EUROPEAN
SPALLATION
SOURCE

EUROPEAN
SPALLATION
SOURCE

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

Stuart Ansell

European Spallation Source, Lund, Sweden.

December 9, 2015

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

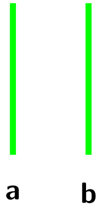
Object Composition

- 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

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

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

Object Composition

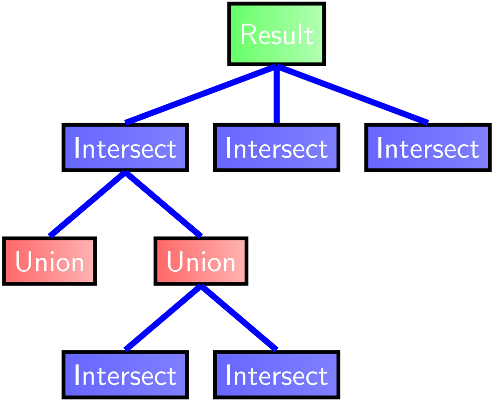


- Use can be made of $b \implies a$ and $a' \implies b'$
- b $\Longrightarrow$ a := b'+a and a' $\Longrightarrow$ b' := b+a'
- Add these rules as intersections to the main rule

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



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.

HeadRule/Rules

Cells (parts of cells) need to be combined

HeadRule provides a way to manipulate the boolean logic.

It can be created either with true surface or pseudo surfaces

```

// HEAD Rule created but no actual surface pointers
HeadRule HR(" 1 -2 3 4 -5");

// Now surface pointer exist:
HR.populateSurf();

ELog::EM<<"Point is in object "<<HR.isValid(Geometry::Vec3D(1,2,3))
<<ELog::endDiag;
```

Support services for combinations

Can be used with intersections, unions and complenets to build complex volumes.

```

HeadRule A(" 1 -2 3 4 -5");
HeadRule B(" (11:1) -9 ");
HeadRule C(" 5 -6 7 ");

A.addUnion(B);
A.addUnion(C.complement());

ELog::EM<<"Cell == "<<A.display()<<ELog::endDiag;
```

This HeadRule can be used to construct a cell.

HeadRule diagnositcs

Very often intersections of line/points are needed.

The thermo-nuclear system is this:

```

const Geometry::Surface* APtr=SMap.realSurfPtr(56);
const Geometry::Surface* BPtr=SMap.realSurfPtr(57);
const Geometry::Surface* CPtr=SMap.realSurfPtr(58);

Geometry::Vec3D XPt=SurInter::getPoint(APtr,BPtr,CPtr,Origin);
```

Calculates the closed point to the intersection of three quadratic surface.

HeadRule diagnositcs

If the line is available:

```

HeadRule A("1 -2 3 -4 5");

std::vector<Geometry::Vec3D>& InterPoints;
std::vector<int>& InterSurfaces;


if (A.calcSurfIntersection(Origin,Axis,
     InterPoints,InterSurfaces))
  {
    ELog::EM<<"Closest surface point == "<<
        SurInter::nearPoint(InterPoints,Origin)<<ELog:endDiag;
  }
```

HeadRule quick and easy methods

Also provides:

- Test for intersection between two objects
- Removal/Substitution of surfaces
- Testing a point with a surface true/false/appropiate

Making object available for work

The point of making persistant objects is to use them later!!

```

makeBoxModel::makeBoxModel() :
  MyBoxObj(new MyBox("Box"))
 /*!
    Constructor
 */
{
  ModelSupport::objectRegister& OR=
    ModelSupport::objectRegister::Instance();

  OR.addObject(MyBoxObj);
}
```

MyBoxObj is now globally available accessed by name.

Global access to components

Any time an attachComp item is required

Getting any attachComp item:

```

ModelSupport::objectRegister& OR=
  ModelSupport::objectRegister::Instance();

const attachSystem::FixedComp* TPtr=
  OR.getObject<attachSystem::FixedComp>("MyBox");

// or if you need the CellMap

const attachSystem::CellMap* CPtr=
  OR.getObject<attachSystem::CellMap>("MyBox");
```

Note that if a FixedGroup etc is used , then the name ”MyBox:BeamLine” will get the sub-component.

Point tallies

- Point tallies are nothing more than a point in space that we want to know the flux at.
- Important points in space are related to link points [normally]
- They are **NOT** part of the model and should be constructed from the command line only

Typical construction:

```

./myBox -r -T point free 'Vec3D(3,4,5)' AA
./myBox -r -T point object MyBox front 5.0 AA
./myBox -r -T point objOffset MyBox front 'Vec3D(5.0,10,20)' AA
./myBox -r -T point help AA
```

Note the 5.0 is an offset – do you really want a point tally ABSOLUTELY on a surface boundary ???

Modifications

The default tally is 99% likely to NOT be what you want
Use -TMod to apply modificationss until it is correct.

```

./myBox -r -TMod particle 0 n p AA
./myBox -r -TMod energy 0 '1e-9 54log 1e3' AA
./myBox -r -TMod movePoint 5 'Vec3D(10,0,0)' AA
./myBox -r -TMod help AA
```

Note the crime of tally multiply cards and other MCNP(X) horrors is NOT supported.

Mesh Tallies/Mesh Geometry

Mesh tallies are possible in MCNP

```

./myBox -r -T mesh free void Vec3D(0,0,0) Vec3D(100,100,100) 20 30 50
./myBox -vtk -r -T mesh free void Vec3D(0,0,0) Vec3D(100,100,100)
      20 30 50 AA.vtk
```

- MCNP only allows one mesh tally. The manual is WRONG.
- A VTK file can be output of the geometry
- More additions are expected to this (I hope)

Construction – using existing components

CombLayer has a number of existing components : All of the following a good solutions

- Just use the component
- Make an adaption to the component to make it more general
- Copy the component and specialize
- Build the component and modify the resulting simualation
- Wrap several components into a bigger component

Global access to components

Many objects exist in models but interesting stuff gets pulled into beamlines or System/construct

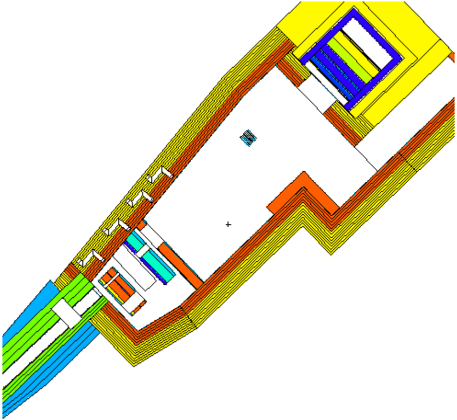
Example of construction – using 5 items [ONLY!!]

Examing the Model/ESSBeam/dreamInc/DREAM.h
It only uses GuideLine, VaccumBox, DiskChopper,
ChopperHousing, and VacuumPipe

Modification of objects

- Modifing components is necessary of many reasons
- Use CombLayers understanding of surfaces/objects to apply geometry transforms
- The two most important are mergeTemplate and LayerDivide3D

LayerProcess



- Layer compartments to improve sampling/variance reduction
- Compartment Used to reduced cell complexity

layerProcess : Preparation

```
void
MyBox::populate(const FuncDataBase& Control)
{

  nLayers=Control.EvalVar<size_t>(keyName+"NLayers");
  ModelSupport::populateDivideLen(Control,nLayers,keyName+"BaseLen",
                                depth,baseFrac);

}
```

Use vectors of fractions (for division).

```

  ModelSupport::surfDivide DA;

  for(size_t i=1;i<nTopLayers;i++)
  {
    DA.addFrac(topFrac[i-1]);
    DA.addMaterial(wallMat);
  }
  DA.addMaterial(wallMat);
```

This tells the surface divider which fractions to create and which materials to assign to each layer

Curtain.cxx : layerProcess

```

  DA.setCellN(getCell("topWall"));
  DA.setOutNum(cellIndex,curIndex+1001);

   ModelSupport::mergeTemplate<Geometry::Plane,
           Geometry::Plane> surroundRule;
  surroundRule.setSurfPair(SMap.realSurf(curIndex+15),
                     SMap.realSurf(curIndex+6));

  OutA=ModelSupport::getComposite(SMap,curIndex," 15 ");
  OutB=ModelSupport::getComposite(SMap,curIndex," -6 ");

  surroundRule.setInnerRule(OutA);
  surroundRule.setOuterRule(OutB);

  DA.addRule(&surroundRule);
  DA.activeDivideTemplate(System);
  cellIndex=DA.getCellNum();
```

This needs talking though !!!!