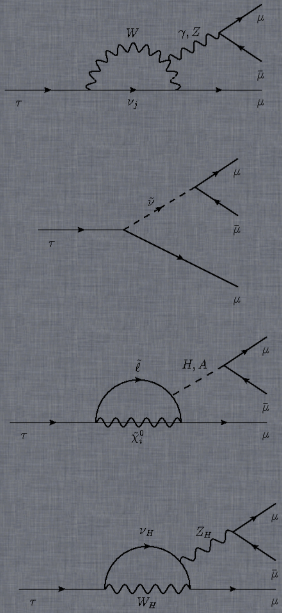
Final State Radiation correction in MC truth matching

Marcin Chrząszcz

Institute of Nuclear Physics,
Polish Academy of Science,
and INFN sezione di Pisa

27th February 2013

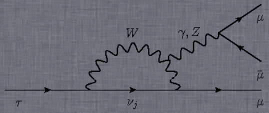


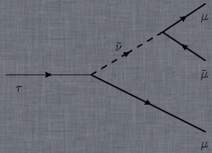
Truth matching description

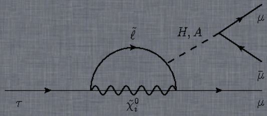
Code structure

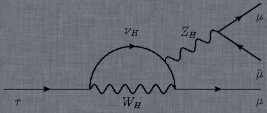
Tests

BtaTupleMaker









How does truth matching work?

Let's say we what to study a decay of $\mathrm{D}^0 \to \pi K$.

1. From π and D^0 you construct the D^0 with background.
2. At some point you may want to know how many actual D^0 survived the selection. Then you do in the code (or some module does it for you) mcFromReco(...).
3. However the number this function returns will be an underestimation of the actual number.
4. If FSR happen you will end up with a channel $\mathrm{D}^0 \to \pi K \gamma$, which software will not match with search channel: $\mathrm{D}^0 \to \pi K$.

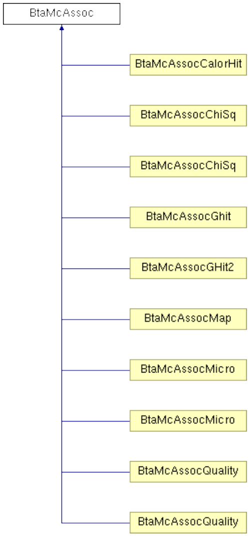
How does truth matching work?

How ever code does one thing for you:

1. For example if you have a decay: $\tau^- \to \pi^+\pi^-\pi^-\nu_\tau$
2. Of coz there is no ν BtaCandidate list ☺ .
3. Same thing was implemented for FSR correction.
4. The new code should not create problems with the old modules etc.

Cod structure

- The Top class is BtaMCAssoc, however the function is virtual ☹
- It's also defined in BtaMcAssocGHit, BtaMcAssocMicro, BtaMcAssocQuality, BtaMcAssocQuality, BtaMcAssocGHit
- They are distributed over the packages: BetaCoreTools, BetaMC and BetaMicroTruth.
- Call mcFromReco is from BtaMcAssocMicro. Then it's checked that particle is a composite and if it is then mcFromRecoComposite is called.



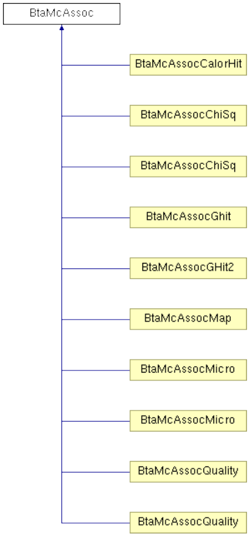
Cod structure

The original code:

Listing 1: Oryginal declaration:

```
    // Abstract interfaces
    virtual BtaCandidate* recoFromMC(
        BtaCandidate const *mc, int
        which = 0, bool checkMassHypo=
        false) const = 0 ;

  virtual BtaCandidate*
      mcFromRecoComposite(BtaCandidate
        const *reco, int which =0, bool
        checkMassHypo=false) const;
```



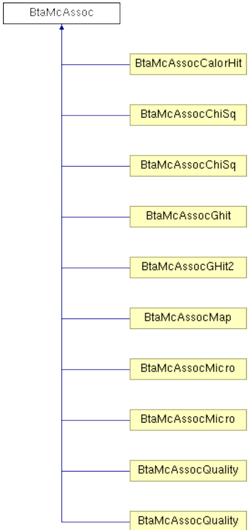
Cod structure

Listing 2: Implement switch:

```
    // Abstract interfaces
    virtual BtaCandidate* recoFromMC(
        BtaCandidate const *mc, int
        which = 0, bool checkMassHypo=
        false, bool ignoreFSR=false)
        const = 0 ;

  virtual BtaCandidate*
      mcFromRecoComposite(BtaCandidate
        const *reco, int which =0, bool
        checkMassHypo=false, bool
      ignoreFSR=false) const;
```

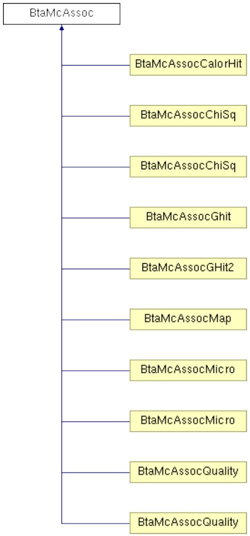
Definition of those functions have to be changed in every inheritance class.
There is a default argument set to false so earlier code written by others will not see the difference.



Cod structure

How does the code work? For details see the backup slides. To make long story short:

- Software creates a std::map between daughters and mothers.
- If there is a missing particle in the candidate daughters list it checks if it is a γ. If it is than it ignores it and returns the correct pointer.



Tests

For $\mathrm{D}^0 \to \pi K$ there exists a working code in the tutorial:

LINK

Before the modifications

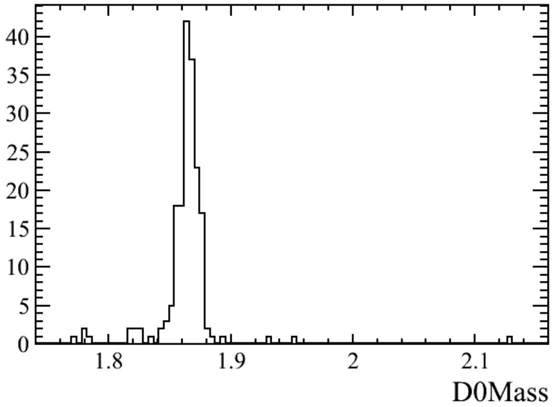


Figure: 183 events in data sample

After modifications and ignoreFSR==true

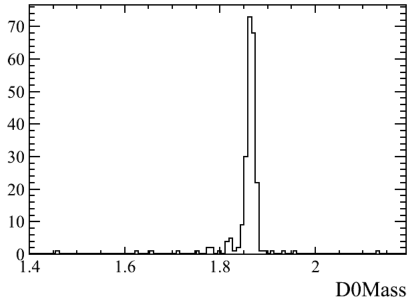


Figure: 230 events in data sample

Many checks have been done (backup slides) to see if this is working properly. ie. if switched off it behaves exactly as previous code and if switched on it really takes the correct particles.

BtaTupleMaker

- If someone uses BtaTupleMaker for ntuple production the BtaTupleMaker does the truth matching for him by calling mcFromReco.
- Can ignore the FSR in the tcl file (if not set it is false by default and code acts exactly as previous):

Listing 3: Implement switch:

```
talkto BtuTupleMaker {
    listToDump set myB0List
    ignoreFSR set true
    fillMC set true

    eventBlockContents set "EventID CMp4 BeamSpot "
    eventTagsBool      set "BGFMultiHadron B0ToD0barNonCPKPi
        "
    eventTagsInt       set "nTracks nGoodTrkLoose"
    eventTagsFloat     set "R2All"

    mcBlockContents    set "Mass CMMomentum Momentum Vertex"
```

Summary

- Code well tested.
- Works well with ana52.
- documented on wiki page: CLIC

Only thing that still needs to be done is to check if it works with ana53.

Thank you for your attention

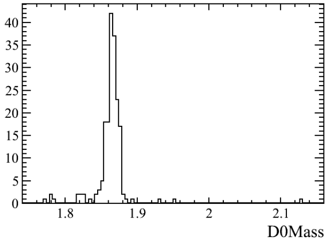
BACKUP Slides

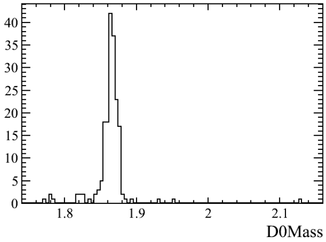
Real code

Listing 4: Implement switch:

```
if(aMother != 0 && aMother->nDaughters() != listOfDaughters.
    size()) {
      // the mother has more daughters than on the list
// something not impotent ...
  if(FSR)
          {
            if( dau->pdtEntry() != 0 )
              {
                string name( dau->pdtEntry()->name() );
                if( name.find( "gamma" ) != string::npos){
                  std::cout<<"SUCCESSSSS!!!!!!!, we found
                      gamma"<<std::endl; // to be commented
                      out
                  continue;
                }
                  } // if we have pdt entry!
          } // end of FSR
```

Consistency test old vs new with FSR == false

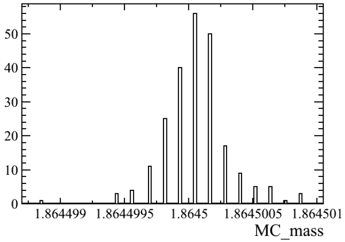




Spot the difference ☺. They have the same number of events: 183.

Consistency test 2

Do we really get truth D0 in ntuple? Of coz I checked the lund but this plots is nicer to show...



I don't know different particle with mass so close do D^0 so i think I can declare it that it works.