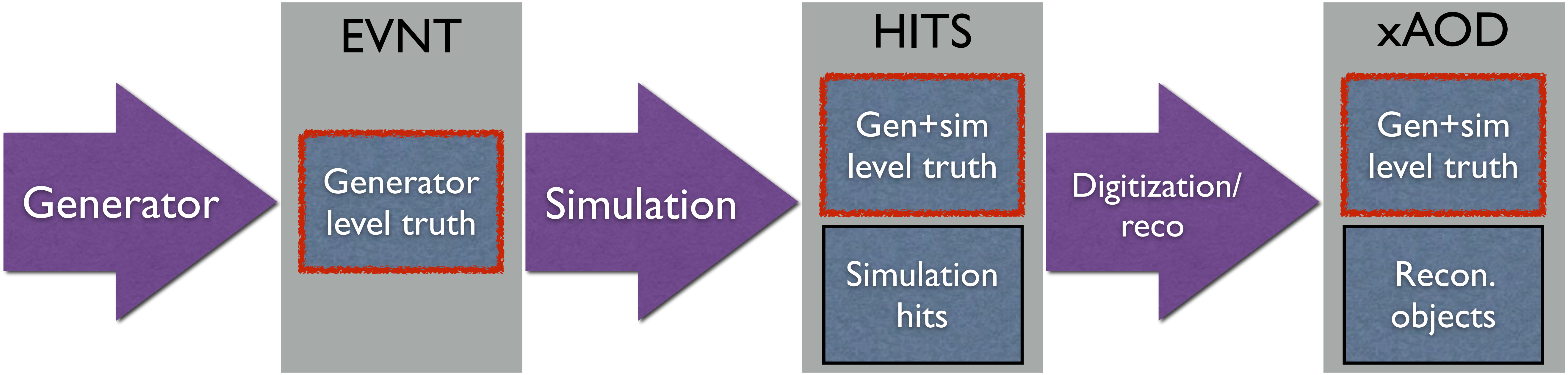
Monte Carlo Truth in the xAOD

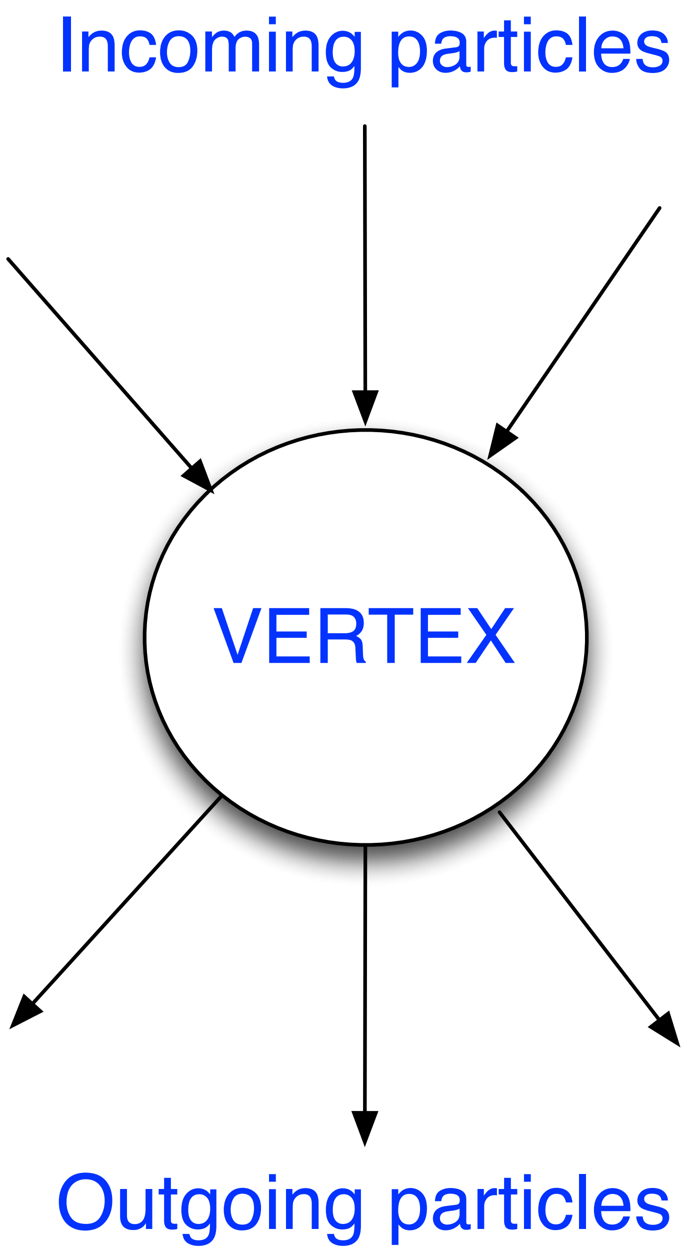
James Catmore (UiO)

What is truth?

- The record of particles that were really in your events
  - Output from event generators (Pythia, Herwig, Sherpa): also inputs to the detector simulation
  - Additions by the detector simulation (material interactions, photon conversions, decays): also inputs to digitization and reconstruction
- Truth is kept in the reconstructed MC simulation and can be matched with reconstructed objects
  - You can ask a reconstructed object what "true" particles it corresponded to

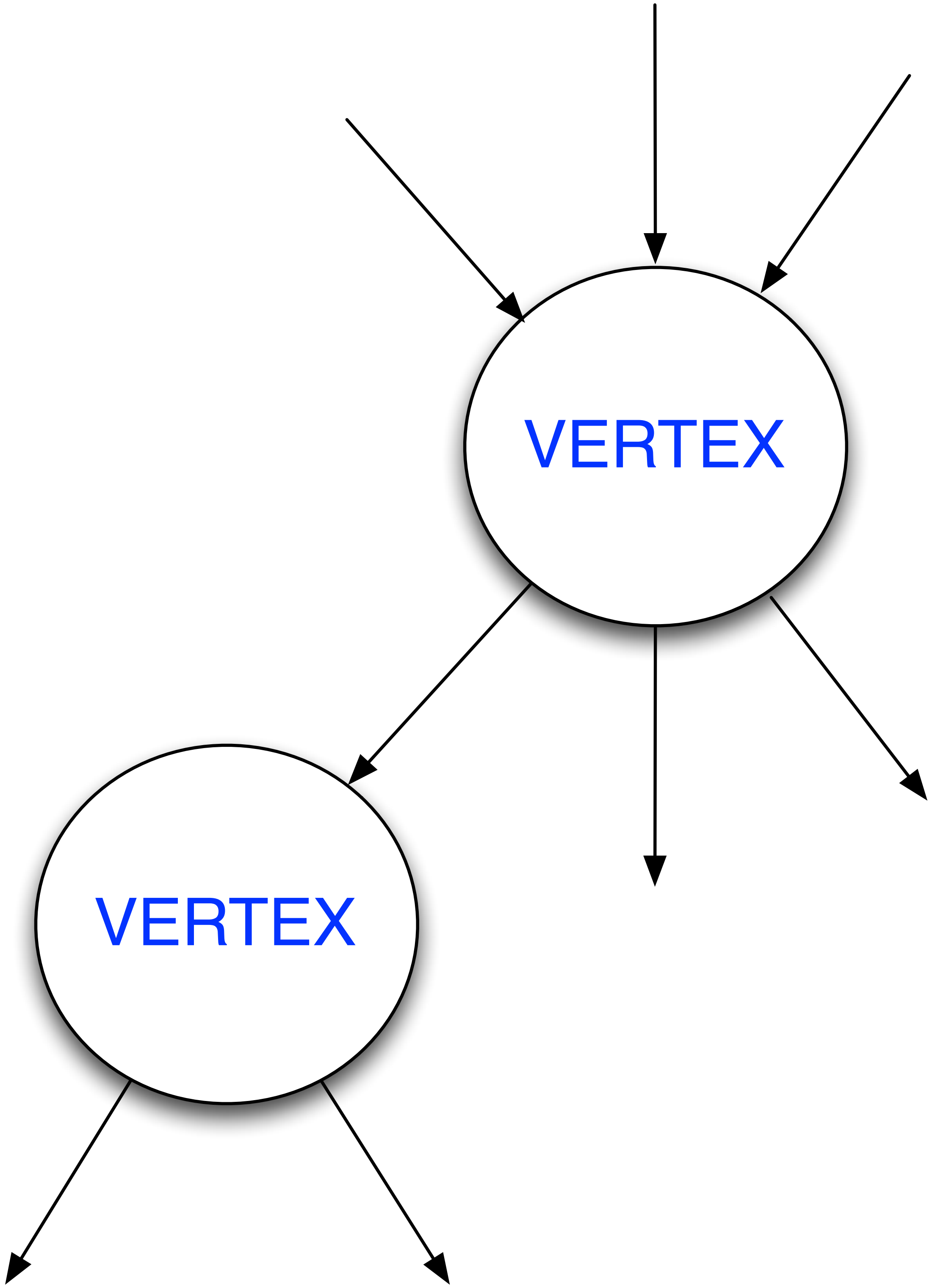


How is truth represented?



- Every interaction is encoded as
  - a list of *incoming particles*
  - an *interaction vertex*
  - a list of *outgoing particles*
- *Particles* have kinematic and type (e.g. μ, K, π, W, Z, H) properties
- *Vertices* have temporal and spatial properties (when and where the interaction happened)
- Both classes have links to each other
- Both have unique *barcodes*

How is truth represented?



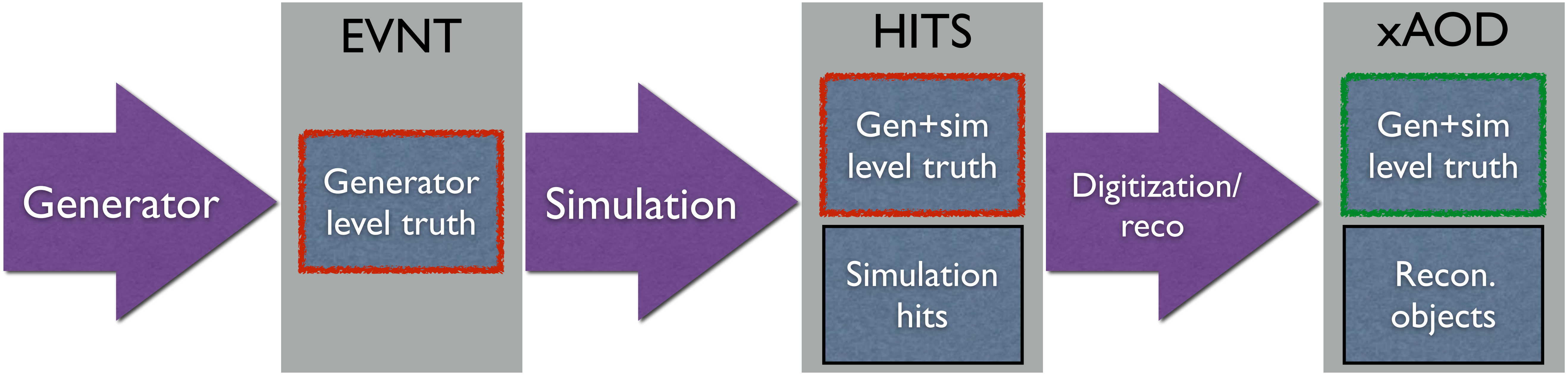
- The outgoing particles for one vertex may be in the incoming particles for another
  - The vertex that a particle "comes out of" is its *production vertex*
  - The vertex that a particle "goes into" is its *decay vertex*
  - A stable particle has no decay vertex
  - **Particles may *loop***
- A given *event* will contain hundreds of particles and vertices in long chains
- Frequently users are most interested in the particles alone
- Different generators provide very different information about "truth"

Generator Myths

- The generator event record is NOT a connected tree of branchings
  - There may be loops, breaks, particles may disappear or appear…
  - Some generators in particular (Sherpa) omit some particles (Zs and Ws)
  - This is why you ALWAYS want to look at observables if they are available!!

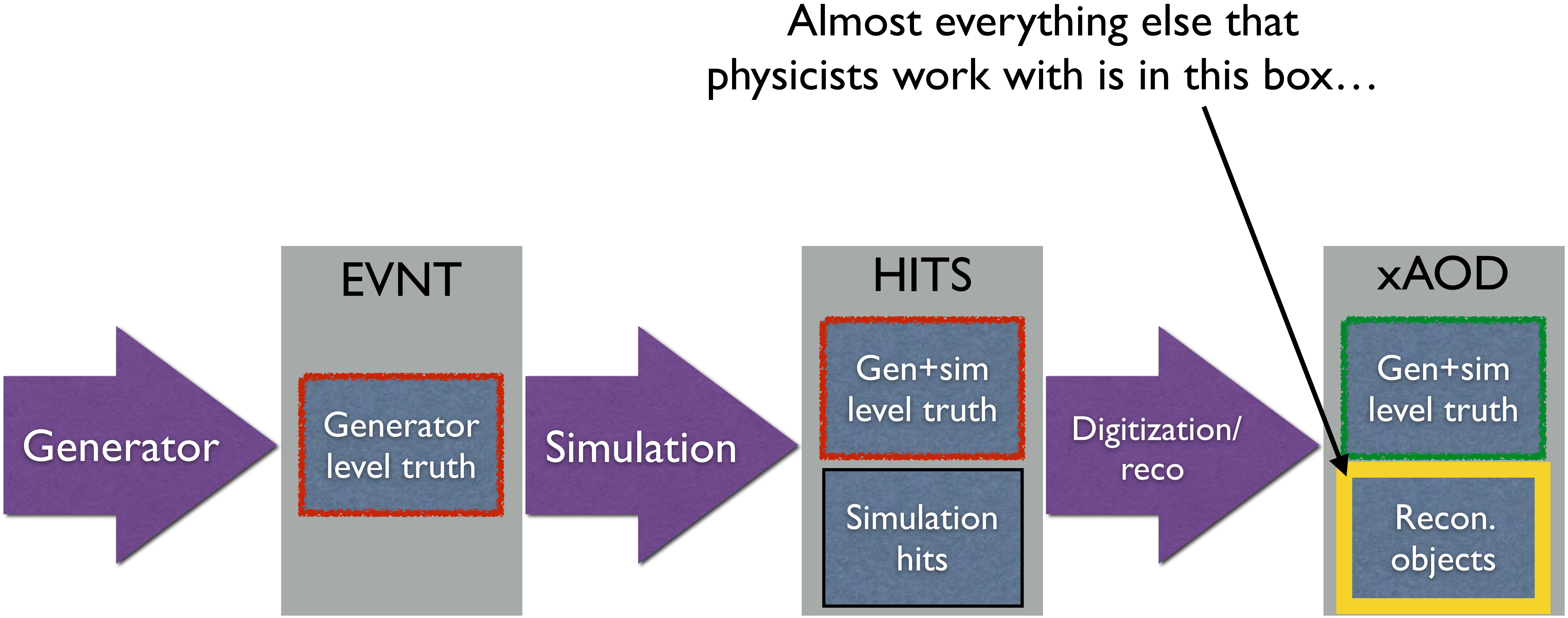
Truth formats

- There are two Monte Carlo truth structures in use in ATLAS
  - **HepMC**
    - independent of ATLAS: used widely across the HEP community
    - is the output format for many event generators and the input to many truth-level tools (e.g. Rivet)
    - not directly readable in ROOT
  - **xAOD truth**
    - internal to ATLAS
    - used in the xAOD format but NOT the EVNT and HITS (HepMC still used here)
    - directly readable in ROOT (click and look - like all xAOD containers)



Truth formats

- BUT: xAOD mimics the HepMC vertex-particle model described previously
  - once you've got to grips with one, you should be able to deal with both
  - you may never need to use HepMC unless you are involved with generator development/validation, or work on interpretation of results using software such as Rivet



Almost everything else that physicists work with is in this box…

xAOD truth objects

- Three fundamental classes
  - xAOD::TruthEvent
  - xAOD::TruthParticle
  - xAOD::TruthVertex
- The particle and vertex encode the structure shown previously
- The event is used to demarcate which hard scatter interaction a set of particles/vertices belongs to (and contains common information such as beam energy and PDF information)
- Some of the most important accessors shown on the next few slides

xAOD::TruthEvent

```
const xAOD::TruthEventContainer* xTruthEventContainer = NULL;
CHECK( evtStore()->retrieve( xTruthEventContainer, m_xaodTruthEventContainerName));
```

Container

```
xAOD::TruthEventContainer::const_iterator itr;
for (itr = xTruthEventContainer->begin(); itr!=xTruthEventContainer->end(); ++itr) {

    std::pair<const xAOD::TruthParticle*,const xAOD::TruthParticle*> beamParticles = (*itr)->beamParticles();
```

Beam particles ★

```
    const std::vector<float> weights = (*itr)->weights();
```

Event weights

```
    int id1(0); (*itr)->pdfInfoParameter(id1,xAOD::TruthEvent::id1);
    int id2(0); (*itr)->pdfInfoParameter(id2,xAOD::TruthEvent::id2);
    int pdfId1(0); (*itr)->pdfInfoParameter(pdfId1,xAOD::TruthEvent::pdfId1);
    int pdfId2(0); (*itr)->pdfInfoParameter(pdfId2,xAOD::TruthEvent::pdfId2);
    float x1(0.0); (*itr)->pdfInfoParameter(x1,xAOD::TruthEvent::x1);
    float x2(0.0); (*itr)->pdfInfoParameter(x2,xAOD::TruthEvent::x2);
    float scalePDF(0.0); (*itr)->pdfInfoParameter(scalePDF,xAOD::TruthEvent::scalePDF);
    float pdf1(0.0); (*itr)->pdfInfoParameter(pdf1,xAOD::TruthEvent::pdf1);
    float pdf2(0.0); (*itr)->pdfInfoParameter(pdf2,xAOD::TruthEvent::pdf2);
```

PDF information ★

```
    float scale = (*itr)->eventScale();
    float qcd = (*itr)->alphaQCD();
    float qed = (*itr)->alphaQED();
```

Scales and constants ★

```
    int nVert = (*itr)->numTruthVertices();
    int nPart = (*itr)->numTruthParticles();
    const xAOD::TruthVertex* vertex = (*itr)->truthVertex(iVtx);
    const xAOD::TruthParticle* particle = (*itr)->truthParticle(iPart);
}
```

Access to constituent particles and vertices

★Warning! May not always be sensible! Don't be shy in asking for help!

xAOD::TruthVertex

```
const xAOD::TruthVertex* vertex = event->truthVertex(iVtx);
```

ALWAYS access from the event (also possible from the container directly but there is a very good reason why not to do this: see later)

```
int barcode = vertex->barcode();
```

Barcode

```
int id = vertex->id();
```

ID

```
float x = vertex->x();
float y = vertex->y();
float z = vertex->z();
float t = vertex->t();
```

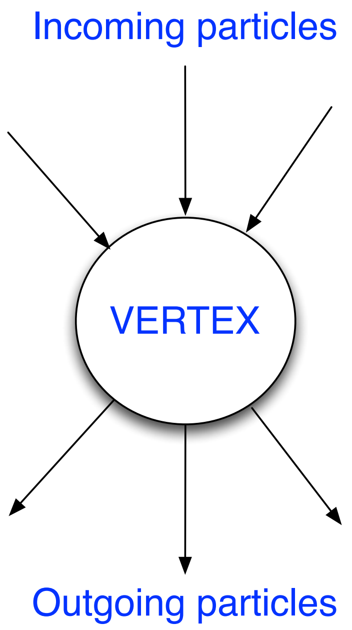
Spatial/ temporal

```
std::vector<float> weights = vertex->weights();
```

Weights

```
int vertex->numIncomingParticles();
const xAOD::TruthParticle* particle = vertex->incomingParticle(iPIn)
int vertex->numOutgoingParticles();
const xAOD::TruthParticle* particle = vertex->outgoingParticle(iPIn)
```

Incoming and outgoing particles



xAOD::TruthParticle

```
const xAOD::TruthParticle* particle = event->truthParticle(iPart);
```

ALWAYS access from the event (also possible from the container directly but there is a very good reason why not to do this: see later)

```
int barcode = particle->barcode();
```

Barcode

```
int status = particle->status();
```

Status

```
int pdgId = particle->pdgId();
```

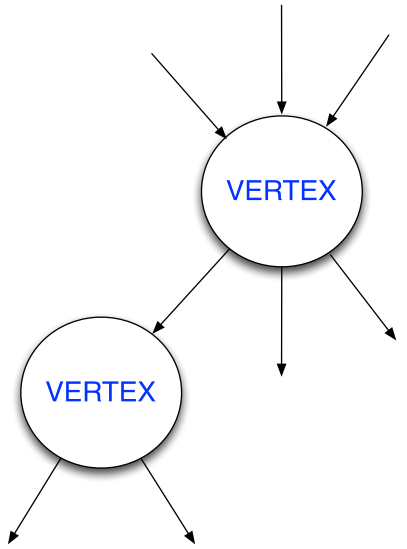
PDG ID

```
float px = particle->px();
float py = particle->py();
float pz = particle->pz();
float e = particle->e();
float m = particle->m();
```

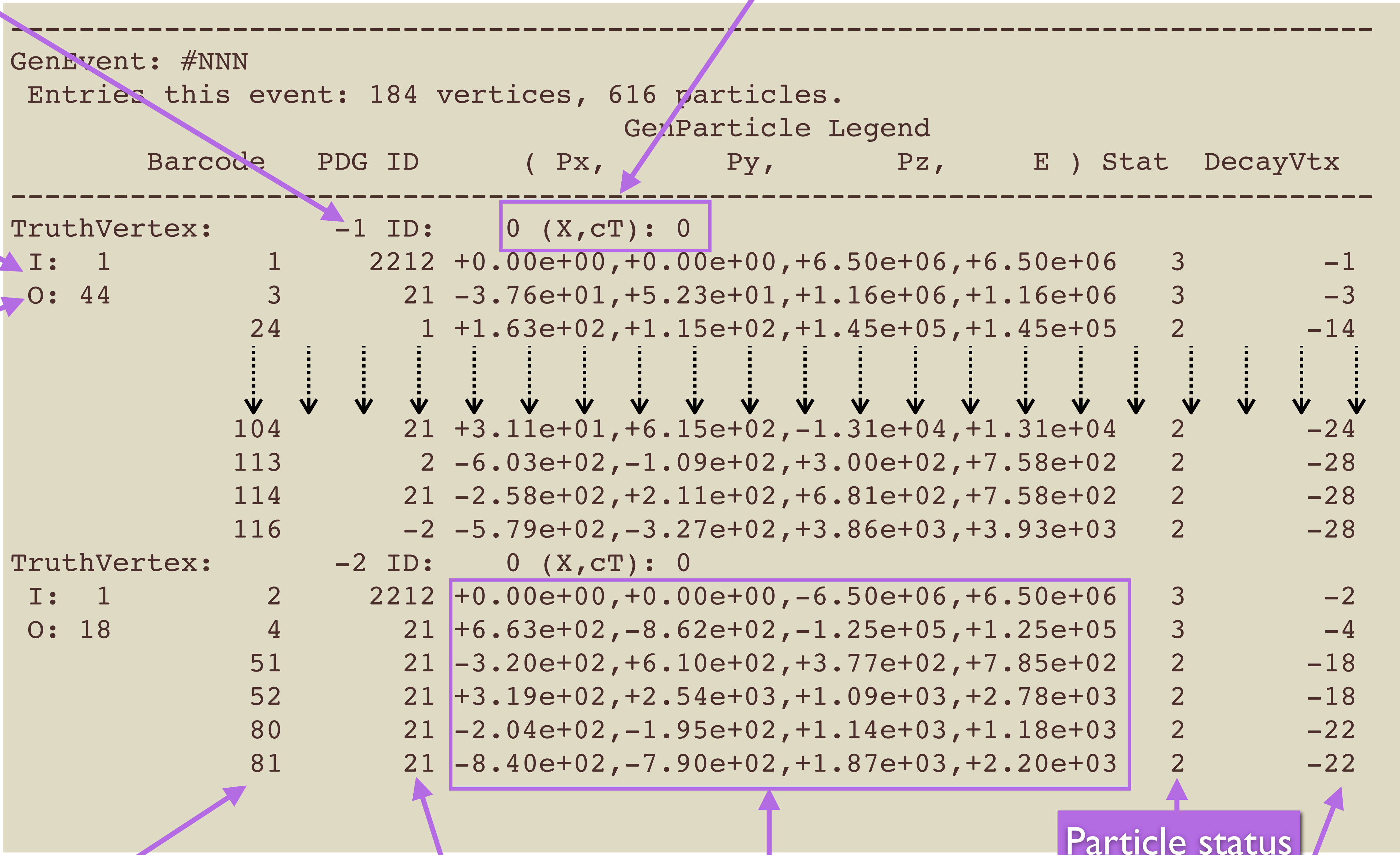
Kinematics

```
bool hasDecayVtx = particle->hasDecayVtx();
bool hasProdVtx = particle->hasProdVtx();
const xAOD::TruthVertex* prodVtx = particle->prodVtx();
const xAOD::TruthVertex* decayVtx = particle->decayVtx();
```

Access to production and decay vertices



HepMC ASCI form



```
--------------------------------------------------------------------------------
GenEvent: #NNN
 Entries this event: 184 vertices, 616 particles.
                                    GenParticle Legend
        Barcode   PDG ID      ( Px,       Py,       Pz,     E ) Stat  DecayVtx
--------------------------------------------------------------------------------
TruthVertex:       -1 ID:    0 (X,cT): 0
 I:   1        1     2212 +0.00e+00,+0.00e+00,+6.50e+06,+6.50e+06   3         -1
 O: 44         3       21 -3.76e+01,+5.23e+01,+1.16e+06,+1.16e+06   3         -3
              24        1 +1.63e+02,+1.15e+02,+1.45e+05,+1.45e+05   2        -14

             104       21 +3.11e+01,+6.15e+02,-1.31e+04,+1.31e+04   2        -24
             113        2 -6.03e+02,-1.09e+02,+3.00e+02,+7.58e+02   2        -28
             114       21 -2.58e+02,+2.11e+02,+6.81e+02,+7.58e+02   2        -28
             116       -2 -5.79e+02,-3.27e+02,+3.86e+03,+3.93e+03   2        -28
TruthVertex:       -2 ID:    0 (X,cT): 0
 I:   1        2     2212 +0.00e+00,+0.00e+00,-6.50e+06,+6.50e+06   3         -2
 O: 18         4       21 +6.63e+02,-8.62e+02,-1.25e+05,+1.25e+05   3         -4
              51       21 -3.20e+02,+6.10e+02,+3.77e+02,+7.85e+02   2        -18
              52       21 +3.19e+02,+2.54e+03,+1.09e+03,+2.78e+03   2        -18
              80       21 -2.04e+02,-1.95e+02,+1.14e+03,+1.18e+03   2        -22
              81       21 -8.40e+02,-7.90e+02,+1.87e+03,+2.20e+03   2        -22
```

Vertex barcode (-ve)

Vertex spatial/temporal information

Incoming

Outgoing

Particle barcode (+ve)

PDG ID

Particle kinematics

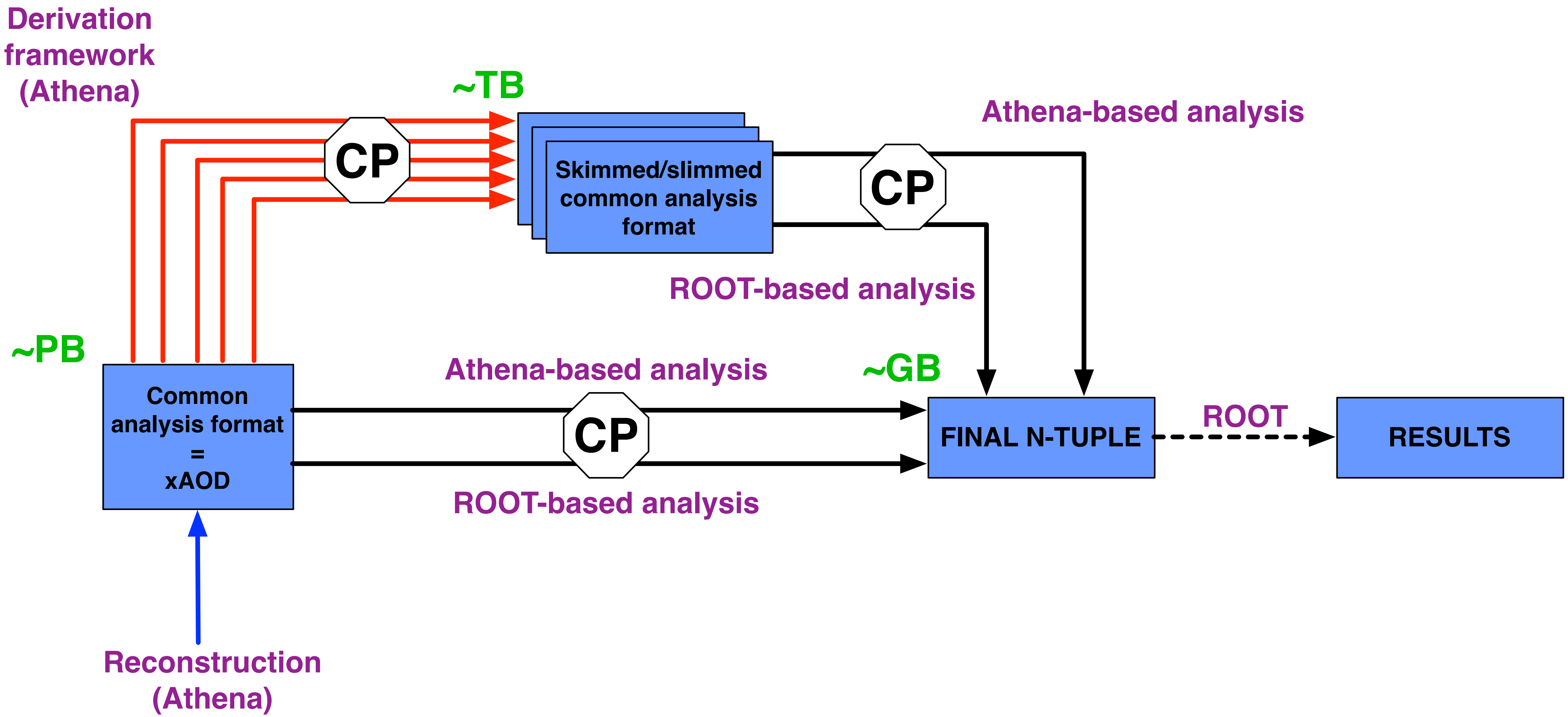
Particle status

Barcode of decay vertex for this particle

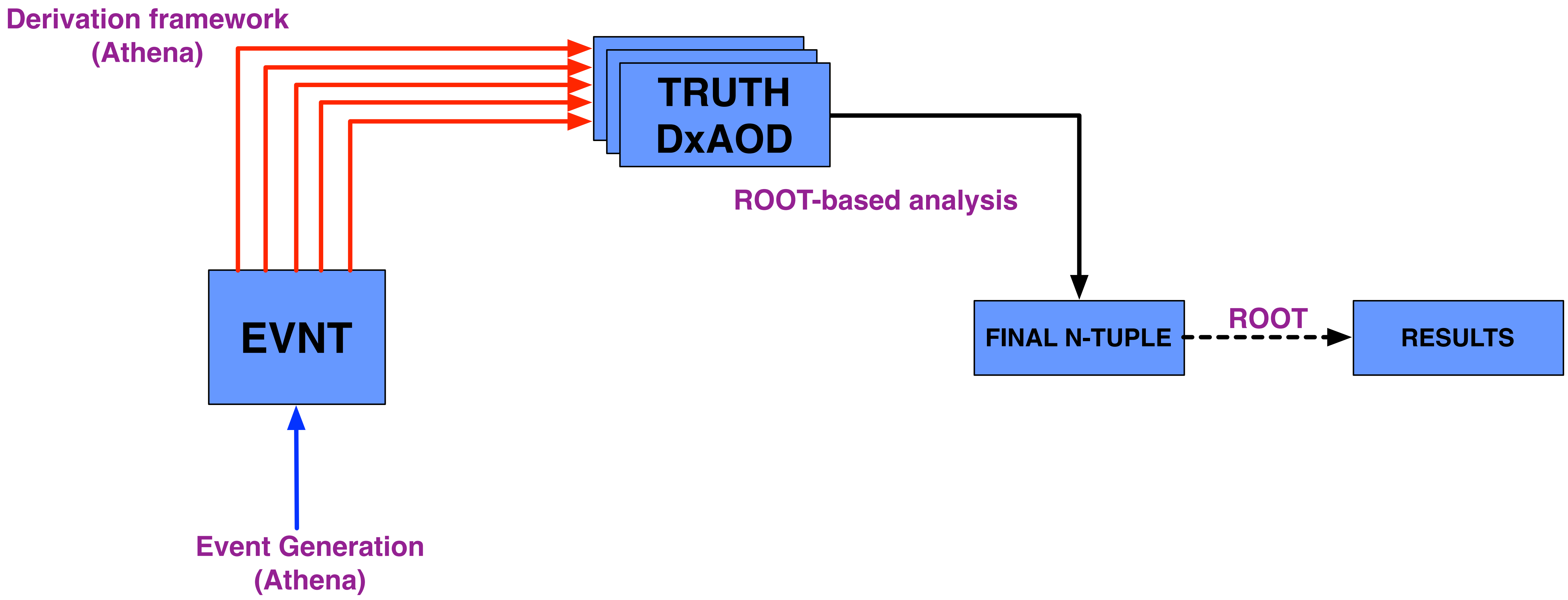
Pile-up truth

- Pile-up truth events are stored in a different event class and container
  - xAOD::TruthPileupEvent, TruthPileupEventContainer
- However, pile-up particles and vertices are in the same container as the signal!
  - This is necessary due to the making of links between truth particles and reconstructed objects
  - But it means that, if you loop over the particles/vertices directly, you will get signal and pile-up together
    - This is why you should always navigate to the particles via the event
- Most MC samples (MC15) have pile-up truth, but the particle record is collapsed to a single particle (a "geantino") and vertex apart from for some special samples
  - There are sometimes also truth pileup jets in there

ATLAS analysis model for reconstructed data/MC



ATLAS analysis model for MC truth



DxAOD truth formats

- We use the derivation framework to produce xAOD truth files directly from EVNT
  - They are referred to as "truth DxAODs"
- Like all xAOD files, they are ROOT-readable and work with the usual ATLAS analysis tools
- They can be made as part of central production on request to the DPD production group, or privately
- There are five different formats currently available, of which three are in regular use for physics

What is in the DxAOD truth formats?

- The formats contain different information/objects at different levels of detail, and may have some or all of the following
  - Full or thinned truth record
    - Thinned = unwanted particles or vertices removed to save space
    - Thinning usually leads to loss of *graph completeness*, e.g. ability to navigate from parent to child
  - Dedicated particle containers, e.g. for truth electrons, truth muons etc
  - Classification and summary information
  - Truth jet and MET containers built by ATLAS reconstruction packages

Truth DxAOD summary table

Format	Size/event (KB) ttbar events	Full truth record?	Partial truth record?	Hard process treatment	Truth classification / summary info	Dedicated truth particle containers	Truth jets/ MET	For analysis use?
TRUTH0	25	✓						✓
TRUTH1	7.8		✓		✓	✓	✓	✓
TRUTH2	40		✓	✓			✓	
TRUTH3	1.6				✓	✓	✓	✓
TRUTH4	40	✓					✓	

Doing truth analysis: what format do we use?

- TRUTH0
  - use it for generator validation and applications where access to the full event record is absolutely essential
  - very big, no summary/classification info or jets/MET
- TRUTH1
  - currently the main truth format for ATLAS analysis; contains the important parts of the truth record (maintaining graph completeness), dedicated containers, jets/MET, some classification
  - still big
- TRUTH3
  - we hope that this format will become the main truth format in ATLAS
  - it will be introduced during this tutorial - you'll be the amongst the first to use it so your comments/suggestions will be essential
  - Ben will cover it in detail in the next talk

A bit more about TRUTH I

- Main truth record
  - B-hadrons
  - Hadrons from tau decays
  - Non-SM particles and their decay products
  - W, Z, H, γ and their decay products
  - Top quarks and their decay products
  - First 10 partons in the event record
  - The ancestors of all of the above

- Extra containers
  - Muons, electrons, photons, neutrinos
  - Truth jets and MET

- Origin type and classification

What’s all this classification about?

- “Classification” of a truth particle is a pair of numbers which tells you
  - what class of particle it is (type) - NOT the same as the PDG ID
  - where it came from (origin)
- Calculated via a tool called the MCTruthClassifier
  - List of codes: http://acode-browser.usatlas.bnl.gov/lxr/source/atlas/PhysicsAnalysis/MCTruthClassifier/MCTruthClassifier/MCTruthClassifierDefs.h
- e.g. a particle with `origin==22` and `type==6` would be an isolated muon from a SUSY decay
- This number is NOT part of HepMC or the main xAOD truth (since you don’t normally need them if you have the full truth record)
- But if you have removed large parts of the truth record, the classification becomes very important
- Classifications are added automatically to many of the Truth DxAODs as extra variables (“decorations”) to the particles

Relationship between truth DxAODs and MC DxAODs

- DxAODs from reconstructed MC also contains truth information
- It is made by the same tools as the truth DxAODs but it is often very different. Why?
  - The source is different: truth DxAODs are made from EVNT which contains no simulation information, whereas MC DxAODs include particles produced by simulation (Geant)
  - Physics groups have full control over what they write into their own group formats, and they vary greatly as to what they need
- However, some of the TRUTH1 containers are being written into the MC DxAODs, at the discretion of the physics groups
  - We hope to encourage them to move to TRUTH3 containers once they are happy with them

More information on xAOD truth

- xAOD classes:
  - https://svnweb.cern.ch/cern/wsvn/atlasoff/Event/xAOD/xAODTruth/? : main classes
  - https://svnweb.cern.ch/cern/wsvn/atlasoff/Event/xAOD/xAODTruthCnv/? : converter from HepMC and an ASCI dumper
- HepMC manual (physics meanings are identical in xAOD, which is just a re-organisation of the same information)
  - http://lcgapp.cern.ch/project/simu/HepMC/
- Truth DxAOD documentation
  - https://twiki.cern.ch/twiki/bin/view/AtlasProtected/TruthDAOD
- Python definitions of the truth DxAODs
  - https://svnweb.cern.ch/cern/wsvn/atlasoff/PhysicsAnalysis/DerivationFramework/DerivationFrameworkMCTruth/trunk/share

In this tutorial…

- More information about TRUTH3 from Ben in the next talk
- Hands-on session
  - Make TRUTH3 from EVNT
  - Run an analysis on TRUTH3
  - Experiment with the analysis code and play with the new format
  - Make TRUTH0 from EVNT and dump contents to ASCI