The Data Model: Some Considerations

Ties Behnke, January 15, 2003

1 The Monte Carlo Information

The Monte Carlo Information has two parts:

- The generator particles
- Particles produced by GEANT during the showering and tracking

To control the number of tree particles to be stored clearly a selection has to be made on the ones produced by GEANT.

Ideally this selection would be

- Only store MC tree information up to the point where the particles start showering
- Once they start showering, save for each particle in the shower / each hit in the shower only the pointer to the last tree particle before the shower started

This definition has a number of problems:

- How does one define “start of shower”
- How does one treat the case where from a shower an “energetic” particle is produced and induces a second, separable shower?
- How does one treat the connection between two physically distinct calorimeters? Again individual particles punching through one calorimeter might start a second shower.

An approximation to the ideal definition above might be the following:

- Store the MC history including decays of all particles up the point where they physically enter a calorimeter volume.
- After this only store the pointer to the particle at the entry point.

This scheme does not a priori treat:

- Secondary showers
- Particles which backscatter from the calorimeter into the tracking volumes: such particles will have no clear history

Even in this scheme the number of particles to store in the MC tree is fairly large. It is dominated by photons and electrons produced in the transport through the tracking volumes and the associated materials. It should therefore be possible to apply an energy cut to the particles, below which they are not stored in the tree. Hits produced by such particles should point to the parent particle.

A simple way to implement such a scheme might be to define two types of “mother volumes” for the detector, one delimiting the tracker volumes (in which the history of a particle is stored), one delimiting the calorimetric part (in which the history is not fully stored). If a particular particle crosses from the tracking type into the calorimeter type volume, its history will be truncated at this point. If a particle crosses in the opposite way, from a calorimeter into a tracking type volume, the recording of its history will resume. The history record will then follow the particle in the second tracker volume, jump across the calorimeter volume to the particle initiating the shower in the calorimeter, and follow the particle through the first tracking volume.

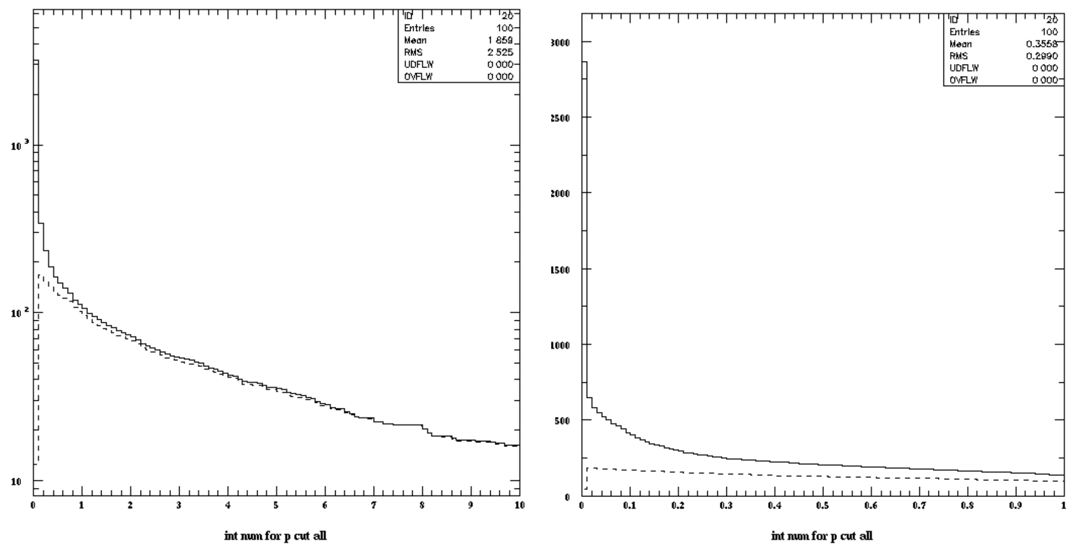

Figure 1: Total number of MC objects as a function of momentum cut

In Figure 1 the expected number of objects to be stored is shown, as a function of the momentum cut applied to the secondary particles produced. The plots were made for tt-events at 500 GeV. Shown are: (solid line) all particles in the MC tree, (dashed line) only the primary particles. The right plot shows the region below 1 GeV expanded, on a linear y scale. A moderate cut on the momentum of the secondaries, well below 1 GeV, can reduce the total number of objects to be stored significantly. Something like 100 MeV might be a good compromise.

In Figure 2 the particle composition of the secondary and the primary particles is shown. The particle code used is that of GEANT. It is obvious that electrons and photons dominate the secondary particles; nuclear interactions play only a less important role.

A possible concrete proposal would therefore be:

- Store all primary tree particles
- Store all secondary tree particles if they have an energy of more than 0.1 GeV

In this way the average number of tree particles will be around 300-400, only about 50% larger than the primary event multiplicity.

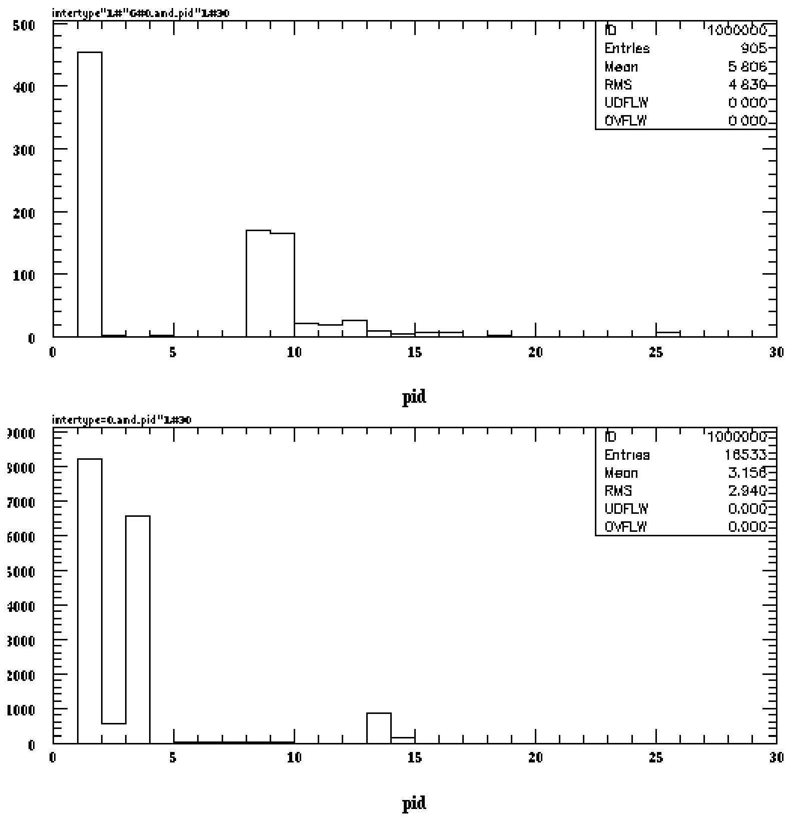

Figure 2: Particle composition of (top) primary and (bottom) secondary particles. The particles codes are: 1=photon, 2=positron, 3=electron, 8=pion+, 9=pion-, 10=K0L, 11=K+, 12=K-, 13=n, 14=p, 15=Anti-p, 16=K0s

2 The Data Model for Reconstructed Objects/ Particles

2.1 The data model in data.xml, dated 1-15-2003

The current data model described in the file data.xml knows about two different types of reconstructed entities: the block ReconstructedParticle, and the block ReconstructedObject.

ReconstructedParticle: Every object reconstructed from signals in the detector and which is assigned by the reconstruction program to be an individual entity is a member of the ReconstructedParticle. These objects are the primary output objects of the particle flow algorithm. They are typically constructed from a list of tracks and calorimeter objects, but are supposed to represent only one particle in the detector. They are constructed only from those particles, which themselves leave signals in the detector.

ReconstructedObject: Anything, which is a compound object, is a ReconstructedObject. ReconstructedObject Members are constructed from ReconstructedParticles. The same ReconstructedParticle can be a member of more than one ReconstructedObject. A mechanism should be probably foreseen which allows recursion: ReconstructedObjects can have as members other ReconstructedObjects, or a mixture of ReconstructedObjects and ReconstructedParticles. A number of different types of ReconstructedObjects can be defined, which may have, in addition to the standard properties and methods, specific ones (example: the Jet ReconstructedObject might include information about the particular jet-finding algorithm used, about typical cutoff parameters etc. A VERTEX ReconstructedObject will include more information about the particles, which are a member of the vertex, and about the quality of the assignment to this vertex).

The advantage of this scheme is that it separates very cleanly between the building blocks – the reconstructed particles – and the compound, derived objects – things like jets, V0's, etc etc. It allows the person working with the data to build up its own chain of ReconstructedObjects, while conserving the underlying result of the reconstruction.

The disadvantage of this scheme is that the distinction between a ReconstructedParticle and a ReconstructedObject is in some cases artificial. Whether a K0s should go into the ReconstructedParticle category or into the ReconstructedObjects category is a matter of discussion. This scheme also makes it harder for the user to work with "particles at the IP" as opposed to particles reconstructed – for many analyses however the relevant quantities are "particles at the IP".

2.2 An alternative data model

Another approach to the data model does away with the separation between ReconstructedParticles and ReconstructedObjects. It only knows about ReconstructedParticles in its furthest sense. Everything is treated as a ReconstructedParticle, which are connected by a parent – daughter relationship. A flag / name is used to distinguish different types of Objects from each other: Particle, Jet, Vertex etc..

The advantage of this scheme is that philosophically this is closer to the goal of the reconstruction: reduce everything to the few primary partons / particles in the event. Looping over all particles at the IP is as simple as looping over all Objects checking on a flag whether or not this object is defined at the IP or not. However the user will still have to account for double counting himself (e.g., jets and particles will be defined both at the IP, and the user has to decide whether he wants to loop over jets or over particles).

In this scheme a K0s would be just another ReconstructedParticle, which has two daughters, the two-pion tracks, from which is was constructed. Flags would be used to separate the different types. A jet would be jet another type of ReconstructedParticle, which has a list of daughters, pointing to other ReconstructedParticles.

The disadvantage of this model is that a more complicated scheme of flags etc etc is needed. The user has to navigate these flags even in the simplest cases. There is no longer a clear logical separation between things reconstructed and things constructed from reconstructed objects – but then: is this needed?

2.3 A compromise scheme

A possible compromise between both approaches might be the following:

- ReconstructedParticles are all objects, which are the result of the reconstruction of detector signals.
- Any Object which is constructed from ReconstructedParticles, and which REPLACES the original objects, is considered another Reconstructed Particle. In this category, V0's like K0s or Lambdas would live: the new particle in a certain sense replaces the two (or more) ReconstructedParticles used in its construction. Using a simple bit the two (or more) ReconstructedParticles will be flagged as decay products, and a new Reconstructed Particle, the K0s for example, will be added to the collection.
- Everything which is constructed from ReconstructedParticles, and which forms a compound object, but which does not replace the original ReconstructedParticles, is a ReconstructedObject. Jets, Vertices etc fall into this category. Typically Reconstructed Objects are build by the user at the time of an analysis, though it is of course possible that certain default objects will be constructed also in advance. Usually however such objects would not be persisted together with the rest of the event, but rebuild at analysis time.