Philosophy, Principle, and Method for the CombLayer: Day Two

Stuart Ansell

European Spallation Source, Lund, Sweden.

December 8, 2015

DataBases

CombLayer provides two specialized DataBases

- Variable DataBase
- Materials DataBase

Variables

The variable system is the OLDEST part of CombLayer.
Variables:

- need to be set to be used.
- **only** are in the output file if read.
- can be set from the command line
- Can have type and be coded with (!)

Variable setting

Variables are set in a function defined in the setVariables namespace

```

Control.addVariable("LSupplyActive2",31);
Control.addVariable("LReturnPPt0",Geometry::Vec3D(0,30,0));
Control.addVariable("LStemWallMat","Stainless304");

// Add a variable based on something else
Control.addParse<double>("nmxBA0Length","nmxBALength+2.3");
```

Note: Type is inferred in addVariable but must be given in parse.

Usage of Variables

The variable state treated as **VOLITILE**

- All variables should be compile time constructed [if possible]
- Recored variable state within class
- Use a populate function
- FixedComp reads its OWN variables – other values via link-points and other pipe streams
- Bring state of all variable before object construction

Reading Variables

Example of some variables:

```cpp

void
Mybox::populate(const FuncDataBase& Control)
{
  ELog::RegMethod RegA("MyBox","populate");

  leftPhase=Control.EvalVar<double>(keyName+"LeftPhase");
  rightPhase=Control.EvalVar<double>(keyName+"RightPhase");

  nSectors=Control.EvalVar<size_t>(keyName+"NSectors");
  ModelSupport::populateRange(Control,nSectors,
              keyName+"SectAngle",
              leftPhase,rightPhase,sectPhase);

}
```

Make use of composite if appropiate.

Setting variables

Priority of variable setting :

1. Command line
2. xml file
3. default value

Setting variables from the command line

```

    ./simpleBox -r -v MyBoxLength 3.4                \
                   -v MyBoxPoint 'Vec3D(3,4,5)'      \
                   -va MyBoxExtra 5.6                \
                   -v MyBoxTitle 'This is a box'     \
                   AA
```

1. double precision length
2. Vector values
3. Addition of non-existant variable
4. Setting a string

Setting variables from an XML file

Command Line::

```
./simpleBox -r -x MyBox.xml AA
```

XML file:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
   <metadata_entry>
     <Variables>
     <variable name="MyBoxLength" type="double">3.4</variable>
     <variable name="MyBoxPoint" type="Geometry::Vec3D">
       3 4 5 </variable>
     <variable name="MyBoxName" type="std::string">
        This is my Name</variable>

    </Variables>
</metadata_entry>
```

Use the command -X to output an XML file of the variables.

Materials

The materials are handled by CombLayer in a database.
Accessing an existing material is done by **name** [1].

```
// in some variable file:
   Control.addVariable("MyBoxWallMat","Stainless304");

// in populate:
 wallMat=ModelSupport::EvalMat<int>(Control,keyName+"WallMat");

// Use it when constructing an object:
System.addCell(MonteCarlo::Qhull(cellIndex++,wallMat,0.0,Out));
```

wallMat should be an integer.

[1] Legacy number system available – don't even think about it!

Adding your favourite materials

First class materials can be added in DBMaterial.cxx.

```
// Material #112: Aluminum 5251
// (Dave Bellenger version for WaterMod aluminium)
// Total atom density 0.059693
MObj.setMaterial(112,"Alum5251",
 "13027.24c 5.739771e-02 14028.70c 2.307178e-04 "
                 "24000.50c 4.673301e-05 25055.70c 1.474348e-04 "
 "26054.70c 8.477590e-06 26056.70c 1.330800e-04 "
 "26057.24c 3.073398e-06 26058.70c 4.090129e-07 "
 "22046.70c 4.188076e-06 22047.70c 3.776883e-06 "
 "22048.70c 3.742363e-05 22049.70c 2.746363e-06 "
 "22050.70c 2.629604e-06 29063.70c 2.644224e-05 "
 "29065.70c 1.179672e-05 30000.70c 3.714985e-05 "
 "12024.70c 1.263543e-03 12025.70c 1.599624e-04 "
 "12026.70c 1.761186e-04","al.20t",MLib);
setMaterial(MObj);
```

Care with zaids please!!!

Adding your favourite materials (Part 2)

This is too much work! – Make a composite

```

// Mixture of 80% D2O with 20% H2O
Control.addVariable("RodsOuterMat","D2O%H2O%80.0");
```

Materials

The *-matDB* option allows ”multiple” different data-bases, substitution of material names etc.

Object Composition

- Objects in MCNP are only boolean state systems that operate on a point or a track
- Each surface is a *discrete literal*
- Logic of a cell can be expressed as a normal boolean expression

$$1\ \text{-}2\ 3\ \text{-}4\ 5\ \text{-}6\ (\text{-}11 : 12) \rightarrow ab'cd'ef'(g'+h)$$

- Primary importance is to remove literals [not typical]
- Secondary importance is to sequence the logic into maximum surface area first

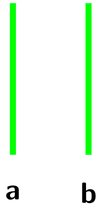
Object Composition

- Objects in MCNP are only boolean state systems that operate on a point or a track
- Each surface is a *discrete literal*
- Logic of a cell can be expressed as a normal boolean expression

$$1 \text{ -2 } 3 \text{ -4 } 5 \text{ -6 } (\text{-11} : 12) \rightarrow \text{ab'cd'ef'(g'+h)}$$

- Primary importance is to remove literals [not typical]
- Secondary importance is to sequence the logic into maximum surface area first

Object Composition

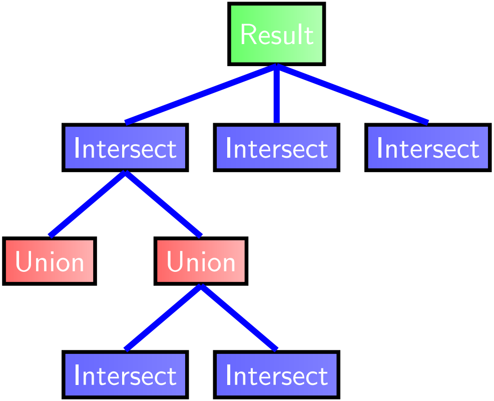


- Use can be made of $b \implies a$ and $a' \implies b'$
- b $\implies$ a := b'+a and a' $\implies$ b' := b+a'
- Add these rules as intersections to the main rule

CombLayer Provides:

- CNF / DNF resequencing [Quine Method]
- Weak boolean algebra division
- Doesn't provide two factor minimization [yet!]
- Selection of minimal literal format

Object Composition



Result

Intersect Intersect Intersect

Union Union

Intersect Intersect

Level 0 : Surface List

Level 1 : Surface List

Level 2 : Surface List

- Maximize level 0 components
- Observe that interaction points can be calculated as level-0 sub units

How to use/achieve object interaction

This is a complex part of Comblayer.