Hierarchy for storing particle and decay data

For WPEC Subgroup #38

December 10, 2013 at JAEA, Tokai, Japan

Caleb Mattoon

Lawrence Livermore National Laboratory

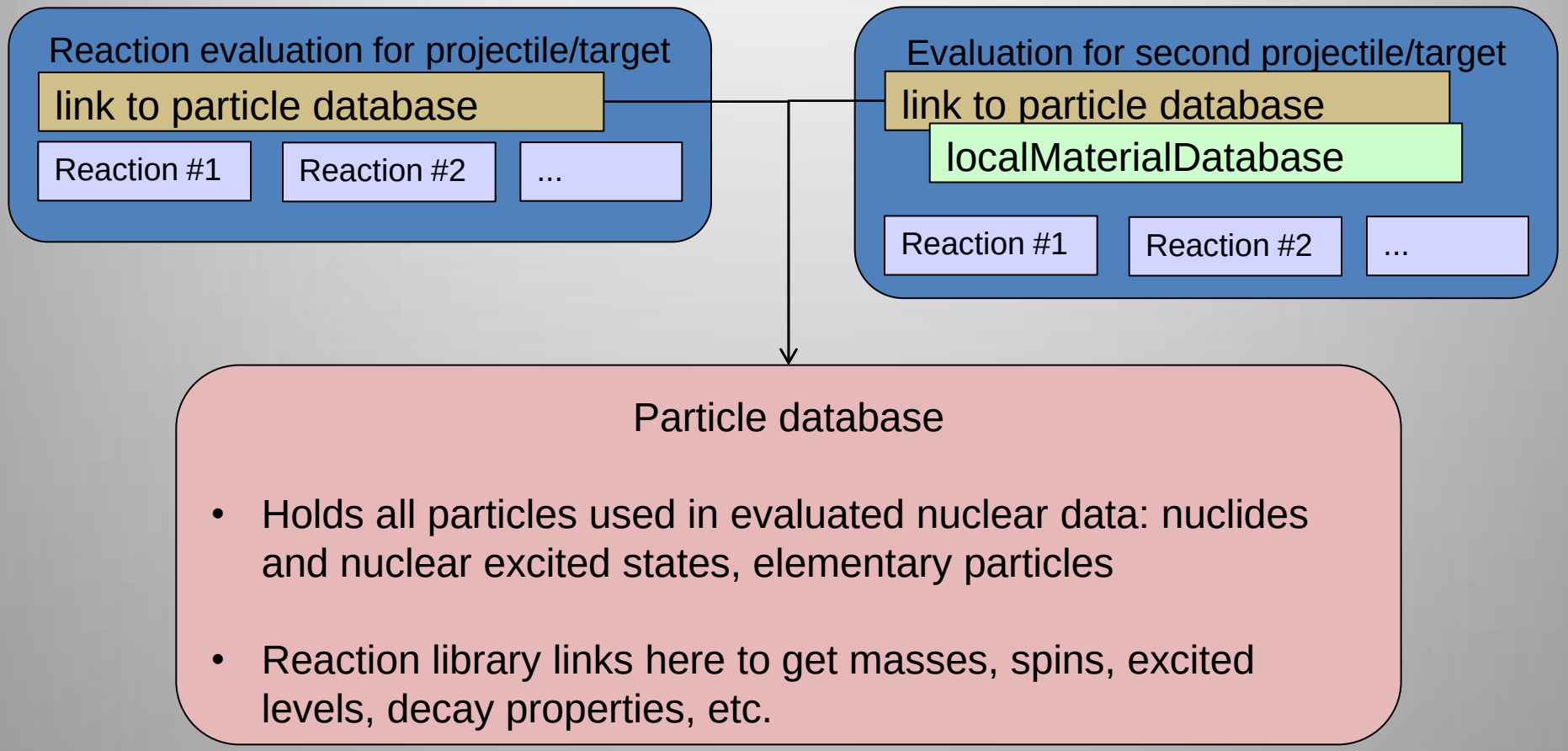
LLNL-PRES-647378

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

SG38 project #3: design a structure to store particle data (masses, levels, decays, etc.)

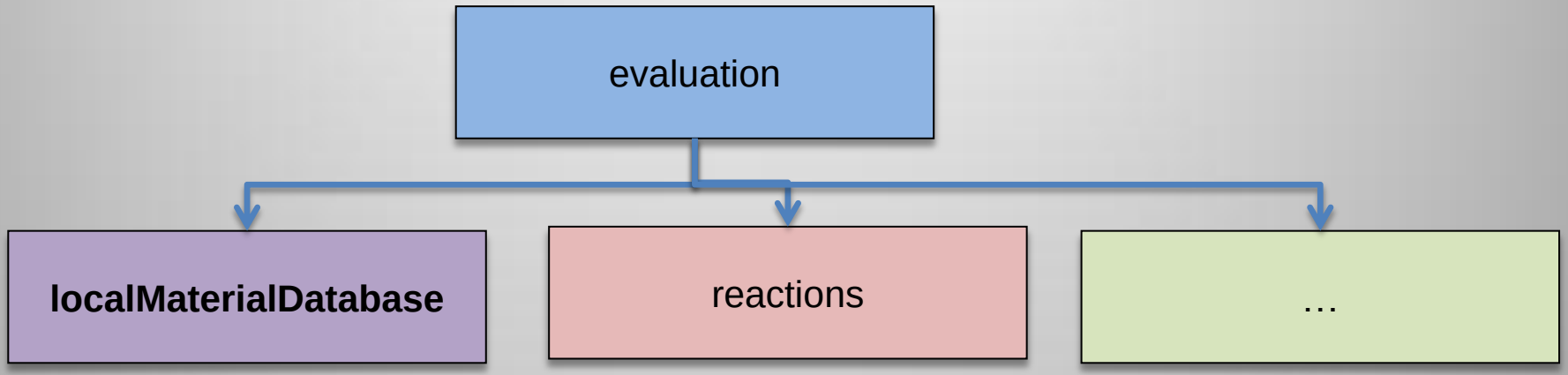
- Goals of this project:
  - Unify how nuclear structure data are stored
  - Reduce redundant information in reaction databases
  - Support (require?) version numbers
  - Support external links to a specific version of the database, and perhaps to a specific particle within a version

Reaction evaluations need an external particle database, *and* ability to override it with internal version



How does this fit into the reaction evaluation hierarchy discussed yesterday?

- From Dave's talk, the top level looks like:



- localMaterialDatabase and external particle database both should use the 'particles' format discussed here

Particle database should also be a unified source for decay data:

- Users who need decay data must use a combination: ENDF MF8 from several sub-libraries, plus RIPL and likely ENSDF.
- Many applications need decay data for:
  - predicting decay spectra
  - using correlated decays for better detection
  - decay heat calculations
  - etc.
- New database should provide easier access

Decision at last SG38 meeting: use RIPL as starting point for new particle database.

- New database will be used for similar purposes as the ‘levels’ and ‘masses’ sections of RIPL. Data types include:
  - Masses, binding energies, deformation parameters
  - Energies and level indices of nuclear excited states
  - Half-lives, spins and parities
  - Gamma decays including branching ratios
  - Uncertainties

- However, RIPL does not include all the data we need...

Goals for today:

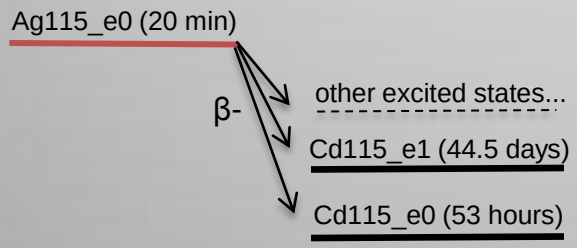
1. Discuss a proposal to extend the RIPL ‘levels’ database with additional data

2. Review proposed new hierarchy for storing particle data
   - Naming convention for particles
   - Status of the RIPL-to-GND translation

What data do we need that are not currently supported by RIPL?

1. Properties of particles other than nuclei.
   - database should include any particle that may appear as either projectile or target in a reaction database
   - photons, electrons, muons, ...

2. More complete alpha and beta decay description.
   - consider Ag115 β-decay:



RIPL does not distinguish between β-decay branches, but they are important for many applications

RIPL levels files are quite simple, only three types of data

- Lines in RIPL levels files may
  - Identify the isotope: ZA, # of levels, separation energies
  - Identify the level: energy, index, spin, parity, # of gammas, halflife, type(s) of decay
  - Identify a gamma: energy, final level, branching ratios

```
 48Mn    48    25    17    18     1     1   15.132001     2.050000
  1   0.000000    4.0  1   1.58E-01  0                      4+   2 = 100.0000 %EC+%B+ = 0.2800
%B+P
  2   0.312000    2.0  1              1                   (2+)   0
                                          1       0.312   0.000E+00   0.000E+00   6.066E-03
  3   0.430000    5.0  1              1                   (5+)   0
                                          1       0.430   0.000E+00   0.000E+00   8.237E-04
```

Should the RIPL format be extended to support storing additional data?

- Makes non-backwards-compatible changes + updates to codes that use RIPL.

- Another option would be to store these data only in the new particle database.
  - Makes translating to and from RIPL more complicated since data may come from multiple sources.
  - Preserve backwards-compatibility in the short term, add new data types later?

Part 2: LLNL's proposed particle data structure:

- Naming scheme for particles (and aliases)
- PoP (Properties of Particles): proposed hierarchy for storing data
- Status of translation

Sticking to a common naming scheme for particles makes data easier to read

- Every particle needs a unique name (or ‘id’). A reaction database can then refer to the particle by id.
  - The id is shorter and easier to understand than an xPath-style link
- These particle IDs should follow a standard convention to avoid confusion

Proposed particle naming scheme, compatible with current GND:

- Identify any atomic nucleus by symbol and 'A':
  - H2, Li7, Fe56, U235...
- Support discrete, continuum and summed excited states:
  - Fe56_e0, Fe56_e1, ..., Fe56_c, Fe56_s, others?
- Other examples of names:
  - n (neutron). Also need e-, mu, ...
  - Allow for aliases (more detail in a moment)

The particle naming convention influences how reactions are labeled in GND:

- Most reactions in GND are identified by the incident and outgoing particles. Examples:
  - n + Mn55 -> n + Mn55_e1 (equivalent to MT51)
  - n + Mn55 -> n + Mn55_c (MT 91)
  - n + Mn55 -> n[multiplicity='2'] + Mn54 (MT 16)

How should we identify natural elements?

- Libraries are moving away from elemental evaluations, but some applications still require them
  - Atomic scattering

- Suggestion: use ‘SYM_natural’ (i.e. Fe_natural, C_natural, etc.)

- Another option: Fe0, C0… but these could be confused with molecules!

Does the naming scheme need to distinguish between atom and bare nucleus?

- Consider the reaction n + N14 -> H1 + C14
  - Is the ‘H1’ product a nucleon or an atom? The difference in mass has an impact on reaction kinematics.
- Our users (NIF) want option to get both bare nuclear and atomic masses of C12
- Proposed naming convention: use lower case to denote nucleus, upper case for atom
- Examples: c12 = nucleus, C12 = c12 + 6 electrons + binding energy

Differentiating Atom vs bare nucleus is fine, but there's a potential rabbit hole here:

- If the naming scheme supports both atoms and bare nuclei, why not partially-stripped nuclei?

- Possible naming scheme: Isotope[chargeState]
  - For example: C12[charge=2+]
  - This is ugly, but that may be fine since we believe this is an infrequent use case
  - Should that be C12 or c12?

Support aliases and links to give users more flexibility

- Problem: ‘metastable’ means different things to different users
  - ‘long’ lifetime to one user can be ‘short’ to another
  - Proposal: use an alias to define metastables, so Am242_m1 points to Am242_e2 using a link
  - Alias could be overridden by a reaction evaluation

- Other possible aliases:
  - h1 (points to p)
  - d (points to h2)
  - a (points to he4)

Too many aliases make the naming scheme useless. Do we need to put limits on aliases?

- Aliases are special exceptions to the rule of the naming scheme. They should be used sparingly.
- One possibility: limit aliases to a short list: metastables + light nuclei
- May never become an issue: perhaps we should just wait to see if aliases actually get abused?

Proposed hierarchy:

- The basic element of the particle database is a ‘particle’. Each particle must contain *at least*:
  - id (unique name identifying the particle)
  - mass
  - charge
  - spin and parity
  - half-life
- Some of these properties may be inherited (discussed shortly), but *every* particle must have a unique id. That way, the id can be entered into hash table for fast lookup

Proposed hierarchy:

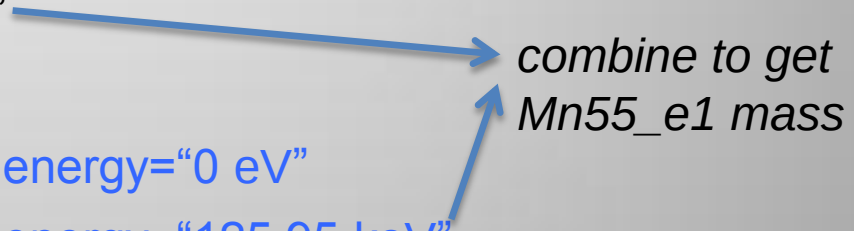
- Particles may include additional information such as:
  - id (unique name identifying the particle)
  - mass + uncertainty
  - charge
  - spin and parity + uncertainty
  - half-life + uncertainty
  - decay information if applicable
  - documentation and/or references
  - … and other relevant information

Prevent redundancy by allowing particles to inherit data. Example: excited nuclear levels

- In an early version of GND, excited states were all treated as independent particles. For example:
  - particle: id="Mn55_e0", mass="54.9380441 amu", ...
  - particle: id="Mn55_e1", mass="54.9381793 amu", ...
  - particle: id="Mn55_e2", mass="54.9391008 amu", ...
  - ...
- Excited state mass should be calculated from excitation energy, not the other way around!
- Instead, nest the excited states so that they inherit ground state mass from the parent isotope

Prevent redundancy by allowing particles to inherit data. Example: excited nuclear levels

- atomicNucleus: id="Mn55", A=55
  - mass
    - atomicMass="54.938045 amu"
  - nuclearLevels
    - level: index=0, id="Mn55_e0", energy="0 eV"
    - level: index=1, id="Mn55_e1", energy="125.95 keV"
    - level: index=2, id="Mn55_e2", energy="984.26 keV"
    - ...



Group all isotopes together under an ‘element’ since they share atomic data

- element: name=“Calcium”, symbol=“Ca”, Z=20
  - atomic information if applicable (excited atomic states, etc.)
  - atomicNucleus: id=“Ca40” ...
  - atomicNucleus: id=“Ca41” ...
  - ...
- Are elements considered particles? If so they also need an ‘id’ attribute.

Finally, organize all particles into a hierarchy:

- particles: version="..."
  + aliases
  - bosons
    + particle: id="photon" *'gamma' handled as an alias*
    - ...
  - leptons
    + particle: id="e-"
    + particle: id="muon"
    - ...
  - nucleons
    + particle: id="n"
    + particle: id="p"
  - composite
    - element: name="Hydrogen", symbol="H", Z=1
      + atomicNucleus: id="H1", A=1
      + atomicNucleus: id="H2", A=2
      + ...
    - element: name="Helium", symbol="He", Z=2
    - ...

Finally, organize all particles into a hierarchy:

- particles: version="..."
  + aliases
  - bosons
    + particle: id="photon" *'gamma' handled as an alias*
    - ...
  - leptons
    - particle: id="e-"
      - mass
      - charge
      - spin, parity
      - half-life
      - documentation, references
    + particle: id="muon"
    - ...
  - nucleons
    + particle: id="n"
    + particle: id="p"
  - composite
    - element: name="Hydrogen", symbol="H", Z=1
    - ...

Finally, organize all particles into a hierarchy:

- particles: version="..."
  + aliases
  - bosons
    + particle: id="photon" *'gamma' handled as an alias*
    - ...
  - leptons
    + particle: id="e-"
    + particle: id="muon"
    - ...
  - nucleons
    + particle: id="n"
    + particle: id="p"
  - composite
    - element: name="Hydrogen", symbol="H", Z=1
      + atomicNucleus: id="H1", A=1
      + atomicNucleus: id="H2", A=2
      + ...
    - element: name="Helium", symbol="He", Z=2
    - ...

Elements and isotopes are nested under the ‘composite’ particles section:

- composite
  - element: name=“Hydrogen”, symbol=“H”, Z=1
    - + atomicNucleus: id=“H1”, A=1
    - + atomicNucleus: id=“H2”, A=2
    - + atomicNucleus: id=“H3”, A=3
  - element: name=“Helium”, symbol=“He”, Z=2
  - ...

Elements and isotopes are nested under the ‘composite’ particles section:

- composite
  - element: name=“Hydrogen”, symbol=“H”, Z=1
    + atomicNucleus: id=“H1”, A=1
    - atomicNucleus: id=“H2”, A=2
      + mass
      + nuclearLevels
      + documentation, references
    + atomicNucleus: id=“H3”, A=3
  - element: name=“Helium”, symbol=“He”, Z=2
  - ...

Elements and isotopes are nested under the ‘composite’ particles section:

- composite
  - element: name=“Hydrogen”, symbol=“H”, Z=1
    + atomicNucleus: id=“H1”, A=1
    - atomicNucleus: id=“H2”, A=2
      - mass
        - atomicMass, separationEnergies, etc.
      + nuclearLevels
      + documentation, references
    + atomicNucleus: id=“H3”, A=3
  - element: name=“Helium”, symbol=“He”, Z=2
  - ...

Elements and isotopes are nested under the ‘composite’ particles section:

- composite
  - element: name=“Hydrogen”, symbol=“H”, Z=1
    + atomicNucleus: id=“H1”, A=1
    - atomicNucleus: id=“H2”, A=2
      + mass
      - nuclearLevels
        - level: index=0, id=“H2_e0”, …
          - energy
          - spin, parity
          - half-life and decay information
          - …
      + documentation, references
    + atomicNucleus: id=“H3”, A=3

Decay information should be given explicitly (including final state) when possible:

- particle (or nuclearLevel)
  - decays
    - decay: decayType="beta-", probability=0.99
      - product id="Sn121_e0"
      - product id="e-"
      - product id="anti_nu_e"
    - decay: decayType="beta-", probability=0.01
      - product id="Sn121_e1"
      - ...

Identify all products by id

Are there too many ways of specifying decay probabilities?

- RIPL and ENSDF prefer to provide the ‘intensity’ rather than decay probability:
  - Intensity = P(decay) / P(most likely decay)

- They store a total ‘electromagnetic’ decay probability along with internal conversion coefficient:
  - P(electromagnetic) = P(gamma) + P(IC)
  - We suggest treating these as two separate decays

What's the most useful way to specify very small decay branches?

- RIPL may use '0 probability' gamma decays to denote a very unlikely decay branch.
- These small branches are useful to record, but they could be a nuisance for applications trying to sample the decays.
- Suggestion:

  - decays
    - …
    - uncommonDecays
      - decay
      - …

Extending the hierarchy to include other types of data should be simple:

- particles
  - ...
  - composite
    - element: name="Hydrogen", symbol="H", Z=1
      - atomicLevels
      - atomicNucleus: id="H1", A=1
      - atomicNucleus: id="H2", A=2
        - atomicLevels
      - ...
    - element name="Helium", symbol="He", Z=2
    - ...

Do we need *atomic* structure data? Could be added to each element, or to each isotope

Also needed: tools for interacting with the new hierarchy

- RIPL and AME masses can now be translated into new format
  - translation back to original formats still needed
- XML schema (to check format) + physics checking codes
- API for reading/writing, sampling
- Visualization tools

Particle database currently includes a subset of the data available in RIPL:

- Contents of RIPL:
  - Optical model parameters
  - Fission barriers
  - Resonance parameters for incident photons (GDR) and neutrons
  - Level schemes, gamma decay info
  - Masses, natural abundances, deformation parameters
  - Level densities

Blue: currently supported by new database

Green: should be added to new database

A few other questions (some of these are only loosely related to a particle database):

- Should molecules be included in particle database (for thermal scattering data)?
  - Possible naming scheme: use ‘ts_’ prefix to designate a thermal scattering material? ‘c_’?
  - ts_H-in-ZrH
  - ts_ortho-H2
  - etc.
- What should we do with parts of RIPL that are not related to particle data?

Good documentation of evaluations includes listing all input parameters!

- Optical model parameters, fission barriers and so on need to be recorded!
- Add a ‘parameters’ section to the documentation of every reaction evaluation?
  - This could just point to RIPL, but it should also allow recording additions or changes to model parameters made by the evaluator.
  - Not just free text: needs to be computer-readable

Top-level hierarchy question:

- If each single evaluation is defined as being single-temperature, how do we handle interpolating between temperatures?

- The metaEvaluation could specify the interpolation rule, but that makes it more complex…

Lawrence Livermore
National Laboratory

The SG38 GForge project is meant to help organize our efforts.

- GForge is a collaboration tool, offering a wiki, discussion forum and document repository (along with other features).
- New SG38 project was created in June
  - Registration required in order to edit, but not to view
- Goal: help keep SG38 momentum going between meetings