UCRL-PRES-207437

New Nuclear Data Format

Towards a richer representation

Jason Pruet, Dennis McNabb & Scott McKinley

This work was performed under the auspices of the US Department of Energy by University of California Lawrence Livermore Laboratory under contract W-7405-ENG-48.

Overview

- Introduction
  - Working with data
- Importance of structures used to represent data
  - Simple example
  - Note on nuclear data
  - Benefits of using rich structures
- Writing the data down
  - XML as a promising choice
- Working example: Pointwise data
- Summary: What does a richer representation have to offer the nuclear data community?

Three parts to working with data

Structures representing data

Writing & storing data

Accessing, processing, computing with data

Three parts to working with data

Structures representing data

Writing & storing data

Accessing, processing, computing with data

Impacts whether problem is soluble:

- **QM Scattering:**
  - **Path integrals (hard)**
  - **Schrodingers equation (easy)**
- **CM Orbits:**
  - **Lagrangian**
  - **Force diagram**

Three parts to working with data

Structures representing data

Writing & storing data

Accessing, processing, computing with data

Binary vs. ascii vs. graphs vs. …

Three parts to working with data

Structures representing data

Writing & storing data

Accessing, processing, computing with data

- **Relational vs. hierarchical vs. collection**
- **Transport vs. presentation**
- **Object-oriented vs. FORTRAN vs. ??**

In general different structures can represent data

Example: Differential cross section data

E, θ, σ

Choice 1)

a number: doesn't work

In general different structures can represent data

Example: Differential cross section data

E, θ, σ

Choice 2)

a matrix: e.g. a FORTRAN array

1	0	1
1	90	2
2	0	1

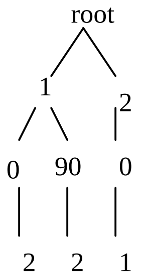
Interpretation rule: first column is E, 2nd is theta, third is sigma.

In general different structures can represent data

Example: Differential cross section data

$$E, \theta, \sigma$$

Choice 3) a simple tree



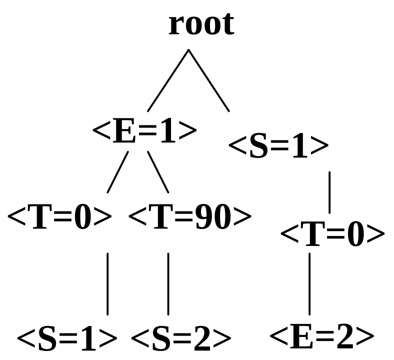
rule: daughter of root is E, daughter of E is theta, daughter of theta is sigma.

In general different structures can represent data

Example: Differential cross section data

E, θ, σ

Choice 4) Tree with named nodes



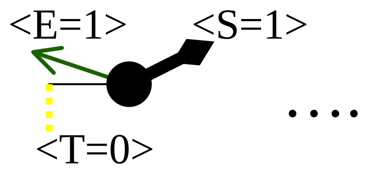
Rule: ‘E’ means E
‘T’ means theta
‘S’ means sigma
each lowest node
corresponds to a triplet

In general different structures can represent data

Example: Differential cross section data

E, θ, σ

Choice 5) More complicated things



rule: ?

Choice of data structure can be critical -- Determines how easy it is to work with data

Example: Extend the simple database described before.

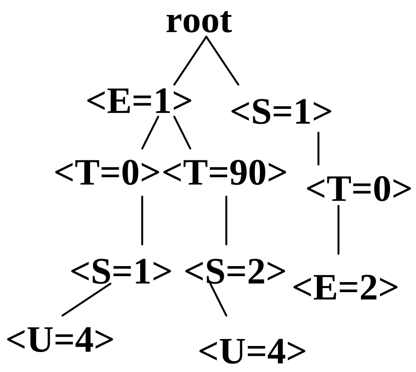
Extension 1) Add uncertainties to the cross section

i) Matrix representation: Add another column

1	0	1	4
1	90	2	4
2	0	1	-1

Rule change: fourth column is uncertainty
-1 means no uncertainty present

ii) Tree with named nodes: Add a tag name



Rule change: 'U' means uncertainty of parent node

Choice of data structure can be critical -- Determines how easy it is to work with data

Example: Extend the simple database described before.

Extension 2) Allow upper and lower uncertainties

i) Matrix: Add two columns

Rule change:

5th column is lower uncertainty,
6th is upper uncertainty

ii) Tree w/ named nodes: Add two tags

Rule change:

'LU' -> lower uncertainty of parent node
'RU' -> upper uncertainty of parent node

Choice of data structure can be critical -- Determines how easy it is to work with data

Example: Extend the simple database described before.

Extension 3) Let all quantities have uncertainties…
and let these uncertainties have uncertainties

i) Matrix: Add 27 columns

Rule change:

… column 10 is the lower uncertainty of the upper uncertainty of the energy …
… column 27 is the upper uncertainty of the lower uncertainty of the cross section

ii) Tree w/ named nodes: No change

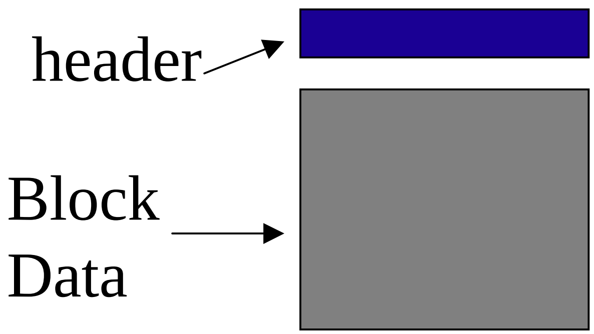
Rule change: none

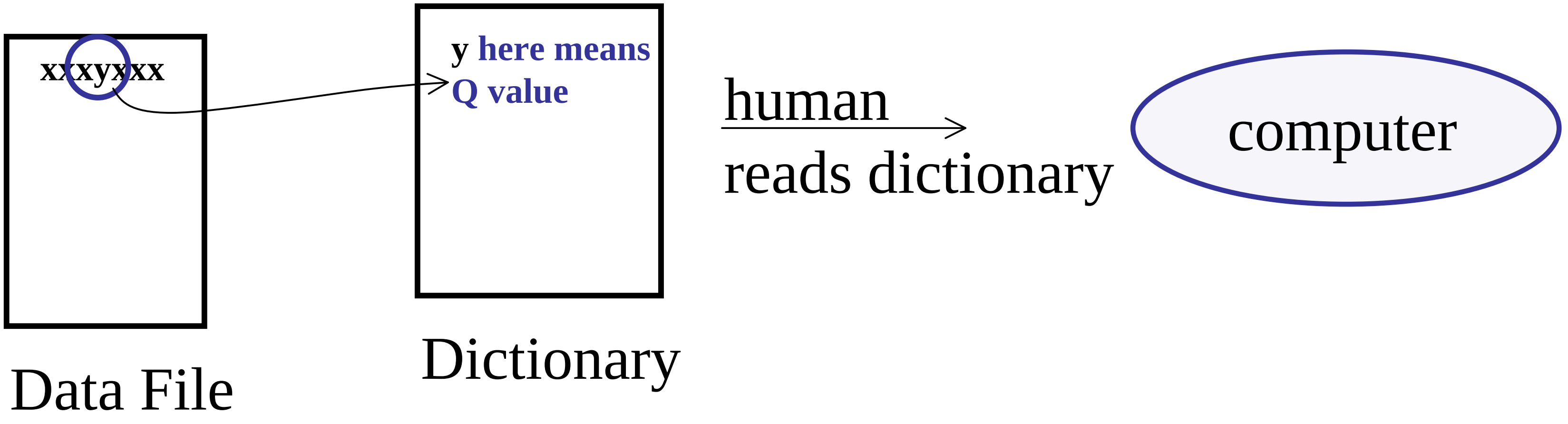
Data representation should allow archivists to:

- **Choose structures needed to best represent data**
- **Determine when data is valid**
  - **Conforms to definition of the structures**
- **Specify relationships between structures**

Current representations are flat -

Context-sensitive, dictionary-interpreted character stream





Advantages and disadvantages of current fixed-format approach

Advantages

- **Very compact**
- **Fast I/O**
- **Useful for non-object oriented programming**
- **Enforces rigid definitions**

Disadvantages

- **Rich structure hard to describe**
- **Significance is encoded by humans reading dictionaries**
- **Burden is on processing codes**
  - **Data doesn’t describe processing**
  - **Expert knowledge of best class structure is lost**
- **Changing data breaks processing codes**

If we move to a richer representation, how do we actually write or store the data?

Goodness criteria:

- **Widely supported**
- **Easily parsed by codes if not humans**
- **Well studied**
  - **Method allows easy representation of complicated structures**

Some possibilities are:

- **Bagpipe recordings**
- **Shelved object instances (python or C++ or …)**

XML: a rich language for describing and defining data structures

Shelved Object Instances -- Advantages and Disadvantages

- Benefits:
  - Already in computer form
  - Extremely rich
- Disadvantages
  - Not readily communicated
  - Ties us to a specific language

This is probably the future --
But lack of a standard is problematic

Disadvantages to an XML-based approach

- Hard to fit on punch cards
- Memory and processor intensive
- Deciding on the structure for the representation takes work

- Lots of work needed for such a change

Advantages to an XML-based approach

- **"Self describing"**

```
<nucleus>
        <Z> 10 </Z>
        <N> 20 </N>
        <mass value="30.2" units="amu"/>
        <level>
                <J value="0" units="unitless"/>
                <pi value="+" units="unitless"/>
        </level>
</nucleus>
```

- **Does a good job of representing a wide variety of complicated structures**
- **Supported and used by thousand of programmers**
  - **All major programming languages and web browsers.**
- **Many useful XML-tree related tools. For example, one line of code will:**
  - **Pick out all q_value nodes**
  - **Re-order all nuclei in the database by decreasing A**
  - **Get all nuclei that have n,2n reaction data available**

Markup languages, particularly XML are supported standards

Conversion from object instances to XML to human-readable forms is straightforward

Close relationship between tree-structures and simple classes

I) Next best thing to having shelved object instances.

II) Humans can't really parse complicated trees.

Example:

A=[x,y,z]
x=['a',1,j]
y=82
z={'Bob':'555-1211'}
j='energy' 'cross-section'
1 5 barns

We can tell what this means and easily write it.

Computers can easily work with this

```
<?xml version="1.0" ?>
<params>
<param>
<value><array><data>
<value><array><data>
<value><string>a</string></value>
<value><int>1</int></value>
<value><array><data>
<value><array><data>
<value><string>energy</string></value>
<value><string>cross-section</string></value>
</data></array></value>
<value><array><data>
<value><int>1</int></value>
<value><array><data>
<value><int>5</int></value>
<value><string>barns</string></value>
</data></array></value>
</data></array></value>
</data></array></value>
</data></array></value>
```

Working example

Motivated by LLNL transport code needs

- Propagate uncertainties in data
- Use data represented as functions
- Fix a few format-imposed inconsistencies

A first draft that includes uncertainties and pointwise data has been made

How structures are represented

i + T $\xrightarrow{\text{reaction}}$ final

Reaction:
- **incident={particle | nucleus}**
- **target={particle | nucleus | mixture}**
- **final=finalConfiguration**
- **ambient**
- **reactionDescription**

Particle:
- **name(e.g. ‘neutron’)**

Nucleus:
- **Z**
- **N**
- **mass**
- **life**
- **excitationState={level | thermal | continuum | preEquilibrium}**

Ambient:
- **temperature**
- **(density, B?)**

reactionDescription:
- **name(e.g. ‘n,2n’)**
- **quantityDescribed(e.g. ‘angularDistribution’)**
- **columnDescription (e.g. c1=‘incidentEnergy’ ,…)**
- **reactionData={pointwise | functional}**

finalConfiguration:
- **residualNucleus**
- **constraints (e.g. E_gamma=4 MeV, …)**

……

Example of pointwise data in XML


```
<?xml version='1.0' encoding='UTF-8'?>
<alldata>
  <nuclear_database>
    <incident particle_name='neutron'>
      <target>
        <description>
          <single_nucleus Z='94' element_name='Plutonium-239' N='145'>
            <mass unitPower='' unit='amu' value='239.052'/>
            <Life unitPower='' unit='sec' value='769000000000.0'/>
          </single_nucleus>
        </description>
        <target_level>
          <description>
            <E_excitation unitPower='' unit='MeV' value='0.0'/>
          </description>
          <reaction>
            <description reaction_name='total'>
              <Q value unitPower='' unit='MeV' value='0.0'/>
```


…


```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type="text/xsl" href="u.xsl"?>
<dataNode>
<c1 value='1.0000000e-11'>
<c2 value='4.8326500e+04'/>
</c1>
<c1 value='1.2384750e-11'>
<c2 value='4.3425500e+04'/>
</c1>
<c1 value='1.5338200e-11'>
<c2 value='3.9021700e+04'/>
</c1>
<c1 value='1.8995980e-11'>
<c2 value='3.5064600e+04'/>
</c1>
<c1 value='2.3526040e-11'>
<c2 value='3.1508800e+04'/>
</c1>
<c1 value='2.9136410e-11'>
<c2 value='2.8313700e+04'/>
</c1>
<c1 value='3.6084720e-11'>
<c2 value='2.5442800e+04'/>
</c1>
<c1 value='4.4690020e-11'>
<c2 value='2.2863200e+04'/>
</c1>
<c1 value='5.5347460e-11'>
<c2 value='2.0545300e+04'/>
</c1>
<c1 value='6.8546450e-11'>
<c2 value='1.8462500e+04'/>
</c1>
<c1 value='8.4893050e-11'>
<c2 value='1.6591100e+04'/>
</c1>
<c1 value='1.0513790e-10'>
<c2 value='1.4909500e+04'/>
</c1>
<c1 value='1.3021070e-10'>
<c2 value='1.3398600e+04'/>
</c1>
```


…

XML as seen by a web browser
(i.e. node names changed to html tag names)

Benefits to the wider community -- An appeal for collaborators on this project

With the switch to an XML based representation the nuclear data effort could concentrate on:

- Concepts needed to represent nuclear data
- Relationships between these concepts

While spending very little time on :

- Formatting data
- Accessing data
- Viewing data
- Modifying existing data
- Transmitting data
- Updating formats

Summary

- **Choice of ideas (structures, classes, containers …) is very important**
  - **Often the most overlooked aspect of data representation.**
- **Excellent tools available for representing rich and complicated data structures**
  - **XML may be the most promising of these**
  - **Limitations that historically required flat dictionary-interpreted files are gone**
- **We have implemented a first version of a new format**
- **Community-wide effort aimed at developing a structure-based approach to nuclear data offers a lot of promise.**
  - **Writing processing codes very easy.**
  - **Data would be more broadly understandable.**
  - **Revisions would be simple**
  - **Representation of new kinds of data less painful.**
  - **Our data effort would be brought under the purview of the work of thousands of excellent programmers. They have already done the work of figuring out how tree-like structures are to be displayed, how to efficiently ‘prune’ trees, how certain kinds of data are best represented, …**