.
  - This is the experience communicated from the collider folks
- We need capable implementers to use the spec their experiment's tools
  - *N.B.* G4 has a HepMC3 interface that should be able to read NuHepMC events 'for free'
  - LArSoft
  - HK – Though some recent work on WCSim to read HepMC3 events
  - Make sure that sufficient NuHepMC 'passthrough' information is available at analysis stage

Thanks for Listening!

```
gevgen --seed 13371337 --tune $GENIE_XSEC_TUNE --cross-sections $GENIE_XSEC_FILE -n 100000 -t 1000180400 -o test.root,ghep,test.hepmc3,hepmc -e 0.1,10 -p 14

 NuHepMC-config --build genie_read.cxx -I$(genie-config --topsrcdir) -lz -llzma -lbz2 $(genie-config --libs) $(root-config --cflags --glibs) -llog4cpp -lxml2 -L${LHAPDF_LIB}
-lLHAPDF -lMathMore -lEGPythia6 -lGeom
```