Summary of Output Record and Factorizable Hadron Rescattering …

Kevin McFarland and Jeremy Wolcott
from two Sackler-free Institutions
10 January 2020

Roadmap

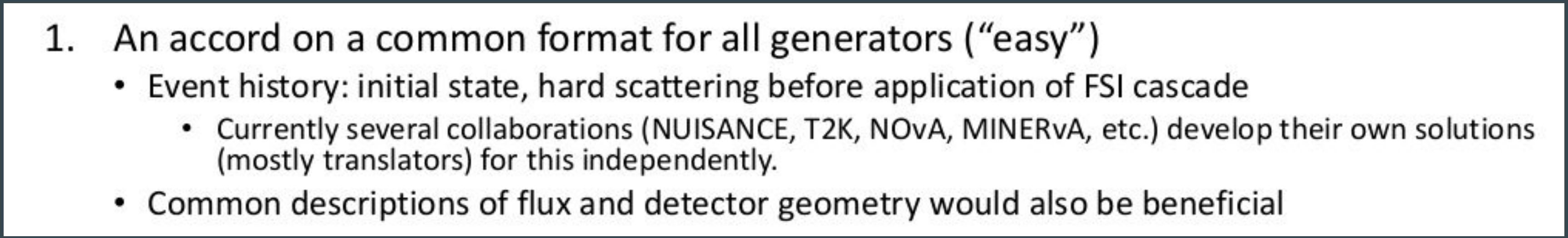
For each of our charges (“common generator format,” “factorization of hadron transport”), we’ll remind you:

- **Needs/context**. What problem(s) are we trying to address? (What’s the situation?)
- **Requirements.** What were the consensus features a solution would need?
- **‘Gotchas’.** What are the pitfalls we need to watch out for?
- **Concept sketches.** Our distillation of what might work, based on discussion
- **Next steps.** What are our action items?

“Common record format”

“Common record”: needs/context

“Workshop goals” intro (KSM)



1. An accord on a common format for all generators (“easy”)
   - Event history: initial state, hard scattering before application of FSI cascade
     - Currently several collaborations (NUISANCE, T2K, NOvA, MINERvA, etc.) develop their own solutions (mostly translators) for this independently.
   - Common descriptions of flux and detector geometry would also be beneficial

Use cases:

- Easier **sharing of experiments’ custom model adjustments/uncertainties** → NOvA-T2K experience: much harder than it needs to be
- **“Lightweight generator” handoffs** to GENIE in “onion model” (no re-inventing the wheel in incompatible ways)
- **Easier to link to external packages, e.g. LArSoft, to multiple generators**
- **Handoffs between generators** in FSI factorization scheme (see later slides)

On nomenclature:

“Format” includes *both* “event storage grammar” (including in-memory representation as C++ data structures) and “persistency” (to/from in-memory representation)

“Common record”: requirements

Non-negotiable event-wise information:

- “Snapshots” of event particles at various stages (initial state, after hard scatter, steps in hadron transport): x^μ, p^μ in lab frame, lineage, status code
- Hard scatter info (but see pitfalls next slide): reaction type, sub-mode (resonance ID?), ...
- Interaction vertex location within nucleus
- Decay, hadronization info (n.b., clear definition of this will need work)
- Storage for neutrino progenitor(s) info (flux): see dk2nu, J-PARC ntuples
- Storage for generator-specific info (i.e., for reweighting)

Essential “metadata”:

- Nuclear potential assumed (global? local, and if so, some details? etc.)
- Generator-specific model configurations

Guidance: **re-use wider HEP community tools where possible** (HepMC?)

Should support e, π, p “probes” in addition to ν by design rather than by accident. (Does that add more requirements?)

“Common record”: ‘Gotchas’

Event and model identifications do not map isomorphically from one generator to another:

- Different primary process partitions
  (e.g.: GENIE “DIS” ≠ NEUT “DIS” ≠ GiBUU “DIS” ≠ NuWro “DIS”)
- Primary processes support different ‘options’ (e.g., how many resonances, which ones?)
- Choices of working frame (lab frame? struck nucleon rest frame? etc.) may differ between generators → choose stored variables carefully; for a subset of critical outputs, will need to agree upon a standard

→ probably **necessitates a community-supported “glossary”** of what every possible entry in the record means as part of the common format (!)

How to record nontrivial interactions between model stages?
→ e.g., hadronization ⊕ FSI ⊕ unstable particle decays (see discussion later)
(what do we do about events that were rejected internally between ‘stages’?)

“Common record”: concept sketches

Possible paths forward:

1. **Translators** (clearly the first step)
   - NUISANCE has already done the input portion of this in order to create its native format. What can we learn/borrow?
   - Output to the “common” format will be a green-field development

2. **Generators write directly** to this format
   - Probably not *hard*, but requires generator buy-in
   - More robust in the long run (esp. if unit tests, etc. included directly)
   - Generators could reasonable write both a native and a common format because storage for this stage is not a driver for computing needs

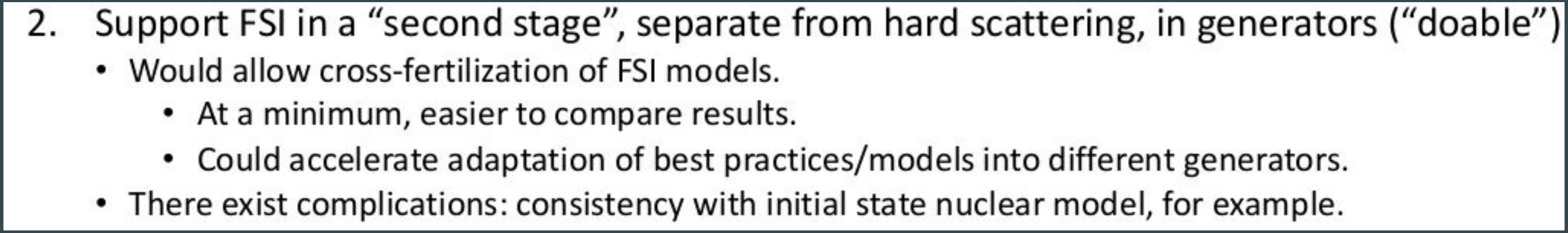
“Common Record”: next steps

1. **Design a proposal**
   - Convene a small group to build it. Likely members:
     - Generator representatives
     - Existing storage format expert(s) (HepMC, other alternatives?)
     - LArSoft representative (“we will switch if a standard format emerges”)
     - NUISANCE representative
   - Invite folks from this workshop (and other interested parties) to comment
   - Iterate as necessary
2. **Develop (at least) one representative translator**
3. **Develop (at least) one representative “consumer”**
   (migrate NUISANCE to use the common format? Or have one generator use this as part of experimenting w/ “factorized FSI”?)
4. **Test**
5. **Report on lessons learned, iterate as needed**
6. Implement full solution

Factorization of hadron transport from hard scattering

Factr8n. had. xport: needs/context

“Workshop goals” intro (KSM)



2. Support FSI in a “second stage”, separate from hard scattering, in generators (“doable”)
   - Would allow cross-fertilization of FSI models.
     - At a minimum, easier to compare results.
     - Could accelerate adaptation of best practices/models into different generators.
   - There exist complications: consistency with initial state nuclear model, for example.

Use cases:

- “Bisection” of **which parts of model correlate with observable behaviors in data** (e.g.: G. Garvey’s request for “no FSI” GiBUU; MINERvA 3D QE-like xsecs)
- Quantify **consequences of violating internal consistency** (see pitfalls, later slide)
- Quick studies of **“lightweight generator” primary xsec models w/ plausible hadron transport**

Factr8n. had. xport: requirements

Generator support:

- Need to be able to configure and run in such a mode, taking a generator record as input
- Need to accept 'intermediate' particle record (particles) as input
  (without requiring they came from internally simulated neutrino interaction)
  → already exists in at least one generator (GiBUU); unsure about others
  → will need public 'recipes' for doing this
- Stripping of first generator's transport stages should happen *before* output

Intermediate record format:

- Essential that info on nuclear model assumptions recorded
  → Can't *enforce* consistency but can make it easy to *verify*
  (c.f.: Python mantra "bad behavior should be discouraged but not banned")

Factr8n. had. xport: pitfalls

- **This approach inherently supports inconsistencies**
  - In almost all applications one could run for this today, nuclear potential will not be consistently treated in this factorization.
  - Nuclear model used in hard scattering can be inconsistent (potential, density) with nuclear model assumed in the hadron reinteraction.
  - Pion production and absorption happen via same diagram: if potentials are not the same in production and hadron transport, unphysical “step” is introduced
    - Violates time reversal symmetry, as Ulrich pointed out. Wrong, but consequences unclear

- **Nontrivial coupling to hadronization and unstable particle decays**
  - Dedicated slide, next

- **Workflow maintenance**

  Even though all pieces live “inside” generators themselves, this is unusual workflow.
  - Who’s responsible for testing, maintaining workflow as generators evolve?
  - Where does documentation of how to run all the steps live?

Coupling of decay and hadronization to reinteraction

Potential problems

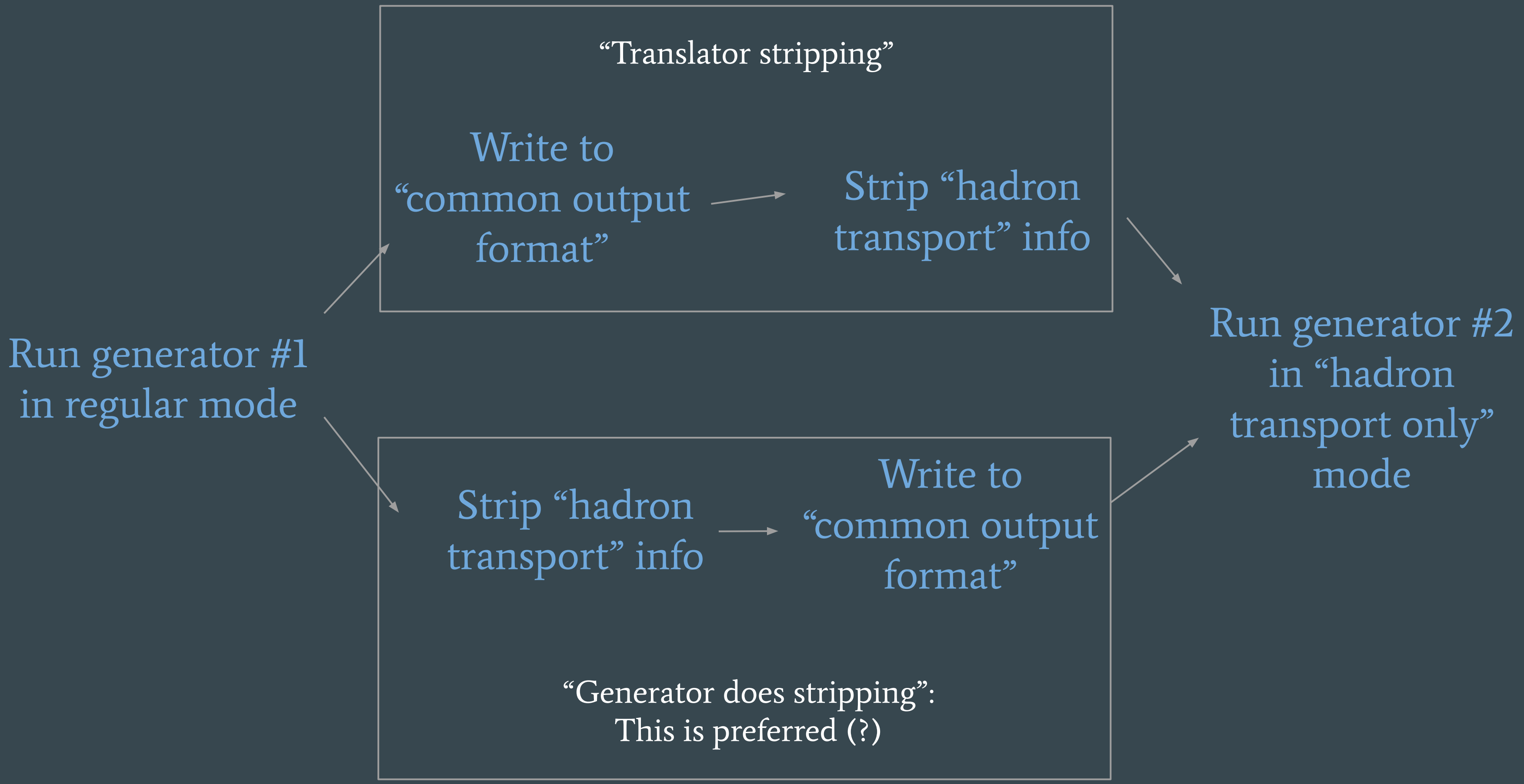
- What happens if a particle is decayed after it undergoes a hadron reinteraction? Would have to be redecayed by the secondary generator.
- For hadronization, would the initial generator provide the formation zone model? If so, the hadrons are not produced at the point of the neutrino interaction.
- What if kinematics after FSI cause the event to “fail” generation? Need to keep the orphaned record because a second stage generator might resurrect it.

Need to work through the use cases and make a concrete proposal.

It could conceivably require some agreements on a standard for these items, or additions to the common record.

Factr8n. had. xport: concept sketches

Workflow options


Run generator #1 in regular mode
"Translator stripping"
Write to "common output format"
Strip "hadron transport" info
Strip "hadron transport" info
Write to "common output format"
"Generator does stripping": This is preferred (?)
Run generator #2 in "hadron transport only" mode

Factr8n. had. xport: Next steps

1. **Design a proposal**
    - Convene a small group to build it. Likely members:
        - Generator representatives
        - Experimental customers
        - “Lightweight generator” candidate representative?
    - Invite folks from this workshop (and other interested parties) to comment
    - Iterate as necessary
2. **Implement “first stage” and “second stage” in representative generators**.
    - “First stage” *could* be done in a translator as a stopgap, despite desire for stripping to be done internally in generator
    - “Second stage”: test driving using current GiBUU machinery is probably enough
3. **Test**
4. **Report on lessons learned**
5. **Iterate as needed**
6. Implement full solution