The Universal Accelerator Parser

David Sagan
Dan Bates
Andy Wolski
Overview

• Project motivation
• Accelerator Markup Language (AML)
  - Overview
  - Example
• Universal Accelerator Parser (UAP)
  - Overview
  - Internal Structure
  - Example
  - Language Module
  - Superposition of elements
  - Control elements
• Project status
• Conclusion
Different accelerator analysis programs use different input formats to describe a lattice

A Tower of Babel:

- Fred designs a complicated beamline using MAD
  Format: SIF

- Eric wants to simulate this using PLACET
  Format: SIF doesn’t work
  Need to translate deck to PLACET’s native dialect!

- Algemon wants to do some BDSIM work

- Brian wants cross-check the result using ELEGANT
Solution

Wish List:
- Comprehensive set of machine elements.
- Ability to describe control room knobs, support girders, klystrons, etc.
- Describe complex machine layouts.
- Describe spatially overlapping elements.
- Ability to use arithmetic expressions.
- Flexible - Easily upgradeable to meet changing requirements.
Accelerator Markup Language (AML)

- AML is standardized lattice description format based upon the eXtensible Markup Language (XML).

Why base AML on XML?
- XML is a standard (HTML is based upon XML).
- The flexibility of XML gives AML the ability to be easily extended when desired.
- The flexibility of XML gives AML the ability to be easily used as the basis for a database describing an entire accelerator complex.
XML represents data as a node tree:

```
<root>
  <child1>
    … etc …
  </child1>
  <child2 attrib = "val">
    … etc …
  </child2>
  <child3 />
  … etc …
</root>
```
AML Example

```xml
<laboratory>
  <doc>Hello World!</doc>
  <element name = "q01">
    <length design = "0.6" />
    … etc …
  </element>
  <sector name = "fodo">
    <element ref = "q01" />
    … etc ..
  </sector>
  … etc …
</laboratory>
```
AML in Brief

• Accelerator Markup Language:
  - Full set of machine elements including:
    wigglers, linac accelerating cavities, etc.
  - Can describe physically overlapping elements.
  - Can describe "control room knobs", power supplies, klystrons and other control elements.
  - Can define multiple "machines" linked together.

AML can serve as the starting point for a database for an accelerator complex.
Universal Accelerator Parser

**Problem:** Implementing software that can read in an AML lattice file can be a time consuming process. To do this for each analysis program represents a great duplication of effort.

**Solution:** An open source collaborative effort to implement a software library for reading AML files.

**The Result:** *The Universal Accelerator Parser* project.
UAP Overview

UAP Internal Structure:

- Input Representation Subtree
- AML Representation Subtree
- AML Flat Lattice Subtree

Parsing

Translation

Evaluation & Expansion

AML Input File

Mad Input File
UAP uses a node tree to store information:

Basic node:

Node:
- Name
- Attribute List
- Ptr to Parent
- Ptr to Children

Attribute:
- Name = “Value”
When the UAP software reads a lattice file it creates a tree. The root node has three children:

**<UAPRoot>**
- **<Input_Representation>** - Mirrors the input lattice file.
- **<AML_Representation>** - AML equivalent lattice.
- **<AML_Flat_Lattice>** - Expanded lattice with all expressions evaluated
Example

Mad input file:

! MAD input file
q1: quad, l = 2*A
s2: sextupole
l1: line = (q1, 2*s2)
use, l1
beam, energy = 5.2

<Input_Representation>
  <doc>
    "! MAD input file"
  </doc>
  <element name = "q1"
    key = "quadrupole" l = "2*A" />
  <element name = "s2"
    key = "sextupole" />
  <line name = "l1">
    <element name = "q1" />
    <element name = "s2" repeat = "2" />  
    <use line = "l1" />
  </beam energy = "5.2" />
</Input_Representation>
<AML_Reprsentation>
  <laboratory>
    <doc>"MAD input file"</doc>
    <element name = "q1" />
      <quadrupole />
      <length design = "2*A" />
    </element>
    <element name = "s2" />
      <sextupole />
    </element>
  </laboratory>
  <machine>
    <element name = "l1">
      <element ref = "q1" />
      <sector repeat = "2">
        <element ref = "s2" />
      </sector>
      <root_sector ref = "l1" />
    </element>
    <beam>
      <energy design = "(5.2) * 1e9" />
    </beam>
  </machine>
  …
</AML_Reprsentation>

<AML_Flat_Lattice>
  <machine>
    <tracking_lattice>
      <element name = "q1">
        <quadrupole />
        <length design = "6" />
      </element>
      <element name = "s2">
        <sextupole />
      </element>
    </tracking_lattice>
    <beam>
      <energy design = "5.2e9" />
    </beam>
  </machine>
</AML_Flat_Lattice>
Adding a Language Module

To add a language module one needs to add parsing and translation routines:

UAP Internal Structure:

- Input File Lang A
- Input Representation Subtree
- AML Representation Subtree
- AML Flat Lattice Subtree

- Parsing
- "Easy"
- "Hard"
- Translation
- Evaluation & Expansion
Language Translation

Input File Lang 1 → Parsing → Input Representation Lang 1 → Translation → AML Representation → Translation → Input Representation Lang 2 → Parsing → Input File Lang 2
Superposition

In AML machine elements may spatially overlap other elements:

```xml
<element name = "sol">
    <solenoid />
    <length = "2" />
</element>

<element name = "drft">
    <drift />
    <length = "2" />
</element>

<element name = "q2">
    <quadrupole />
    <length = "1" />
</element>

<sector name = "this_sect">
    <element ref = "sol" />
    <element ref = "drft" />
    <element ref = "q2" superimpose_at = "1.2" ref_element = "sol" />
</sector>
```
Superposition Con't

AML File:

AML_Flat_Lattice Subtree:

Tracking lattice:

Master List:
Controllers

AML provides a mechanism for defining "control room knobs:

```
<controller name = "ps1"
    variation = "ABSOLUTE" >
    <control element = "q1"
        attribute = "multipole:k1"
        coef = "2.3 * sin(ps1)" />
    <control element = "sol"
        attribute = "multipole:ks"
        coef = "-5.7 * ps1" />
</controller>
```

AML Flat Lattice Subtree:

- Tracking lattice:
  - $q_1$
  - $sol$

- Control list:
  - $ps_1$
Project Status

• Accelerator Markup Language:
  - The basic specification exists.
  - Lots of room for development.

• Universal Accelerator Parser:
  - Currently under development.
  - Expect usable software in 3 - 6 months.
  - MAD-8 and MAD-X Language modules will be implemented.
  - Open source (GNU Lesser General Public License).
  - Source is available at SourceForge.com.
  - Written in C++.
  - Java version.
  - Fortran90 interface.
  - Anyone who is interested in invited to participate.
  - Project home page:

  http://www.lns.cornell.edu/~dcs/aml
Conclusion

• The Universal Accelerator Parser software is currently under development for lattice parsing of AML, MAD-8, MAD-X, ...
• Its use holds the promise of greatly improving the interoperability between different programs.
• The UAP library contains bookkeeping routines to simplify the task of simulating the control system, and defining and manipulating complex beamline features such as physically overlapping elements, etc.
Thanks

Yves Roblin
Mike Forster
Theo Larrieu
Tom Pelaia
Frank Schmidt
Peter Tenenbaum
Nick Walker
Mark Woodley
Nikolay Malitsky
Richard Talman