Lucretia2AML

Steve Molloy
3rd April, 2008

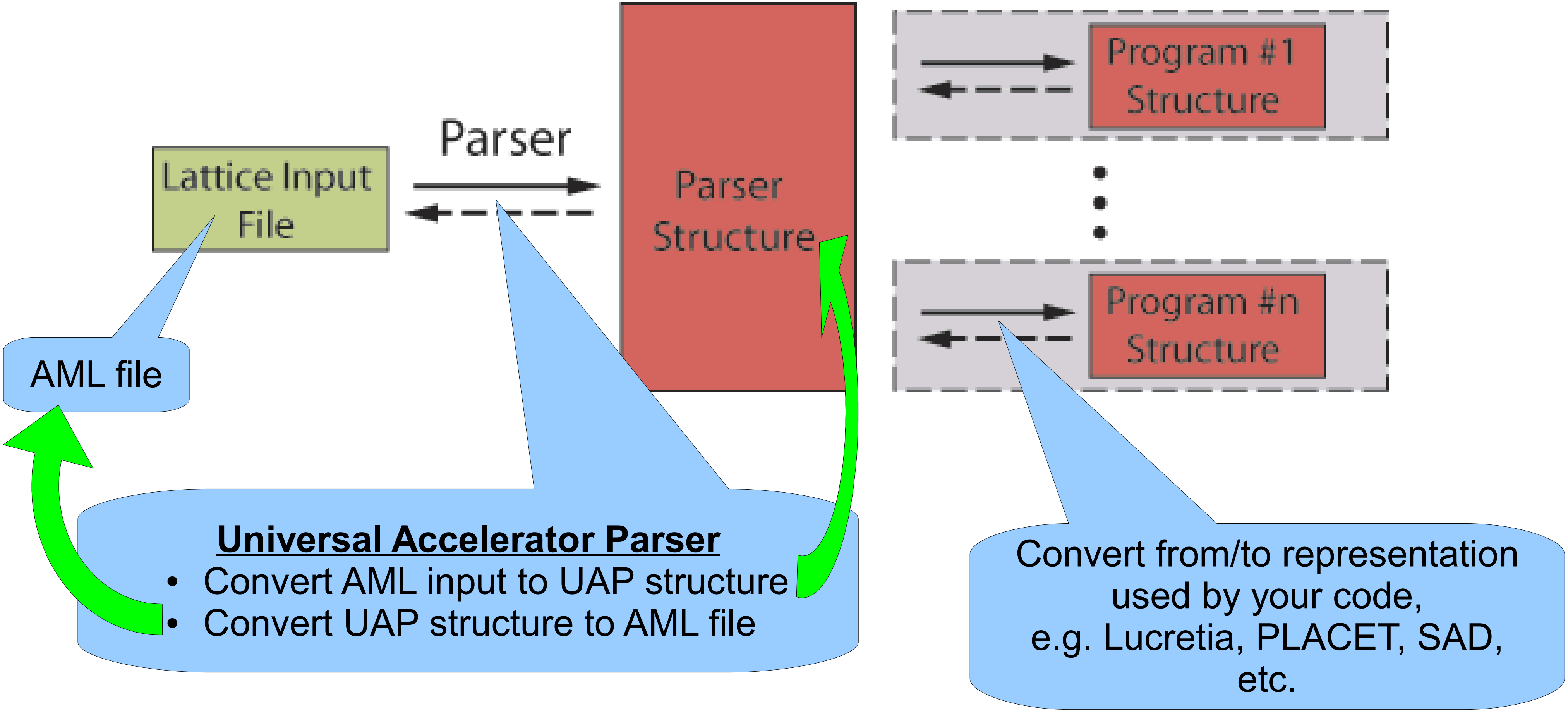
(Updated 19th June, 2008)

AML – A Quick Reminder

- Accelerator Markup Language
  - Based on XML
    - Looks like HTML!
  - Standards designed by W3 Consortium
  - Physics designed by Sagan, et al.
- Designed to allow a generic representation of the physical reality of a beamline
  - No need to split quads
  - Expandable to include engineering data
    - Including the Flight Simulator details

Universal Accelerator Parser (UAP)

- AML is “yet another” lattice representation standard
  - Not much good on its own!!
- The real benefit comes from the UAP



AML file

Universal Accelerator Parser

- Convert AML input to UAP structure
- Convert UAP structure to AML file

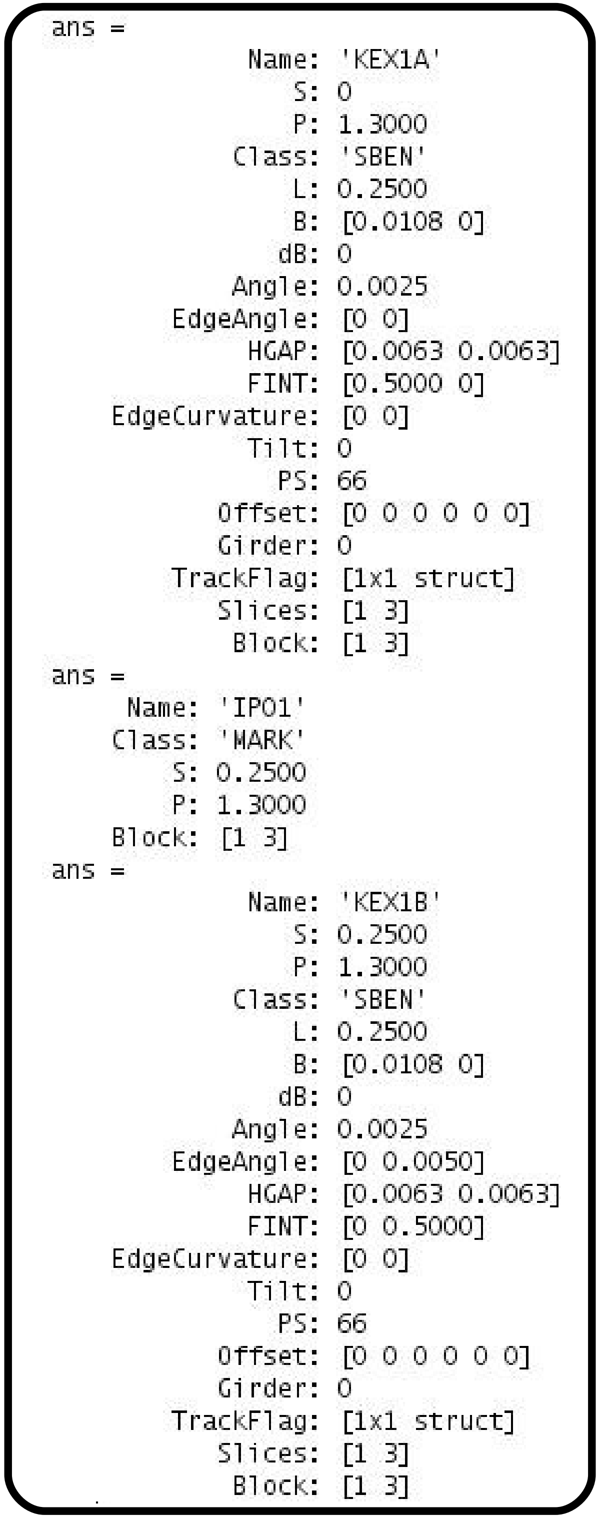
Convert from/to representation used by your code, e.g. Lucretia, PLACET, SAD, etc.

Lucretia2AML – Motivation

- Flight Simulator will allow ATF2 access from Lucretia
- Many potential users use different codes
    - SAD, Placet, MAD, etc.
  - We can't force them to adopt Lucretia
- ILC Deckmasters plan to move from XSIF to AML
  - Thus AML is becoming the new standard
- Lucretia2AML will convert the machine lattice to AML
  - Gives a true representation of the machine in AML

Lucretia and AML lattices: Differences in philosophy

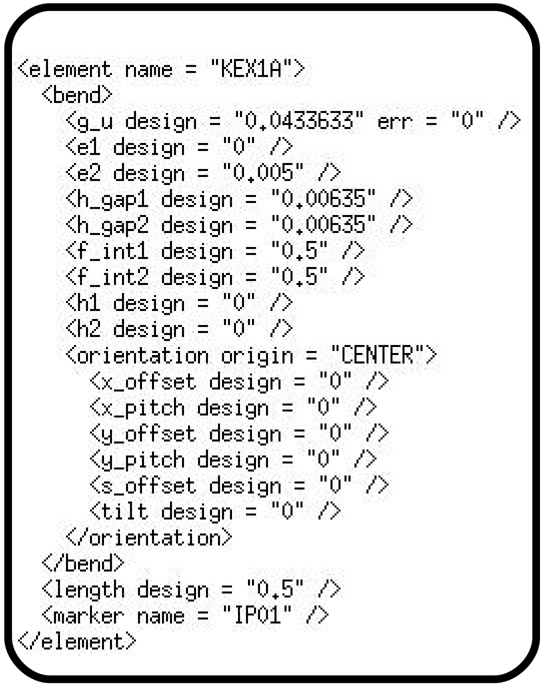
- **Lucretia**
  - Designed for ease of use in a beam tracker
  - Each element represents only one “thing”
    - Drifts, BPMs, markers, magnets, etc.
    - Magnets with internal BPMs must be split
  - Engineering details only present when necessary for tracking
    - e.g. Girders exist for assignment of errors
  - Not extensible



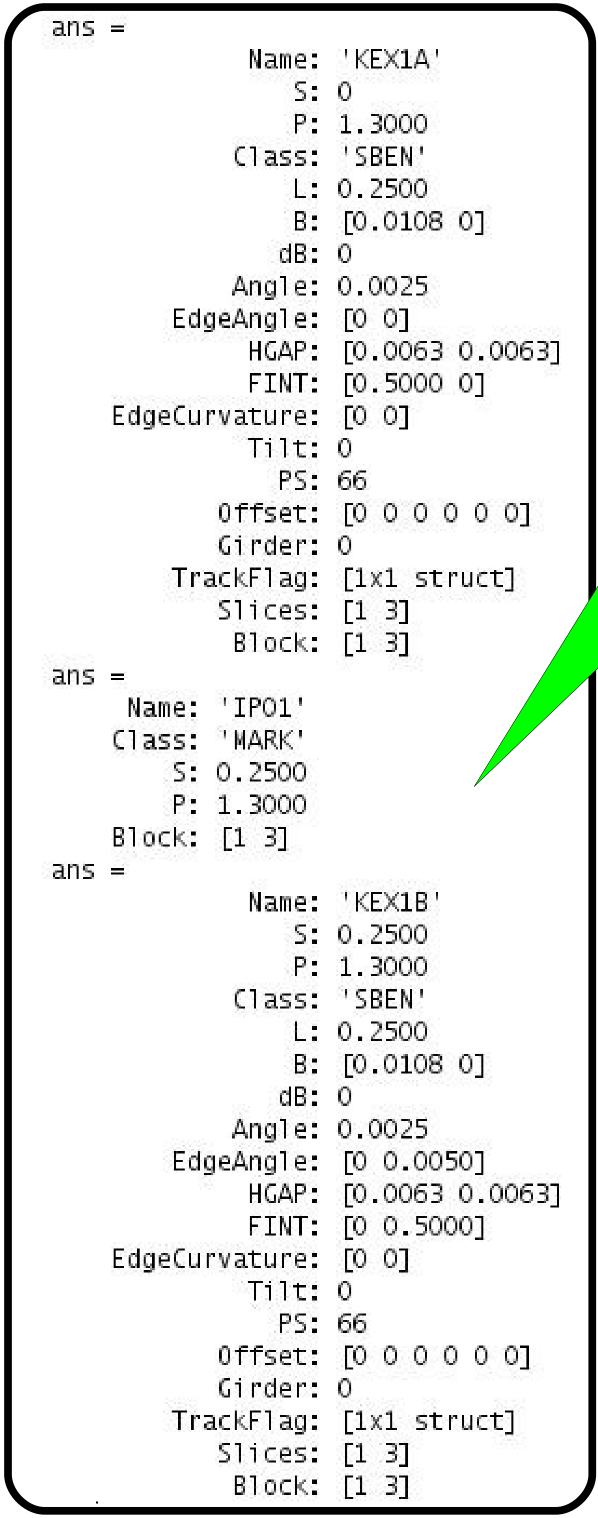
```
ans =
                 Name: 'KEX1A'
                    S: 0
                    P: 1.3000
                Class: 'SBEN'
                    L: 0.2500
                    B: [0.0108 0]
                   dB: 0
                Angle: 0.0025
            EdgeAngle: [0 0]
                 HGAP: [0.0063 0.0063]
                 FINT: [0.5000 0]
        EdgeCurvature: [0 0]
                 Tilt: 0
                   PS: 66
               Offset: [0 0 0 0 0 0]
               Girder: 0
            TrackFlag: [1x1 struct]
               Slices: [1 3]
                Block: [1 3]
ans =
         Name: 'IP01'
        Class: 'MARK'
            S: 0.2500
            P: 1.3000
        Block: [1 3]
ans =
                 Name: 'KEX1B'
                    S: 0.2500
                    P: 1.3000
                Class: 'SBEN'
                    L: 0.2500
                    B: [0.0108 0]
                   dB: 0
                Angle: 0.0025
            EdgeAngle: [0 0.0050]
                 HGAP: [0.0063 0.0063]
                 FINT: [0 0.5000]
        EdgeCurvature: [0 0]
                 Tilt: 0
                   PS: 66
               Offset: [0 0 0 0 0 0]
               Girder: 0
            TrackFlag: [1x1 struct]
               Slices: [1 3]
                Block: [1 3]
```

Lucretia and AML lattices: Differences in philosophy

- **AML**
  - Designed to store the physical state of the machine
    - Tracking information can be extracted when needed
    - Same for engineering information, etc.
    - Each element can represent many “things”.
  - Real magnets aren't split!
    - BPMs/markers/etc can be on their own or part of a magnet.
  - Extensible
    - Lucretia lattice will be a subset of the AML representation.



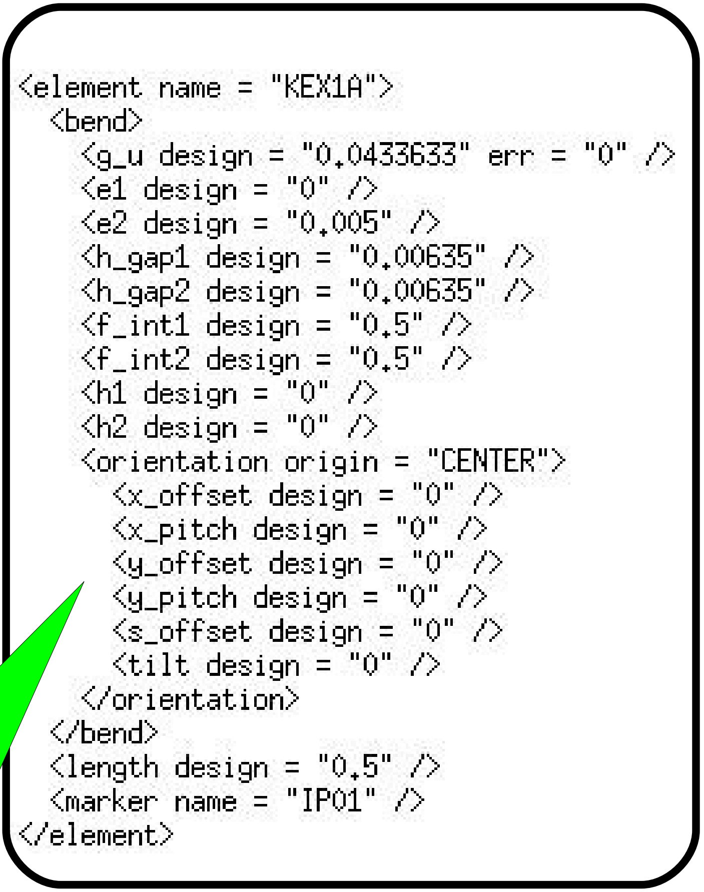
```
<element name = "KEX1A">
  <bend>
    <g_u design = "0.0433633" err = "0" />
    <e1 design = "0" />
    <e2 design = "0.005" />
    <h_gap1 design = "0.00635" />
    <h_gap2 design = "0.00635" />
    <f_int1 design = "0.5" />
    <f_int2 design = "0.5" />
    <h1 design = "0" />
    <h2 design = "0" />
    <orientation origin = "CENTER">
      <x_offset design = "0" />
      <x_pitch design = "0" />
      <y_offset design = "0" />
      <y_pitch design = "0" />
      <s_offset design = "0" />
      <tilt design = "0" />
    </orientation>
  </bend>
  <length design = "0.5" />
  <marker name = "IP01" />
</element>
```



```
ans =
             Name: 'KEX1A'
                S: 0
                P: 1.3000
            Class: 'SBEN'
                L: 0.2500
                B: [0.0108 0]
               dB: 0
            Angle: 0.0025
        EdgeAngle: [0 0]
             HGAP: [0.0063 0.0063]
             FINT: [0.5000 0]
    EdgeCurvature: [0 0]
             Tilt: 0
               PS: 66
           Offset: [0 0 0 0 0 0]
           Girder: 0
        TrackFlag: [1x1 struct]
           Slices: [1 3]
            Block: [1 3]
ans =
     Name: 'IP01'
    Class: 'MARK'
        S: 0.2500
        P: 1.3000
    Block: [1 3]
ans =
             Name: 'KEX1B'
                S: 0.2500
                P: 1.3000
            Class: 'SBEN'
                L: 0.2500
                B: [0.0108 0]
               dB: 0
            Angle: 0.0025
        EdgeAngle: [0 0.0050]
             HGAP: [0.0063 0.0063]
             FINT: [0 0.5000]
    EdgeCurvature: [0 0]
             Tilt: 0
               PS: 66
           Offset: [0 0 0 0 0 0]
           Girder: 0
        TrackFlag: [1x1 struct]
           Slices: [1 3]
            Block: [1 3]
```

Combine 3 Lucretia elements (defined by “Block”) ...

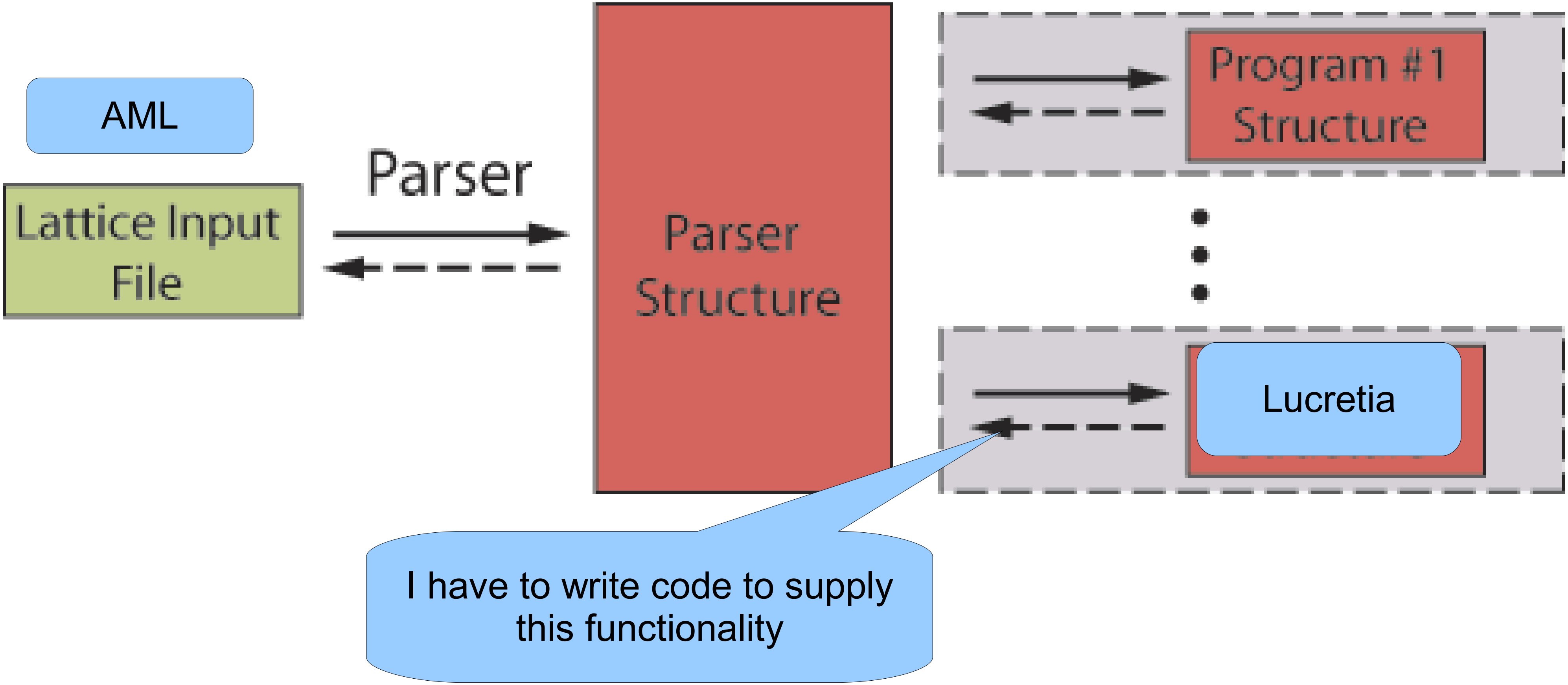
Lucretia2AML



```
<element name = "KEX1A">
  <bend>
    <g_u design = "0.0433633" err = "0" />
    <e1 design = "0" />
    <e2 design = "0.005" />
    <h_gap1 design = "0.00635" />
    <h_gap2 design = "0.00635" />
    <f_int1 design = "0.5" />
    <f_int2 design = "0.5" />
    <h1 design = "0" />
    <h2 design = "0" />
    <orientation origin = "CENTER">
      <x_offset design = "0" />
      <x_pitch design = "0" />
      <y_offset design = "0" />
      <y_pitch design = "0" />
      <s_offset design = "0" />
      <tilt design = "0" />
    </orientation>
  </bend>
  <length design = "0.5" />
  <marker name = "IP01" />
</element>
```

... into 1 AML element with several “attributes”

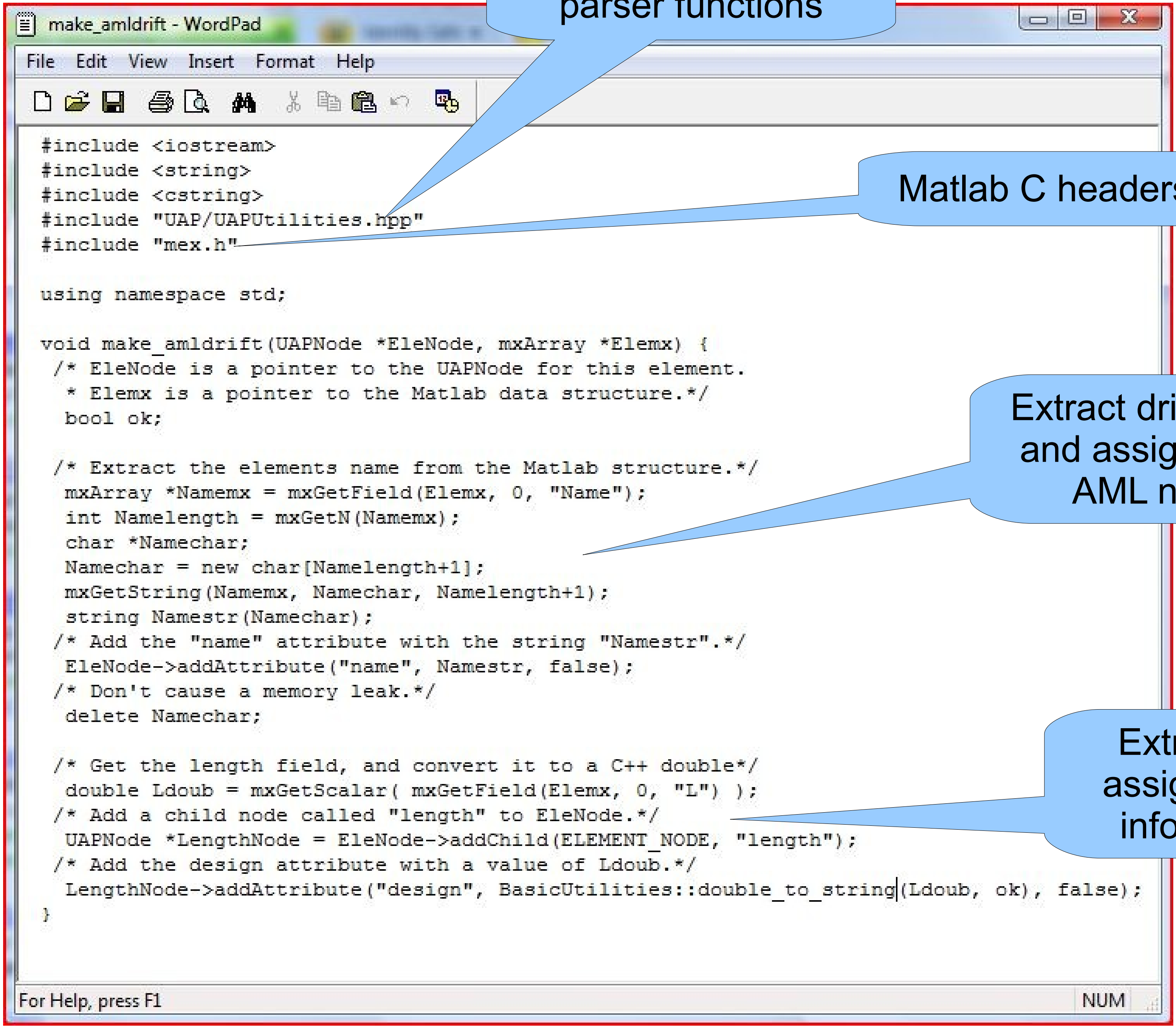
From a previous slide...



I have to write code to supply this functionality

Coding choice

- Lucretia is Matlab based, while UAP relies on C++
  - Matlab allows access to compiled C/C++ via “mex” files
    - Compiled C which includes Matlab-supplied headers
  - Use this to interface with UAP C++ libraries
- Integrate with C at low level, or only when needed?
  - I.e. Majority of Lucretia2AML in C or Matlab?
- Decided to code almost 100% in C/C++
  - Very fast execution speed
  - Still relatively simple source code



```
#include <iostream>
#include <string>
#include <cstring>
#include "UAP/UAPUtilities.hpp"
#include "mex.h"

using namespace std;

void make_amldrift(UAPNode *EleNode, mxArray *Elemx) {
 /* EleNode is a pointer to the UAPNode for this element.
  * Elemx is a pointer to the Matlab data structure.*/
  bool ok;

 /* Extract the elements name from the Matlab structure.*/
  mxArray *Namemx = mxGetField(Elemx, 0, "Name");
  int Namelength = mxGetN(Namemx);
  char *Namechar;
  Namechar = new char[Namelength+1];
  mxGetString(Namemx, Namechar, Namelength+1);
  string Namestr(Namechar);
 /* Add the "name" attribute with the string "Namestr".*/
  EleNode->addAttribute("name", Namestr, false);
 /* Don't cause a memory leak.*/
  delete Namechar;

 /* Get the length field, and convert it to a C++ double*/
  double Ldoub = mxGetScalar( mxGetField(Elemx, 0, "L") );
 /* Add a child node called "length" to EleNode.*/
  UAPNode *LengthNode = EleNode->addChild(ELEMENT_NODE, "length");
 /* Add the design attribute with a value of Ldoub.*/
  LengthNode->addAttribute("design", BasicUtilities::double_to_string(Ldoub, ok), false);
}
```

Utilities for accessing parser functions

Matlab C headers

Extract drift name and assign to an AML node

Extract and assign length information

Status – Completed

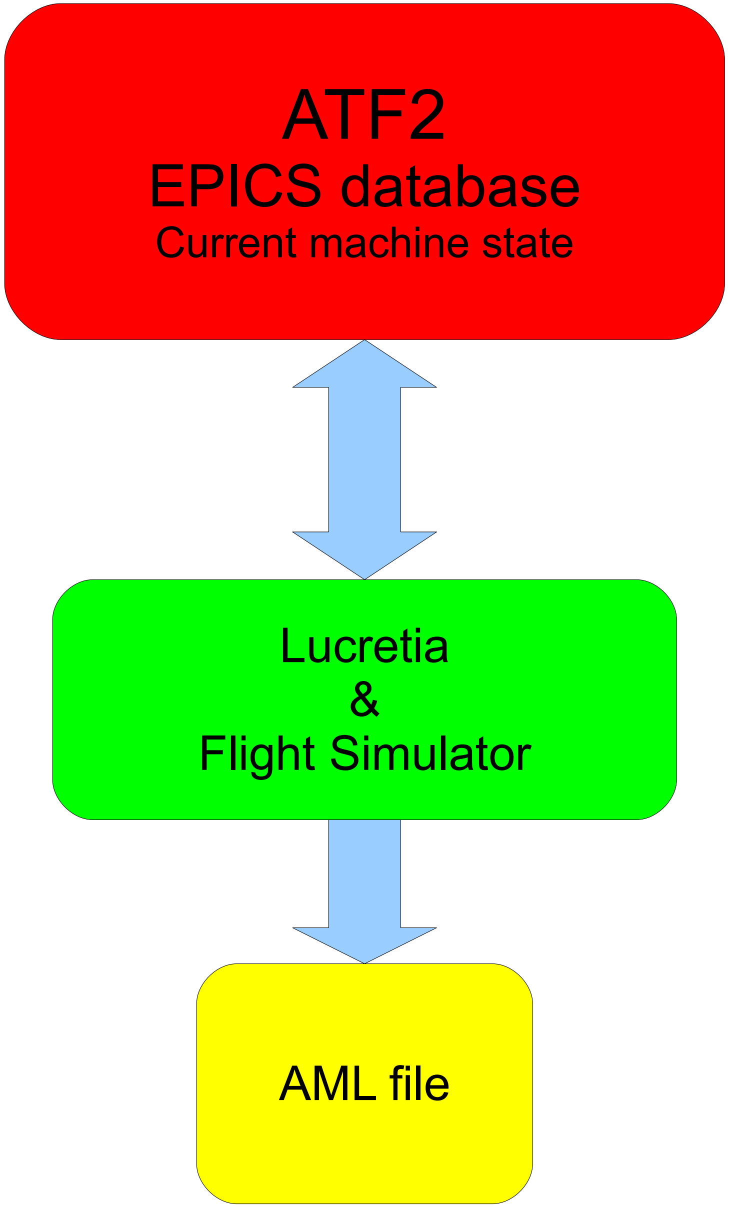
- Detects elements to be “unsplit”.
- Unsplits quads and bends
- Generates
  - BPMs, drifts, instruments, markers, quads, bends, correctors, sextupoles
- Element 6D orientation
- Magnetic field errors
- Power supplies
- Girders
- Flight Sim. Metadata
  - Magnet/mover names
  - Command structure

Status – Still to do

- Higher order magnets (octupoles and higher)
- RF cavities (longitudinal & transverse)
- AML is currently an evolving standard
  - Have to work to keep “up to date”
- Debugging....

AML2Lucretia

- Operation only requires Lucretia2AML
  - Lucretia and EPICS representations are equal
  - Regularly write to AML file for other users
- AML2Lucretia useful in the future
  - Recover the historical machine state



AML2Lucretia – Status

- Partially complete
  - Many elements completed
  - But no power supplies, movers, etc.
- Held up by development of Lucretia2AML
  - Work can now continue with more mature Lucretia2AML
- This is of “lower priority” than the other code

Summary

- Lucretia2AML is an important part of the Flight Simulator
  - Working version demonstrated at recent ATF test and available from Flight Sim. package
    - At the Matlab prompt....
      - *Lucretia2AML('output','outputfilename.aml')*
  - Still some work needed
    - Mostly debugging
- AML2Lucretia is of lower priority, but still important
  - Development version is working, but much work needed.