



LAWRENCE  
LIVERMORE  
NATIONAL  
LABORATORY

# Alternative Approach to Nuclear Data Representation

J. Pruet, D. Brown, B. Beck, D. P. McNabb

July 27, 2005

nuclear instruments and methods b

## **Disclaimer**

---

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

# Alternative Approach to Nuclear Data Representation

J. Pruet, D. Brown, B. Beck & D. P. McNabb

*Lawrence Livermore National Laboratory, 7000 E. Ave, Livermore, CA*

---

## Abstract

This paper considers an approach for representing nuclear data that is qualitatively different from the approach currently adopted by the nuclear science community. Specifically, we examine a representation in which complicated data is described through collections of distinct and self contained simple data structures. This structure-based representation is compared with the ENDF and ENDL formats, which can be roughly characterized as dictionary-based representations. A pilot data representation for replacing the format currently used at LLNL is presented. Examples are given as is a discussion of promises and shortcomings associated with moving from traditional dictionary-based formats to a structure-rich or class-like representation.

*Key words:* nuclear reactions:general, nuclear engineering and nuclear power studies, computer data analysis

*PACS:* 28., 24., 29.85+c

---

## 1 INTRODUCTION

Simulations of nuclear reactors, coherent archival of nuclear knowledge, and progress in understanding reactions all require a mature nuclear data infrastructure. Collating and quantitatively describing nuclear processes in a coherent way is generally non-trivial. For example, ENDF - the nuclear reaction database and format supporting the reactor community - has taken decades of work by many outstanding scientists to develop and maintain [1].

The data effort is made difficult by a number of factors. One of these simply relates to nuclear physics being a rich and complicated subject, with reacting nuclei exhibiting a great myriad of resonances, breakups, transitions, etc. Another complication relates to the specific needs of different nuclear data consumers. For example, the need for including resonance parameters becomes

clear when one understands details of multi-group transport simulations and self-shielding calculations for nuclear reactor studies (e.g. [2]). The last complicating factor relates to the tools used to represent nuclear data. This is the factor studied here.

In this paper we examine the use of structure-based formats and describe how these formats could benefit the nuclear data infrastructure. The term “structure-based” is used here to mean that the format consists of a collection of separate pieces, each of which represents a different idea or data type. This can be better understood by comparing structure-based formats with more traditional formats currently used for nuclear science.

Current nuclear reaction data storage formats consist roughly of a few simple array-like structures and fixed-formatted character strings. These structures are arranged into files according to well-defined templates. A dictionary (or manual) is used to interpret and recover data. As an example, ENDL (the data format used at LLNL) consist of a sequence of two-line headers followed by an appropriately sized matrix of data [4]. The 11th and 12th columns in the first row of a header contain a number representing the kind of reaction described by the data. Looking in the ENDL manual, we see that a “7” in these columns means that the reaction involves an outgoing photon. As another example, materials in ENDF are specified by a “MAT” number. To understand which material a particular MAT number represents, one refers to the ENDF manual:

“...MAT number for isotopes of an element are assigned on the basis of increasing mass in steps of three, allowing for the ground and two meta-stable states ... the lightest stable isotope is assigned the MAT number ZZ25.... For the special case of elements from Es to Lw MAT numbers 99xx are assigned...”

To determine which isotope a MAT number refers to, one needs the table of isotopes and some simple calculations.

The examples above for ENDL and ENDF underscore the remarkable compactness and efficiency of current nuclear formats. Most of the burden of interpretation rests with the dictionary (or manual) rather than with the data file itself. Representing several isomers of every isotope with a single 4 digit number makes very efficient use of memory. This efficiency was essential in the first days of the data infrastructure effort, when punch cards were used. Also, FORTRAN and other languages of the period made implementation of classes or deeply nested structures tedious. This necessitated a limited variety of data structures.

But the storage efficiency and paucity of structures characterizing current nuclear formats comes at a heavy cost. Since the syntax and basic elements comprising these formats are so limited, extension can be ad-hoc and difficult.

For example, it isn't clear how to add a fourth isomeric state to targets in ENDF. As another example, to represent fission fragment distributions in ENDL one uses  $Z=99$ ,  $A=120$ . With enough extensions a format like ENDL eventually devolves into a list of exceptions. Also, it can be difficult to add support for complicated new data types. Our current effort, for example, was motivated by the need to include covariance data and uncertainty data that seem difficult to represent in ENDL.

An alternative approach to representing nuclear data is one that might be characterized as a structure- or class-based representation. In this approach one first defines a set of simple and general data structures. These will include things like representations of nuclei, levels, vectors, matrix-like objects, and so on. Associations between these structure are used to describe more complicated things like reaction channels. This representation is different from traditional approaches. In ENDL or ENDF, new data types are typically accommodated by adding a new number or word in the data file, along with a corresponding dictionary or manual entry. For example, previously unassigned MAT numbers were designated to represent molecules involved in the coherent scattering of low energy neutrons.

Structure-based representations have a number of advantages over traditional formats. By stacking the building blocks comprising the representation very complicated quantities can be expressed in a straightforward way. As a simple example, one could have a matrix whose elements are themselves arbitrary data structures. The first element might itself be a matrix, the second a simple number, the third an array of arrays, .... Also, the basic data structures (or basis structures) can be built to mirror physics. If one needs to describe reactions involving molecules a structure describing molecules is devised. This might contain an account of the atomic constituents of the molecule, information about vibrational modes and the molecule's mass.

Details of how structure-based nuclear data representations can be designed and implemented are discussed in the rest of this paper. The main goal of this presentation is to give an alternative view of an important longstanding effort and also to add to discussions about the future course of the data infrastructure. To make this discussion concrete we will present examples from the new nuclear reaction data format that is being tested and implemented at Livermore. The ENDL format has served the Lab's needs well for four decades. However, and largely because of recent interest in quantifying uncertainties in large-scale engineering simulations, it was judged that a richer representation was needed.

## 2 Basis structures

In the representations discussed here basis structures comprise the alphabet used to express data. Format designers have the job of carefully choosing and designing these structures. Fortunately, essentially all of the hard work has already been done. We don't have to invent the idea of a matrix, or a particle, or a Legendre-expansion representing an angular distribution. Instead we are left with the more mundane job of transcribing these ideas into a form useful for data representation.

In designing LLNL's new format we attempted to design structures that naturally mirror concepts commonly used in nuclear physics. For example, there are descriptions of particles, nuclei, levels, reaction channels, arrays and functions. Though our specific definitions will undergo revision, it seems likely that any structure-based approach will need some account of these common ideas.

### *2.1 Designing Basis Structures*

It might be useful at the outset to give a rough idea of how one goes about constructing basis structures from scratch. First of all, it seems that any nuclear data format will make use of the familiar mathematical constructs - numbers, arrays, matrices, and so on. Representing a simple number doesn't take much work, though some convention about how they are written is needed. An array is less trivial. One of the simplest ideas is that an array is a list of other data structures. In schematic notation we could write

```
array
  list of other data structures
  length (an integer)
```

Matrices could then be constructed as arrays containing other arrays. More generally, since the `array` defined here can hold any kind of data structure, it can be used whenever one needs to represent a collection of objects. For example, we might choose to represent the excitation spectrum of a nucleus as an ordered array of `level` structures. The `array` structure used in LLNL's new format is similar to that defined above, but has additional support for indexing and for relating array elements to externally defined quantities.

To illustrate how concepts from physics can be represented, consider a structure describing nuclei. A simple first pass at a structure describing a nucleus is

```
nucleus
```

```
proton and atomic numbers (integers)
mass
excitation state
```

Here we would still have to define a structure describing mass and a structure describing the nuclear excitation state. The structure for mass might simply specify a number and its associated units. The structure describing an excitation state needs more thought. In LLNL's new format, for example, the excitation state can be a particular level, or a thermal excitation spectrum, or an arbitrary collection of levels with fractional populations.

Note that the nucleus structure described above only describes a nucleus in a particular state. It would not be appropriate for describing the excitation spectra characterizing the nucleus, the pattern of electromagnetic resonances exhibited by the nucleus, or other characteristics that seem better left to a structure database like that provided in ENSDF [3]. Carrying all of that information in a reaction database seems unnecessary and cumbersome. In fact, it may even be desirable to omit the specification of the nuclear mass, since when  $Z$  and  $A$  are specified it is easy enough to find the mass in a table. But with a minimal specification for the nucleus we would face the problem of resolving references to external data sets. In section 6 we discuss how structure based formats could solve this problem and at the same time help to bridge disparate data efforts.

### 3 Representing reactions

Description of reactions lies at the heart of a nuclear reaction data format. With a well designed account of reactions a huge variety of processes can be represented. To illustrate some of the potential power and elegance of structure-based formats, we present here a description of the `reaction` structure used in LLNL's new format.

It would be quite difficult to design a structure describing reactions in one fell swoop. The task is manageable, though, when we break up the description of a reaction in terms of a collection of smaller logically distinct structures. For simplicity we'll take a top-down view (rather than first describe the most elementary structures). A natural description of reactions that might come from reading [5] is

```
reaction
  incoming channel
  outgoing channel
```

Here the incoming channel specifies the state of the system before interactions occur. As it currently stands the outgoing channel specifies of the asymptotic state of the reacting system as well as any needed information about intermediate states. A possibly attractive alternative would be to specify intermediate evolution on a par with the specification for the initial and final states.

Specifying the incoming channel or the outgoing channel for an exclusive reaction involves describing a collection of bodies (particles, nuclei, etc.), each of which may or may not be in well-specified states. To do this we introduce

```
body account
  body (nucleus or molecule or ...)
  multiplicity
  ambient
```

Here `body` is any structure that might describe a target or projectile. This description of the `body account` has some advantages. First, it reuses the structures already contained in the format. Second, the `body account` (and by extension the `channel` elements) can be as rich as, e.g., the description of nuclei. For example, the `nucleus` structure presented in section 2.1 and used in LLNL's new format can represent nuclei in any excited states. This automatically allows representation of reactions involving excited target or product nuclei. The `ambient` element above specifies the temperature, density or other thermodynamic conditions describing the environment in which the body is immersed. When this element is used it is assumed that the body is in equilibrium with its surroundings.

The `multiplicity` element is used to specify constraints on the number of each type of particle involved in the reaction. A simple structure for this is

```
multiplicity
  number (an integer)
  qualifier (one of greaterThan, lessThan, equal, or any)
```

As an example, to represent a reaction with a single neutron in the incoming channel one would use `number=1`, `qualifier=equal`. The real value of a flexible specification for the multiplicity comes when describing outgoing channels. To represent a reaction in which more than five particles occur in the final state one would use `number=5`, `qualifier=greaterThan`. Or, to represent a reaction which is exclusive on a given particle one would use `qualifier=any`.

Our current specification for the incoming channel is simply

```
incoming channel
  body accounts
```

The specification for outgoing channels includes more information

```
outgoing channel
  body accounts
  intermediate states
  Q value
  reaction data
```

The `Q value` structure is just a number with associated units. Because the `q-value` characterizing a reaction is simply determined once the incoming and outgoing channels are specified it is in some sense redundant. One might make the case that the `Q value` should be omitted here, in much the same way that the nuclear mass might be omitted from the specification of a nucleus (see section 6). Pointwise or functional data characterizing the reaction is described by `reaction data` structure. This structure makes heavy use of the mathematical constructs described by the format. For the present purposes we will emphasize the configuration and evolution of particles in the reaction.

Intermediate states of a reaction are described in roughly the same way as initial and final states of the reaction. Here, though, some extra information about order and timescale are needed. Our current structure is

```
intermediate state
  body accounts
  step number (an integer)
  reaction data
  lifetime (number with associated units)
  decay type (hadronic, direct nuclear, preequilibrium, compound
             nuclear, fission, electromagnetic or weak)
```

The `step number`'s are used to convey that the reaction evolved through an ordered series of intermediate states. The `lifetime` can be used to indicate the length of time over which the intermediate state persists. The `decay type` describes the decay mode of the intermediate state and is one of a handful of basic decay modes (weak, compound nuclear, ...).

Note that a `reaction data` element is included in the definition of the intermediate state. This is done so that detailed information about the intermediate state can be conveyed. As an example, suppose that one wants to describe the spectrum of the first neutron emitted in an  $(n,2n)$  reaction. In this case the intermediate state would consist of a single neutron, a nucleus with the same  $Z$  and  $A$  as that in the incoming channel, and reaction data describing the spectrum of the emitted neutron. The final state described in the `outgoing channel` element would contain two neutrons and the  $(n,2n)$  daughter of the nucleus in the incoming channel.

Overall, then, a reaction in the new format is viewed as consisting of an ordered series of snapshots describing the state of bodies in the system. The first snapshot describes the incoming channel. Subsequent snapshots describe intermediate states of the system that persist over timescales given by a lifetime or decay type (or both). The final state of the system and reaction data characterizing this state are described in the outgoing channel.

In Appendix I we present an example of the reaction structure used to describe delayed neutron emission following fission of  $^{238}\text{U}$ . Though it is hard to parse by eye, a few things are evident. First, distinct structures are assembled to describe this reaction. The meaning of the structures is more-or-less evident. Also, it is clear how to change the content of the structures used to describe this particular reaction to instead account for a rich variety of other reactions. For example, to describe an excited state target we would change the specification of the level in the incoming channel. Or, to describe prompt fission neutron emission we would omit the intermediate state.

An example in Appendix II shows how, with some modifications, structures from LLNL's new format could be combined to represent relativistic heavy ion collisions. Heavy ion collisions are different from the nuclear processes important in reactors in many ways. However the basic elements - channels, reaction data, etc. - used to describe both types of processes are quite similar. It is still the case, though, that designing a faithful account of heavy ion collisions would take a directed effort.

## 4 Writing and expressing data

There are two parts to defining a new format. One concerns the basic data structures used to represent the data. The other part of defining a new format is defining the overall syntax of the format. It is not very difficult to invent a system for expressing the kinds of structures we present. For example, one could simply write code in C describing the contents of different structures. ASCII text of the C code would then serve as stored data. However, more mature solutions exist. Among these is XML (extensible markup language), a convention for describing data stored in tree-like structures. One of the first published studies examining the use of modern markup languages for nuclear data representation was presented in [7].

We are currently using XML because it is widely supported, with several options for open source parsers, checkers, etc. In the short run, this means that we don't have to write our own parsing codes. In the long run, the open source nature of the XML tools ensures that we have control of the source code of all the XML tools that we need. It is also not so hard to write our

own parsers. For these reasons concerns about the longevity of XML are not well founded.

Briefly, XML documents consist of a sequence of nested elements. The beginning of an element described represented by a tag `x` is denoted by “`<x>`”. The end of the same element is denoted by “`</x>`”. Simple elements that contain no nested elements can also be denoted “`<x/>`.” Elements can contain other elements. For example, one way to represent a particle is

```
<particle>
  <particleName> neutron </particleName>
</particle>
```

Elements can also contain attributes, which describe simple characteristics of the element. Another way to express the particle described above using attributes is

```
<particle particleName="neutron">
</particle>
```

Or equivalently (and more compactly): `<particle particleName="neutron"/>`.

Simple data types (strings, integers, etc.), such as `particleName` above, can receive their own elements or be described as attributes. For definiteness, we adopted the convention that all simple data types are described as attributes. Complicated data types always need their own elements.

This is about all one needs to know about XML as far as specification of this new format is concerned. A thorough introduction to XML is left to one of several good introductory books [8]. It should be said that a huge focus of these books relates to web presentation. This is not so important for our purposes. Most books also contain some description of common XML programming tools. Among these are programs that convert one XML file into another, that search files for particular nodes, and so on. Again, these are not so important for our purposes and tend to be initially confusing.

## 5 Parsing and Representing Data in Computer Codes

Data written in a structure-based format is parsed and represented quite differently from data written in dictionary-interpreted formats. To illustrate this, let’s consider the data problem from a different perspective. Suppose that a programmer with no idea about how a format is designed wants to write computer code that will represent reactions. Though there are other choices, we’ll assume that a class “reaction” is constructed to do this. The first job of our

programmer is to decide what kinds of information about reactions she wants to include. In pseudo-code she might decide on

```
class reaction:
    reaction name
    reaction data
    incident particle
    reaction timescale
```

To populate this class, e.g. from an ENDL file, the programmer would read the ENDL manual and translate entries in ENDL files to members of the particle class. This translation is difficult and not one to one. For example, our programmer might have to translate ENDL's "yi=01" into "neutron". Worse still, the reaction class above neglects to define the reaction target, and ENDL's definition of a reaction carries no notion of the timescale over which the reaction is occurring. Part of the trouble here is that the programmer had in mind a much different idea about reactions than the format designers. This mismatch arises because ENDL does not have a mechanism for relating the detailed intent of its designers. In other words, there is no explicit account of the base concepts used.

Structure-based formats solve problems associated with writing programs to represent data in a simple way: the format itself defines the containers or classes used to hold data. For example, by looking in section 3, one sees that designers of our current format had in mind a reaction class defined as an incoming channel and an outgoing channel. In fact, there is such a close mapping between structure-based formats and their class counterparts that class representation of data can be generated automatically from the format specification. This was tremendously useful during development of the basis structures. Our format went through dozens of major substantial revisions, but regenerating processing codes for the format only took seconds for each revision. By contrast, generating a parsing code for a context sensitive dictionary-interpreted format is time consuming and requires human attention to changes in dictionary entries.

Translating data between structure-based and traditional formats requires making explicit the concepts used in the dictionary-based format. Because nuclear physics has a well-developed conceptual foundation this is not so hard as it might first seem. In fact, the process of understanding the original intent of ENDL's designers taught us a lot about the kinds of structures needed in a robust representation. Once an initial set of basis structures had been decided on, faithful translation routines only took a few days to write. A similar accomplishment for ENDF, which expresses a much greater variety of processes, would take more effort and could require a sizeable collaboration between different data groups. The effort to translate data back into traditional formats

is essential if data written in the new format is to be useful. It would take several years at least for processing and application codes to be re-written.

Though there is nothing very sophisticated or tricky about processing data written in structure-based formats, it should be said that some languages are relatively unsuited for the task. FORTRAN 77, and to a lesser extent FORTRAN 90, are among these. If one must parse data with FORTRAN 77, a dictionary interpreted format and string parsing are likely the best choices.

## 6 Bridging disparate data efforts with structure-based formats

Understanding data describing nuclear reactions typically requires the use of ancillary information not properly part of the reaction description. Such information might include the masses and lifetimes of elementary particles, branching ratios for different decays, or nuclear structure information. As an example, consider an (n,n') reaction that leaves the daughter nucleus in a definite excited state. To understand kinematics of this reaction one needs masses for the target and incident neutron, as well as the excitation energy of the daughter. One solution for this is to include needed information in the same file describing reaction data. ENDL does this. The mass and lifetime of  $^{239}\text{Pu}$ , for example, are stored in at least 150 separate places in LLNL's current database. This kind of redundancy is inefficient and in some cases undesirable.

Keeping separate databases dedicated to different aspects or applications of nuclear physics is a potentially better approach. The trouble with this, though, is that users of one data set have to resolve references to external data sets and also have to know the meaning of entries in each data set. Structure-based formats offer a powerful solution to this problem by providing a common central repository of basis structures.

To illustrate this, consider the description of a simple particle like the neutron. For most applications it suffices to relate only the particle's name. Once this is specified the magnetic moment, lifetime, and other characteristics are implied. In this sense the particle name serves as a minimal representation, or identifier, of a larger object. Accessing and interpreting information held in the larger object becomes trivial if different data efforts use the same set of basis structures. For example, ENSDF archivists might be authorities on properties of the neutron and so would use a very thorough particle description:

```
particle
  name
  mass
  lifetime
```

```
spin
spin projection
parity
constituent quarks, ....
```

As reaction data archivists we may need all of this information, so we adopt the same rich particle structure. However, we don't expect to be the definitive source for the neutron lifetime, and a description of the quarks comprising the neutron is generally not needed for our applications. For this reason we would carry only the particle name in our data sets, and defer to [3] or the [6] for needed ancillary information. By using a subset of an agreed upon structure we solve the problem of accessing ancillary information. If more detailed particle information is needed to describe a reaction, we are free to include more elements from the complete structure. For example, to describe emission of polarized neutrons the `spin projection` element would be used. A less trivial example relates to the (n,n') example mentioned above. With a good nuclear structure database we could omit explicit specification of the excitation energy and use an integer excitation number as a minimal representation of a given excited state.

A common pool of basis structures could find broader uses than those relating to communication between different data efforts. One obvious possibility relates to the use of nuclear data by those interested in disparate applications. This is already done with ENDF, which finds use by the reactor community, homeland security efforts, and many of the national laboratories' programs. A more audacious and relatively untested suggestion is that developers of processed data and designers of transport codes could refer to the same set of basis structures used by reaction data archivists.

## 7 Conclusion

The computational nuclear physics group at LLNL is in the process of designing and implementing a new structure-based data format. An initial format has been successfully implemented and is now being extensively tested. In addition to representing all of the data described by ENDL, this format also has the capability to represent uncertainties, covariances, and some complicated reactions. Structure-based approaches to data representation are characterized by several useful features. Among these are easy extensibility, nearly automatic representation by classes within programs, and the ability to represent complicated data in a natural way. Data formats currently used by the nuclear community can be viewed as consisting of key-dictionary pairs and are relatively weak by these standards. However, dictionary-based formats have the great advantage of compactly representing complicated data. As well, tradi-

tional formats have a long, proven record of utility and are widely supported in codes. Whether or not these historical advantages will outweigh advantages of structure-based representations remains to be seen.

It should be emphasized that data formats are only a small part of the larger machinery needed by the nuclear science community. For large-scale simulation campaigns and archival applications a mature data library is generally needed. This library contains a well-specified collection of different data sets that satisfy certain constraints. Typical examples of such constraints might include the requirement that all reactions contain an account of outgoing particle spectra, or the sensible demand that there not be two different data sets describing exactly the same reaction. In addition to enforcing content specifications, the library also provides mechanism needed for retrieving information. For example, part of the ENDL format requires that files be stored with particular names in a particular directory structure. The directory structure and filenames - which are just keys explaining the type of data stored in the file - serve as a sort of database.

The structure-based format described here does not serve as a library organization standard, nor does it specify allowed content of libraries consisting of different reaction data sets. We have chosen to implement database services entirely independently. This choice was made in part because of the wide availability of efficient well-studied database technologies. As well, emphasizing the logical distinction between a format and a useful collection of data sets used in applications seems a constructive starting point for designing a new data representation from the ground up.

A challenge greater than designing consistent library standards relates to changes in the coding infrastructure supporting engineering applications. Improvements in the way nuclear data is represented are not useful if they break all of the existing application codes. The natural short term solution, which has been adopted at Livermore, is to maintain translators between structure-based and traditional data representations. With this none of the processing or application codes need to be changed. In the longer term it is hoped that improvements in the data infrastructure will facilitate improvements in the code infrastructure. In particular, a good transparent structure-based representation could bring the job of processing nuclear data within the purview of well-studied code development and object-based programming methods. This might be desirable because the nuclear engineering community currently relies on only one or a very few core codes maintained by a small number of people.

## References

- [1] Cross Section Evaluation Working Group, January 2004, *Data Formats and Procedures for the Evaluated Nuclear Data File ENDF-6*, BNL-NCS-44945-01/04-Rev.
- [2] Argonne National Laboratory, July 1963, *Reactor Physics Constants*, United States Atomic Energy Commission.
- [3] Evaluated Nuclear Structure Data Files <http://www.nndc.bnl.gov/ensdf>.
- [4] Howerton, R.J., Dye, R.E. & Perkins, S.T. 1981, "Evaluated Nuclear Data Library", LLNL report, UCRL-50400 Vol. 4, Rev. 1.
- [5] deShalit, A. & Feshbach, H. 1974, *Theoretical Nuclear Physics*, John Wiley & Sons Inc
- [6] S. Eidelman et al. 2004, *The Review of Particle Properties*, Physics Letters B 592, 1
- [7] Dunford, C. 2002, in *Summary of the 52nd Cross Section Evaluation Working Group Meeting*, 195
- [8] Ray, Erik T. 2001, *Learning XML*, O'Reilly Press

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

## A Examples of a reaction described using a structure-based format

For illustration we present here the reaction structure used in LLNL's new format to describe delayed neutrons emitted following  $^{238}\text{U}(n, f)$ .

```
<reaction>
  <incomingChannel>
    <bodyAccount>
      <multiplicity number='1' qualifier='equal' />
      <particle particleName='neutron' />
    </bodyAccount>
    <bodyAccount>
      <multiplicity number='1' qualifier='equal' />
      <nucleus Z='92' nucleusName='Uranium238' A='238'>
        <excitationState>
          <level>
            <excitationEnergy units='MeV' Val='0.0' />
          </level>
        </excitationState>
      </nucleus>
      <ambient>
        <Temperature units='MeV' Val='2.586e-08' />
      </ambient>
    </bodyAccount>
  </incomingChannel>
  <outgoingChannel>
    <qValue units='MeV' Val='180.0' />
    <intermediateState decayType='weak' stepNumber='1' />
    <bodyAccount>
      <multiplicity number='1' qualifier='equal' />
      <mixture mixtureName='fissionFragment' />
    </bodyAccount>
    <bodyAccount>
      <multiplicity number='1' qualifier='any' />
      <particle particleName='any' />
    </bodyAccount>
    <channelData dataRef='/nuclearData/i.xml' quantityName='nubar' />
  </outgoingChannel>
</reaction>
```

This is interpreted as saying that the incoming channel consists of a single  $^{238}\text{U}$  nucleus and a single neutron. The U is in its ground state in equilibrium with a thermal bath at temperature  $kT = 2.586 \cdot 10^{-8}\text{MeV}$ . An intermediate state persists over weak timescale before the final state is reached and before the delayed neutrons are emitted. The outgoing channel is that resulting from a fission reaction. Note the use of the fission fragment `mixture` element in the

outgoing channel. This is really just a shorthand for specifying that fission occurred. A full account of the independent and cumulative yields characterizing the fragment distribution would appear somewhere else, and would be cumbersome to include in the channel specification. The **any** occurring in the particle name is used to indicate that the process described is inclusive on all outgoing particles.

Pointwise data describing dependence of the mean number of neutrons emitted as a function of incident neutron energy is contained a separate file (`/nuclear/i.xml` in this example). This file itself contains a structure-based description of the pointwise data. Included in the description are rules for interpolating on the data, the frame (lab or center of mass) in which different quantities are described and the specific meaning of described quantities.

## B Example of the description of a heavy ion reaction

Here we give an example that outlines how one could use a structure-based format like ours to represent the collision of two gold nuclei at the Relativistic Heavy Ion Collider (RHIC). This is meant to illustrate the potential flexibility of structure-based formats. As it stands though, our format cannot do justice to the rich variety of processes happening when two heavy nuclei collide at relativistic velocities.

```
<reaction>
  <incomingChannel centrality='15%'>
    <bodyAccount>
      <multiplicity number='2' qualifier='equal'/>
      <nucleus Z='79' nucleusName='Au197' N='118'>
        <excitationState>
          <level>
            <excitationEnergy units='MeV' Val='0.0'/>
          </level>
        </excitationState>
      </nucleus>
    </bodyAccount>
  </incomingChannel>
  <outgoingChannel reactionName='inclusive'>
    <channelData dataRef='/rhicData/pip_spec.xml' quantityName='trans_mom_dist'>
      <particle particleName='pi+'>
    </channelData>
    <channelData dataRef='/rhicData/pipi_hbt.xml' quantityName='two_part_corr'>
      <particle particleName='pi-'>
      <particle particleName='pi-'>
    </channelData>
  </outgoingChannel>
</reaction>
```

```

    </channelData>
    <channelData dataRef='/rhicData/p_flow.xml' quantityName='flow'>
      <particle particleName='proton'>
    </channelData>
      .
      .
      .
    </outgoingChannel>
  </reaction>

```

At RHIC energies, one can no longer specify distinct channels. Rather, all “channels” are really *inclusive* in that they encapsulate whole classes of channels. In this example, we specify which class by stating the centrality of the collision: we consider the 15% most central collisions. Because it is impractical to use the `outgoingChannel` element to specify the outgoing particles explicitly, we embed `particle` elements inside each `channelData` element to denote which particles are described by which channel data. The actual data tables are stored in some external file as in the previous example.