Particle to MC truth association

Juan Palacios (Nikhef)

LHCb software week

June 16 2009

Outline

- Introduction
- New interfaces
- Implementations
- Examples of use
- Further developments

See also talk by Vava for more details about implementation and examples

Introduction

- For a while MC association has been a problem
  - Legacy code with no expert support
  - Complicated structure
  - Issues with composite particles
  - Requires lots of switching depending on particle type
  - Not easily scalable or debugable
- I personally could never answer users questions about how to use it or why it wouldn't work sometimes
  - Use of BackgroundCategoryTool or completely different approach in LoKi framework

Introduction (2)

- Decided to start from scratch
  - What does a simple user need?
  - What would an expert need?
  - What would a user-friendly interface be like?
- No consideration for implementation
  - Design simple interface(s)
  - Then worry about implementation
  - Too much time wasted in the past trying to write code around problems instead of fixing them.
- Avoid coupling
  - Very clever pieces of code are hard to extend and debug
  - Small stupid pieces can be glued together to do clever stuff

Towards new interfaces

- Many discussions with Gerhard, Patrick, Vanya, Vava
- I opted towards tool interfaces that take a particle and a container of MCParticles and return a set of MCParticles in one format or another
  - Weighted associations: each related MCParticle comes coupled with a weight
    - I have issues with weights in an interface: the value of the weights will depend on the implementation
    - Some steps taken to minimise this effect (see Vava's talk)
  - Un-weighted "Tree" or "Line" structure (explained later)
    - What's the word for a tree from which someone has cut all the branches but one? Decay line?

Towards new interfaces (2)

- Also simple method that returns a single MCParticle
  - Users express interest in method const MCParticle* foo(const Particle*)
- Note: we input a container of MCParticles, trying to avoid behind the scenes assumptions
  - I have kept the default MCParticle container location as default value for the corresponding argument
  - The two argument signatures make the interfaces more useful
    - E.g. select MCParticles from a true decay tree and see if our reco particles are associated to them
- Note: results returned to the user, not put on TES

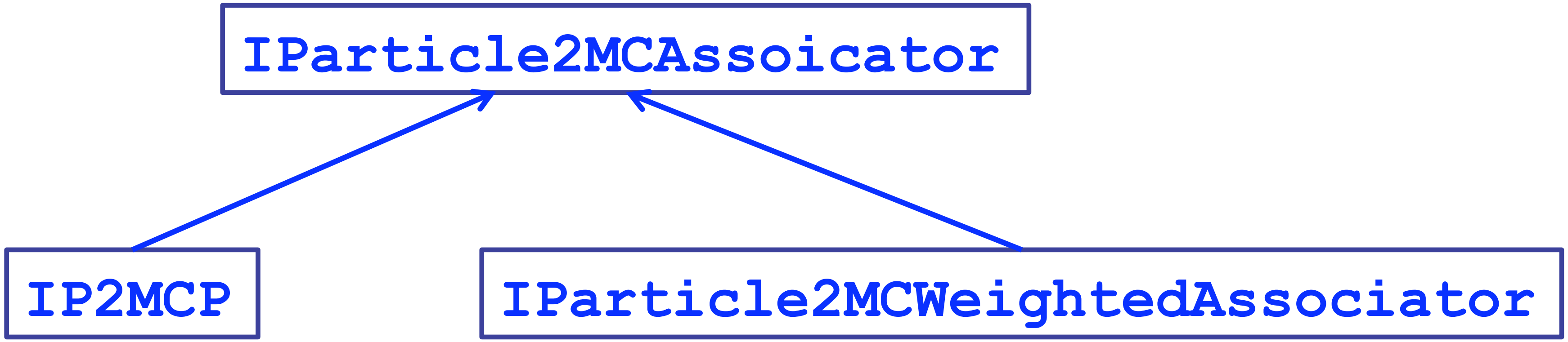
New interfaces: IParticle2MCAssociator

- Simplest user interface:

```
          Calculate and return the LHCb::MCParticle associated to an LHCb::Particle.

virtual const LHCb::MCParticle* relatedMCP(const LHCb::Particle*) const =0

virtual const LHCb::MCParticle* operator()(const LHCb::Particle*) const =0

virtual const LHCb::MCParticle* relatedMCP(const LHCb::Particle*,
                                           const std::string& mcParticleLocation) const =0

virtual const LHCb::MCParticle* relatedMCP(const LHCb::Particle* particles,
                                           const LHCb::MCParticle::ConstVector& mcParticles) const =0
virtual const LHCb::MCParticle* relatedMCP(const LHCb::Particle* particles,
                                           const LHCb::MCParticle::Container&cmcParticles) const =0
```

- Interface speaks for itself.
  - No side-effects should be assumed.
  - In fact, side-effects should be avoided
  - Configurability should be avoided (as with all tools)

IParticle2MCWeightedAssociator and IP2MCP



- Both have relatedMCPs methods
  - IP2MCP's returns vector of vectors of MCParticle*
  - IParticle2MCWeightedAssociator returns vector of MCAssociations (essentially MCParticle*, double pair)

IParticle2MCWeightedAssociator

- Base class **`Particle2MCAssociatorBase`** takes care of implementing all the methods in terms of one or two private methods:
- Either

```
virtual Particle2MCParticle::ToVector
relatedMCPsImpl(const LHCb::Particle* particle,
                const LHCb::MCParticle::ConstVector& mcParticles) const ;
```

- Or both

```
virtual double associationWeight(const LHCb::Particle*,
                                 const LHCb::MCParticle* ) const;

virtual bool isAssociated(const LHCb::Particle*,
                          const LHCb::MCParticle* ) const;
```

- Developers only need to implement this, the rest is taken care of

IP2MCP

- Base class **`P2MCPBase`** takes care of implementing all the methods in terms of two methods:
- Public method

```
virtual  bool isMatched(const LHCb::Particle* particle,
                        const LHCb::MCParticle* mcParticle) const ;
```

- Private method

```
virtual P2MCP::Types::FlatTrees
 sort(const LHCb::MCParticle::ConstVector& mcParticles) const;
```

- The **`FlatTrees`** is just a vector of vectors
- The **`sort`** method will be removed and a proper **`Trees`** or **`DecayLines`** class with self-sorting written in
- Developers only need to implement these two methods, the rest is taken care of

Existing implementations (1)

- **`IParticle2MCWeightedAssociator`**
  - **`P2MCPFromProtoP`**: For stable charged or neutral particles that have a valid **`ProtoParticle`**. Simply gets the weights from the linkers to implement the **`relatedMCPImpl`** method.
  - **`DaVinciSmartAssociator`** (**Vava**): uses the above for charged and neutral particles, and **`BackGroundCategoryTool`** for composite particles. Implements the **`relatedMCPImpl`** method.
- Both implementations require that the **`MCParticles`** are on the TES
  - If an input **`MCParticle`** container is passed, the base class checks for the overlap with **`MCParticles`** from the standard TES location
- See Vava's talk for more details on the **`DaVinciSmartAssociator`** and other issues

Existing implementations (2)

- **`IP2MCP`**
  - **`MCMatchObjP2MCRelator`**: Performs the association in two steps.
    1. Decide if an **`LHCb::MCParticle`** is matched to an **`LHCb::Particle`** using Vanya's **`LoKi::MCMatch`**.
    2. Use other **`LoKi`** MC Truth components to split and sort the resulting set of associated **`MCParticles`** into decay sets (**`FlatTrees`**), such that each set contains only **`MCParticles`** that come from different hierarchy levels in the same decay. Order is mother to daughter.
- This implementation *might* require that stable **`MCParticles`** are on the TES to bootstrap itself. However, it can be given as input a relations table matching stable **`Particles`** to **`MCParticles`**
  - This is done in the **`MicroDST`**

MCMatchObjP2MCRelator

- More details are given here
  - https://twiki.cern.ch/twiki/bin/view/LHCb/Particle2MC#MyAnchor0
- The important point is that a true or false matching is performed in an iterative way
  - No need for decay descriptors
  - Builds up from stable particles
- For illustration, **`FlatTrees`** for correctly matched reconstructed particles could be
  - **`Tree for particle mu+: B_s0, J/psi(1S), mu+`**
  - **`Tree for particle phi(1020): B_s0, phi(1020)`**
- More than one **`FlatTree`** is a symptom of more than one **`MCParticle`** making a contribution to one reconstructed **`Particle`**

Examples of use: C++

```
// here you can use any of the implementations, DaVinciSmartAssociator,
// MCMatchObj2MCRelator or P2MCPFromProtoP if only interested in stable particles,
// since they all implement IParticle2MCAssociator
m_assoc = tool<IParticle2MCAssociator >(MCMatchObjP2MCRelator);
const LHCb::Particle part = ...;
const LHCb::MCParticle* assocMCP = m_assoc->relatedMCP(part,
                                            LHCb::MCParticleLocation::Default);

// Or use the DaVinciSmartAssociator to get a range of MCAssociations
// header file
IParticle2MCWeightedAssociator* m_assoc;
// implementation
m_assoc = tool<IParticle2MCWeightedAssociator>(DaVinciSmartAssociator);
const LHCb::Particle part = ...;
Particle2MCParticle::ToVector assocMCPs = m_assoc->relatedMCPs(part,
   LHCb::MCParticleLocation::Default);
Particle2MCParticle::ToVector::const_iterator iAssoc = assocMCPs.begin()
for ( ; iAssoc != assocMCPs.end() ; ++ iAssoc ) {
  verbose() << "Associated MCParticle at " << iAssoc->to()
     << " with weight " << iAssoc->weight() << endmsg;
}
```

Examples of use: Python

```
#
appMgr = AppMgr(outputlevel=4)
toolSvc = appMgr.toolsvc()
# get an instance of MCMatchObjP2MCRelator and DaVinciSmartAssociator.
MCMatchTool = toolSvc.create('MCMatchObjP2MCRelator', interface='IP2MCP')
SmartAssoc = toolSvc.create('DaVinciSmartAssociator',
                            interface = 'IParticle2MCWeightedAssociator')
# loop over particles
evtSvc = appMgr.evtSvc()

while ( ... ) :
   particles = evtSvc[particlePath]
   for p in particles :
      trees = MCMatchTool.relatedMCPs(p)
      for tree in trees :
         for mcp in tree :
             print "found MCParticle ", mcp
      bestMatchedMCP = MCMatchTool.relatedMCP(p)
      print "Best match is ", bestMatchedMCP
      mcAssociations = SmartAssoc.relatedMCPs(p)
      for mcAssoc in mcAssociations :
```

Populating the TES

- DaVinciAssociators would result in population of Particle->MCParticle linkers on the TES
  - Now this only happens for Particle->ProtoParticle
    - Not by design: this could disappear any time
- Approach now is to instrument a GaudiAlgorithm with an associator tool, give it some inputs, and get it to write a relations table to the TES
  - One working example: **`P2MCRelatorAlg`** using **`IP2MCP`**
    - Now used for MicroDST (example in this morning's talk)

Future work

- MCMatchObjP2MCRelator:
  - Optimise the getting of single, "best" **`MCParticle`** out of **`FlatTrees`**
  - Write self-sorted **`FlatTrees`** or **`DecayLines`** class
    - Remove sort method from P2MCPBase
- Write chi2 based implementations of all interfaces
  - Vava's found where the old chi2 went wrong
  - Only need to implement **`Particle2MCAssociatorBase's`**

```
virtual double associationWeight(const LHCb::Particle*,
                                 const LHCb::MCParticle* ) const;

virtual bool isAssociated(const LHCb::Particle*,
                          const LHCb::MCParticle* ) const;
```

- DaVinciSmartAssociator: See Vava's talk

Wiki

- Visit the wiki:
  - https://twiki.cern.ch/twiki/bin/view/LHCb/Particle2MC