Table of Contents

Learning Pythia6 output for photon+jets

Links

SWGuideDataFormatTable	**Data Format Tables for the Offline Guide**

Contents: Show Hide

SWGuideDataFormatGeneratorInterface	**Data Formats in GeneratorInterface**

Contents: Show Hide

WorkBookGenParticleCandidate	**Generator event format in AOD**

Contents: Show Hide

SWGuideEventGeneration	**Event Generation Offline Guide**

Contents: Show Hide

SWGuidePythia6Interface	**Pythia6 Interface to CMSSW**

Contents: Show Hide

SWGuidePhysicsTools	**Physics Analysis Tools Offline Guide**

Contents: Show Hide

SWGuideCandidateModules	**Common Candidate Modules**

Contents: Show Hide

Products

All GEN-SIM-RECO products

A GEN-SIM-RECO file

The following dataset of Pythya6 origin was chosen for tests:

- /RelValPhotonJets_Pt_10/CMSSW_3_7_0-START37_V4-v1/GEN-SIM-RECO
  - 9000 evs in 5 files: More... Close

```
replace PoolSource.fileNames = {
        '/store/relval/CMSSW_3_7_0/RelValPhotonJets_Pt_10/GEN-SIM-RECO/START37_V4-v1/0026/028B3AC
        '/store/relval/CMSSW_3_7_0/RelValPhotonJets_Pt_10/GEN-SIM-RECO/START37_V4-v1/0024/EA7C919
        '/store/relval/CMSSW_3_7_0/RelValPhotonJets_Pt_10/GEN-SIM-RECO/START37_V4-v1/0024/C450BF0
        '/store/relval/CMSSW_3_7_0/RelValPhotonJets_Pt_10/GEN-SIM-RECO/START37_V4-v1/0024/BE5CCD0
        '/store/relval/CMSSW_3_7_0/RelValPhotonJets_Pt_10/GEN-SIM-RECO/START37_V4-v1/0024/4E27E97
}
```

-
  - Its parent dataset /RelValPhotonJets_Pt_10/CMSSW_3_7_0-START37_V4-v1/GEN-SIM-DIGI-RAW-HLTDEBUG had the following pythia parameters in the config file: More... Close

```
process.generator = cms.EDFilter("Pythia6GeneratorFilter",
    pythiaPylistVerbosity = cms.untracked.int32(0),
    filterEfficiency = cms.untracked.double(1.0),
```

```
    pythiaHepMCVerbosity = cms.untracked.bool(False),
    comEnergy = cms.double(7000.0),
    maxEventsToPrint = cms.untracked.int32(0),
    PythiaParameters = cms.PSet(
        pythiaUESettings = cms.vstring('MSTJ(11)=3     ! Choice of the fragmentation function',
            'MSTJ(22)=2     ! Decay those unstable particles',
            'PARJ(71)=10 .  ! for which ctau  10 mm',
            'MSTP(2)=1      ! which order running alphaS',
            'MSTP(33)=0     ! no K factors in hard cross sections',
            'MSTP(51)=10042 ! structure function chosen (external PDF CTEQ6L1)',
            'MSTP(52)=2     ! work with LHAPDF',
            'MSTP(81)=1     ! multiple parton interactions 1 is Pythia default',
            'MSTP(82)=4     ! Defines the multi-parton model',
            'MSTU(21)=1     ! Check on possible errors during program execution',
            'PARP(82)=1.8387   ! pt cutoff for multiparton interactions',
            'PARP(89)=1960. ! sqrts for which PARP82 is set',
            'PARP(83)=0.5   ! Multiple interactions: matter distrbn parameter',
            'PARP(84)=0.4   ! Multiple interactions: matter distribution parameter',
            'PARP(90)=0.16  ! Multiple interactions: rescaling power',
            'PARP(67)=2.5    ! amount of initial-state radiation',
            'PARP(85)=1.0  ! gluon prod. mechanism in MI',
            'PARP(86)=1.0  ! gluon prod. mechanism in MI',
            'PARP(62)=1.25   ! ',
            'PARP(64)=0.2    ! ',
            'MSTP(91)=1      !',
            'PARP(91)=2.1   ! kt distribution',
            'PARP(93)=15.0  ! '),
        processParameters = cms.vstring('MSEL=10              ! Pythia Photon+Jet processes',
            'CKIN(3)=10.          ! minimum pt hat for hard interactions'),
        parameterSets = cms.vstring('pythiaUESettings',
            'processParameters')
    )
)
```

List of products

The full list of products in the above dataset was obtained by

- copying one event to a separate file (copy job)
- applying the **`edmDumpEventContent`** utility to the one-event file
  - ♦ **the output**

Another type of the list was obtained by running the **`EventContentAnalyzer`**

- **the list**

Product sizes

The *product sizes* were obtained with the **`edmEventSize`** utility (SWGuideEdmEventSize , WorkBook reference)

```
edmEventSize -v -o out rfio:///castor/cern.ch/cms//store/relval/CMSSW_3_7_0/RelValPhotonJets_Pt_1
```

- **the output file `out`**

Note: the two reported numbers for a products are the average *plain and compressed sizes (in bytes).*

The sum of compressed sizes over all products, 355757 bytes, can be compared to the average record size = (file length) / nEvents = 746345755/2000 = 373173 . => The overhead for the event structure seems to be

about 5% .

Generator products

A table from SWGuideDataFormatGeneratorInterface with links corrected:

InputTag/Module (Instance name)	Container	Description
GeneratorInterface collections (in RECOSIM and AODSIM)		
generator	GenEventInfoProduct	General characteristics of a generated event.
generator	GenRunInfoProduct	Run-specific parameters that define event generation, such as cross-sections, etc.
GeneratorInterface collections (in RECOSIM only)		
generator	edm::HepMCProduct	A tree of final-state particles that form a generated event.
generator	GenEventInfoProduct	General characteristics of a generated event.
generator	GenRunInfoProduct	Run-specific parameters that define event generation, such as cross-sections, etc.

Examples of accessing `edm::HepMCProduct` , `GenEventInfoProduct` and `GenRunInfoProduct`

Simple examples are given in SWGuideDataFormatGeneratorInterface in the section How_to_use_the_table

Example: fetching

The example is taken from

- CMSSW/GeneratorInterface/Pythia6Interface/test/HZZ4muAnalyzer.cc

```
//. . .
#include "SimDataFormats/GeneratorProducts/interface/GenEventInfoProduct.h"
//. . .
void HZZ4muAnalyzer::analyze( const Event& e, const EventSetup& )
{
  Handle< GenEventInfoProduct > GenInfoHandle;
  e.getByLabel( "generator", GenInfoHandle );

  double pthat = ( GenInfoHandle->hasBinningValues() ?
                   (GenInfoHandle->binningValues())[0] : 0.0);
//. . .
```

`BuildFile` : taking care of

While accessing a generator product, do not forget to add a coresponding line to the **`BuildFile`** . E.g., in case of `GenEventInfoProduct`, the following line has to be added:

```
<use name=SimDataFormats/GeneratorProducts>
```

Otherwise an unclear **fatal** linking **error** message will be issued by `scramv1 b` :
.../libYOUR-PACKAGE-NAME.so: undefined reference to `typeinfo for GenEventInfoProduct'
collect2: ld returned 1 exit status

reco::

Definition

In the AOD event content, the edm::HepMCProduct format of the generated event is **replaced** by the "lighter" (about 2 times) record/product of type

```
reco::GenParticleCollection
```

The reco::GenParticleCollection is **a typedef** for

```
std::vector < reco::GenParticle >
```

For an info on the ***reco::GenParticle*** , see below

GenParticleProducer

The *reco::GenParticleCollection* is **produced** from the *reco::HepMCProduct* by the **GenParticleProducer**

- CMSSW/PhysicsTools/HepMCCandAlgos/plugins/GenParticleProducer.cc

configured with the

- .../python/genParticles_cfi.py .

There are short descriptions of the module in WorkBook and SWGuide (? both claim that the configuration file is located in the *.../data/* directory which seems to be empty actually).

GenParticleCollection in the edmDumpEventContent output

The *reco::GenParticleCollection* is reported in the above Full List of Pruducts as

```
vector<reco::GenParticle>   "genParticles"    ""    "HLT."
```

There is another product with the label ***genParticles*** also originating from the GenParticleProducer :

```
vector<int>                 "genParticles"    ""    "HLT."
```

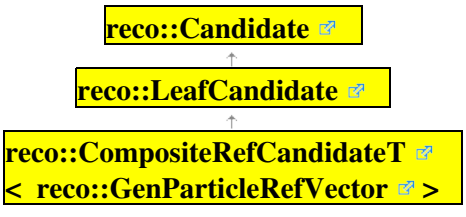
whose content is not clear .

reco::

- Class Reference
- header file:

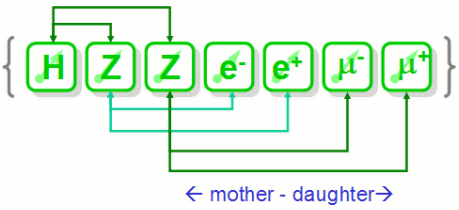
```
#include <DataFormats/HepMCCandidate/interface/GenParticle.h>
```

- inheritance diagram:



reco::GenParticle

- The *generator particles* may contain *mother* and/or *daughter* links to particles in the same collection (picture from here):



`pdgID()` and `status()` member functions of reco::GenParticle

The material is from the WorkBook .

- **`pdgId()`** : a PDG identifier (`pdg_id()` in `HepMC::GenParticle`)
- **`status()`** : a status code (`status()` in `HepMC::GenParticle`). Standard status codes are described in HepMC manual .

  - Status codes have the following convention in **Pythia**:

Value	Meaning
0	null entry
1	existing entry – not decayed or fragmented, represents the final state as given by the generator
2	decayed or fragmented entry (i.e. decayed particle or parton produced in shower.)
3	identifes the "hard part" of the interaction, i.e. the partons that are used in the matrix element calculation, including immediate decays of resonances. (documentation entry, defined separately from the event history. *"This includes the two incoming colliding particles and partons produced in hard interaction."* [*])
4-–10	undefined, reserved for future standards
11-–200	at the disposal of each model builder equivalent to a null line
201-...–	at the disposal of the user, in particular for event tracking in the detector

  - Other generators may have more complex conventions. See, for instance:
    - **Herwig** particle codes

Example of accessing a GenParticleCollection

The example is from the WorkBook .

```
#include "DataFormats/HepMCCandidate/interface/GenParticle.h"

using namespace reco; // to access types reco::GenParticleCollection ,
                      //                 reco::Candidate

void MyModule::analyze(const edm::Event & event, ...) {

   // get the handle
   Handle<GenParticleCollection> genParticles;
   event.getByLabel("genParticles", genParticles);

   // loop over particles
   for(size_t i = 0; i < genParticles->size(); ++ i) {

     // the reference p to the i-th particle:
     const GenParticle & p = (*genParticles)[i];
```

```
      // get pdgId:
      int id = p.pdgId();
      // get status:
      int st = p.status();

      // get pointer to mother (reco::Candidate type!):
      const Candidate * mom = p.mother();

      // get pt, eta, phi, mass:
      double pt = p.pt(), eta = p.eta(), phi = p.phi(), mass = p.mass();

      // get vertex components:
      double vx = p.vx(), vy = p.vy(), vz = p.vz();

      // get charge:
      int charge = p.charge();

      // get no. of daughters:
      int n = p.numberOfDaughters();

      // loop over daughters:
      for(size_t j = 0; j < n; ++ j) {

        // get pointer d to a daughter  (reco::Candidate type!):
        const Candidate * d = p.daughter( j );

        // get daughter's id:
        int dauId = d->pdgId();
        // . . .
      }
      // . . .
    }
}
```

Notes:

- The **`reco::GenParticle`** ***member functions*** mentioned in the above example, are defined as *purely virtual* in the base class **`reco::Candidate`** , then actually implemented in the class **`reco::LeafCandidate`** (see InheritanceDiagram).
- There are ***many more*** inherited ***member functions*** (see *links* from the InheritanceDiagram).
- Check **typedef's** in the ***reco:: namespace***:
  - in the reco::Candidate Class Reference (corresponds to the `DataFormats/Candidate/interface/Candidate.h`),
  - in the CandidateFwd.h File Reference ,
  - and (?) similar references in the inheritance chain.

Using the plugin as a slave class in an EDAnalyzer

The ParticleListDrawer module produces the output which is similar to the one from the Pythia routine PYLIST. The plugin usage is described in SWGuide .

One can also use it directly from c++ of an EDAnalyzer. Here is an example.

1. Add into your **`...cfg.py`** :
   - the **load** statement:

```
process.load("SimGeneral.HepPDTESSource.pythiapdt_cfi")
```

- ♦ ◊ The file `SimGeneral/HepPDTESSource/python/pythiapdt_cfi.py` gets loaded containing

```
HepPDTESSource = cms.ESSource(
    "HepPDTESSource",
    pdtFileName = cms.FileInPath(
        'SimGeneral/HepPDTESSource/data/pythiaparticle.tbl'
    )
)
```

- ♦ the `ParticleListDrawerConfig` *parameter set* to the configuration block of your Analyzer module. This parameter set will be used later on to config a *ParticleListDrawer object* in your code:

```
. . .
process.aName = cms.EDAnalyzer(
    'YourAnalyzerName',
    . . . ,
    #
    # parameter set for a ParticleListDrawer object:
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    ParticleListDrawerConfig = cms.untracked.PSet(
        #      +-------------------------------------+
        #         parameter                  default
        #      +-------------------------------------+
        #         src                        "src"
        #         maxEventsToPrint           1
        #         printVertex                False
        #         printOnlyHardInteraction   False
        #         useMessageLogger           False
        #      +-------------------------------------+
        src                     = cms.InputTag        ("genParticles"),
        maxEventsToPrint        = cms.untracked.int32 (-1)
        #   default settings are commented out:
        #printVertex              = cms.untracked.bool (False)
        #printOnlyHardInteraction = cms.untracked.bool (False)
        #useMessageLogger         = cms.untracked.bool (False)
    )
)
. . .
process.p = cms.Path(process.aName)
```

The name `ParticleListDrawerConfig` is chosen arbitrarily. Also the configuration options may be different.

1. In the **`YourAnalizer.cc`**:
   - ♦ add the following **#include** e.g. as the first `include` :

```
//include other plugins:
#include "PhysicsTools/HepMCCandAlgos/plugins/ParticleListDrawer.cc"
```

- ♦ add a **data member** *in the class prototype*:

```
public:   // or private:
  . . .
  ParticleListDrawer * partListDr;
```

- ♦ **instantiate** the object *in the constructor*:

```
YourAnalyzer::
YourAnalyzer ( const edm::ParameterSet & iConfig)
{
  partListDr = new ParticleListDrawer
    ( iConfig.getUntrackedParameterSet ("ParticleListDrawerConfig") )  ;
  . . .
```

- ♦ finally, **print the list** whenever you want *in the `analyze' function*:

```
void
YourAnalyzer::
analyze ( const edm::Event      & iEvent,
  const edm::EventSetup & iSetup)
{
  . . .
  if (. . .) partListDr -> analyze( iEvent, iSetup) ;
  . . .
```

1. Add to your **`BuildFile`** :

```
. . .
<use name=PhysicsTools/HepMCCandAlgos>
<use name=DataFormats/Candidate>
. . .
<export>
. . .
   <use name=PhysicsTools/HepMCCandAlgos>
   <use name=DataFormats/Candidate>
</export>
```

as a slave

The use case for the ParticleTreeDrawer described in in SWGuide would be ***identical*** (up to a different configuration) to the ParticleListDrawer above ***if*** the **`analyze`** function of the ParticleTreeDrawer ***would not be declared*** **private**.

The way out is

- to copy the `PhysicsTools/HepMCCandAlgos/plugins/ParticleTreeDrawer.cc` to our package `src` directory,
- change **private:** to **public:** for the **`analyze`** function
- split the file into a **`.h`** and **`.cc`** (or may be (?) just rename the **`.cc`** into **`.h`**)
- and then proceed similarily to the ParticleListDrawer case

The **include** line should now be

```
//include other plugins:
//                  In the standard ParticleTreeDrawer, `analyse' is private
//                  => use a local version where it has been  made public
//#include "PhysicsTools/HepMCCandAlgos/plugins/ParticleTreeDrawer.cc"
#include "ParticleTreeDrawer.h"
```

And the configuration for the ParticleTreeDrawer has to be different:

```
. .
process.aName = cms.EDAnalyzer(
    'YourAnalyzerName',
    . . . ,
```

```
    #
    # parameter set for a ParticleTreeDrawer object:
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    ParticleTreeDrawerConfig = cms.untracked.PSet(
        #      +-----------------------------------------------------+
        #         parameter       default
        #      +-----------------------------------------------------+
        #         src             "src"
        #         printP4         False
        #         printPtEtaPhi   False
        #         printVertex     False
        #         printStatus     False
        #         printIndex      False
        #         status          empty list of statuses (means: print all)
        #      +-----------------------------------------------------+
        src            =  cms.InputTag          ("genParticles"),
        printIndex     =  cms.untracked.bool    (True),
        status         =  cms.untracked.vint32  (3)
        #    default settings are commented out:
        #printP4       =  cms.untracked.bool    (False),
        #printPtEtaPhi =  cms.untracked.bool    (False),
        #printVertex   =  cms.untracked.bool    (False),
        #printStatus   =  cms.untracked.bool    (True),
    )
)
. . .
process.p = cms.Path(process.aName)
```

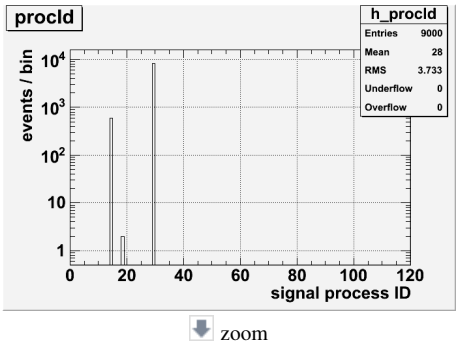
In the above example only the particles from the *hard part* of the interaction (status code = 3) are printed out. It is noteworthy that a *full* listing is too long and indigestible.

PYLIST's and Parton Flow sketches for ISUB=29,14,18

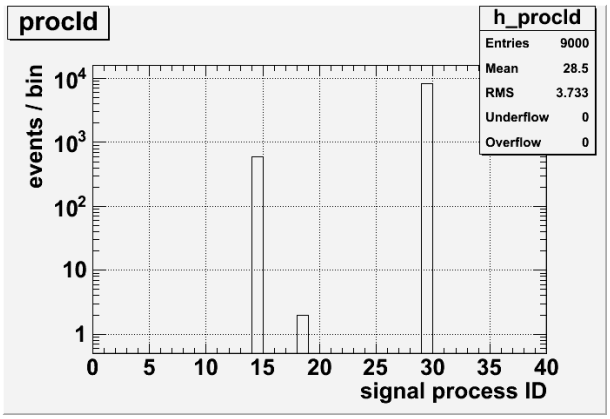
There are three subprocesses occuring in the file above (the subrocess number comes from the `GenEventInfoProduct::signalProcessID()` function, More... Close

```
// get handle to the GenEventInfoProduct
  edm::Handle < GenEventInfoProduct > genEvInfo ;
  iEvent.getByLabel (genEventInfoLabel_ , genEvInfo) ;
  // get pythia subprocess:
  int pyIsub =  (int) genEvInfo -> signalProcessID () ;
```

):



zoom



gif

Thus, one has:

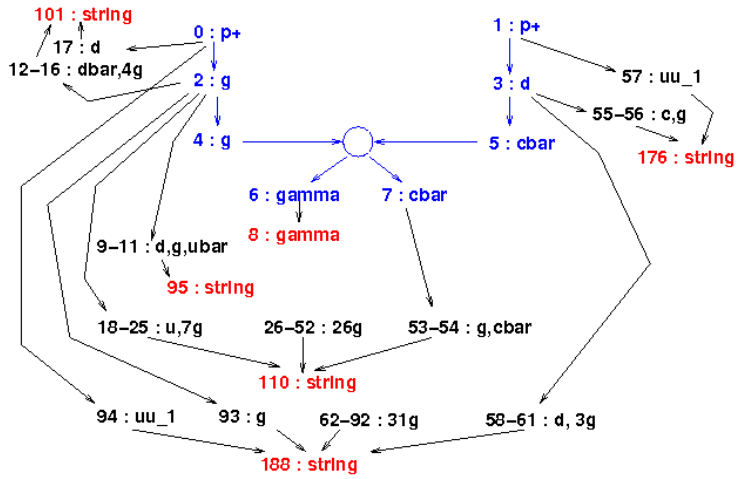
subprocess		fraction
29	$q_i g \to q_i \gamma$	93 %
14	$q_i \bar{q}_i \to g\gamma$	7 %
18	$f_i \bar{f}_i \to \gamma\gamma$	.02 %
possible but not present (too low statistics?):		
114	$gg \to \gamma\gamma$	0
115	$gg \to g\gamma$	0

Some PYLIST's (the outputs of the ParticleListDrawer actually) for the available subprocesses can be found in the following text files:

- ISUB = 29 (10 evs with pthat > 15 GeV)
- ISUB = 14 (10 evs with pthat > 15 GeV)
- ISUB = 18 (2 evs with pthat > 10 GeV)

One can also look at the **parton-flow sketches** made for the first few events from each of the above files: More... Close

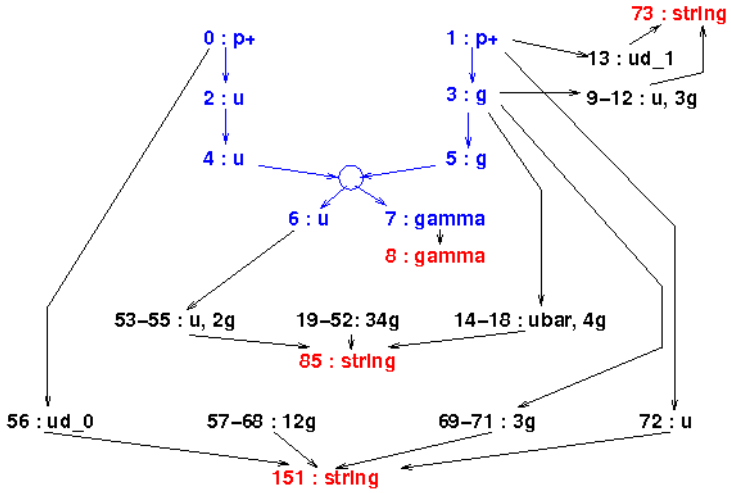
ISUB = 29: More... Close **Event 1:** More... Close



Partonic flow in a pythia6 event, subprocess 29

blue : "hard interaction"

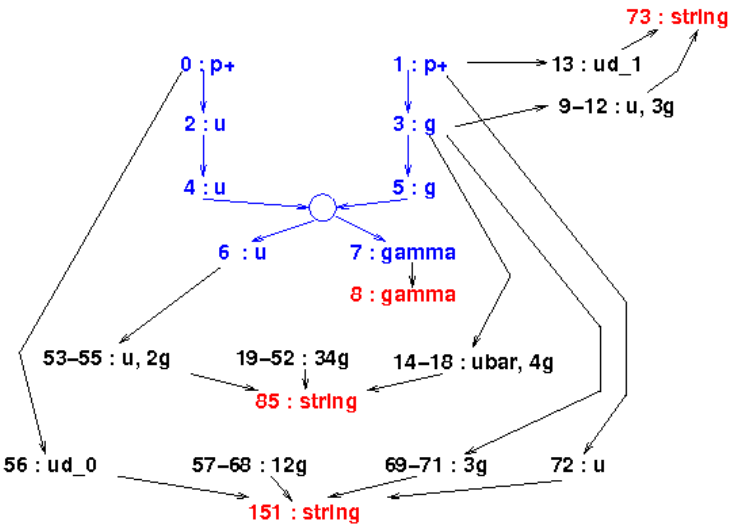
Event 2: More... Close



Partonic flow in a pythia6 event, subprocess 29

blue: "hard interaction"

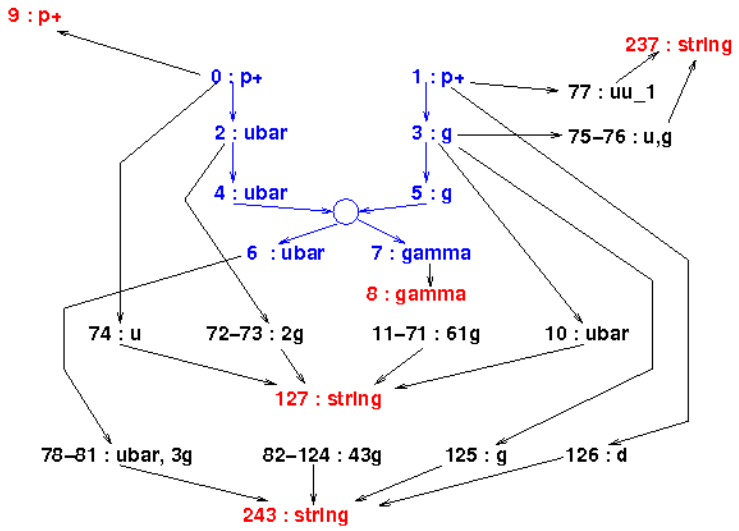
Event 3: More... Close



Partonic flow for a pythia6 event, subprocess 29

blue : "hard interaction"

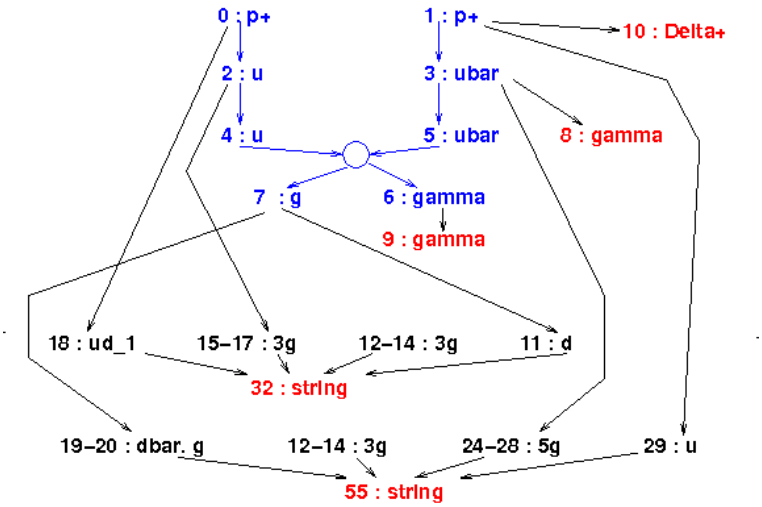
Event 4: More... Close



Partonic flow for a pythia6 event, subprocess 29

blue : "hard interaction"

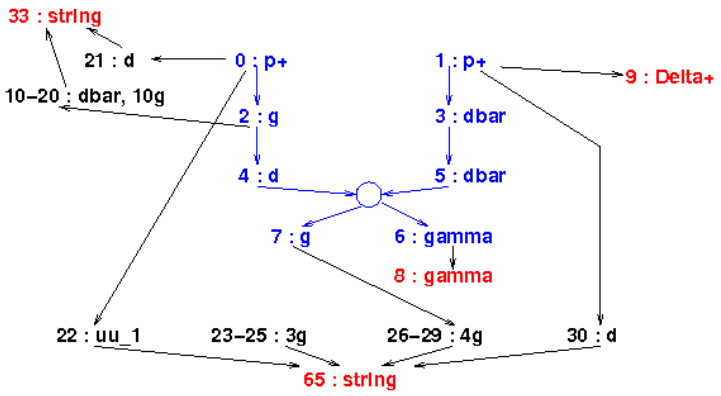
ISUB = 14: More... Close **Event 1:** More... Close



Partonic flow for a pythia6 event, subprocess 14

blue : "hard interaction"

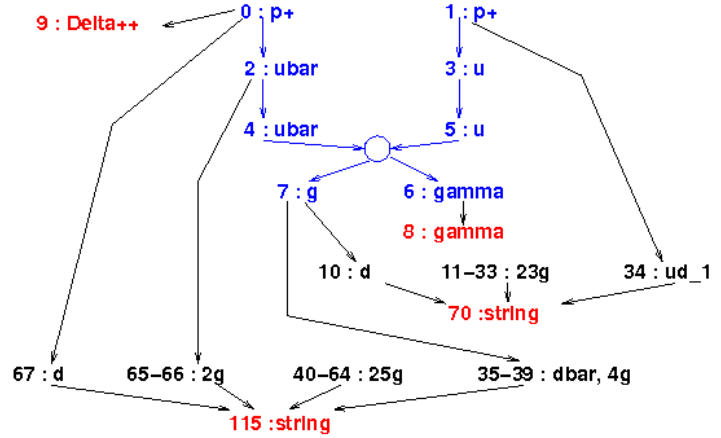
Event 2: More... Close



Partonic flow for a pythia6 event, subprocess 14

blue : "hard interaction"

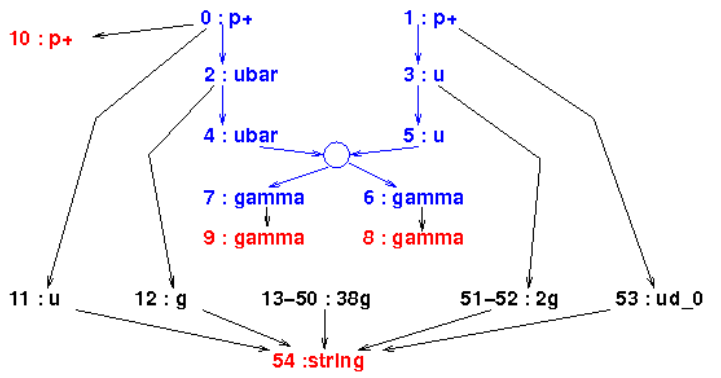
Event 3: More... Close



Partonic flow for a pythia6 event, subprocess 14

blue : "hard interaction"

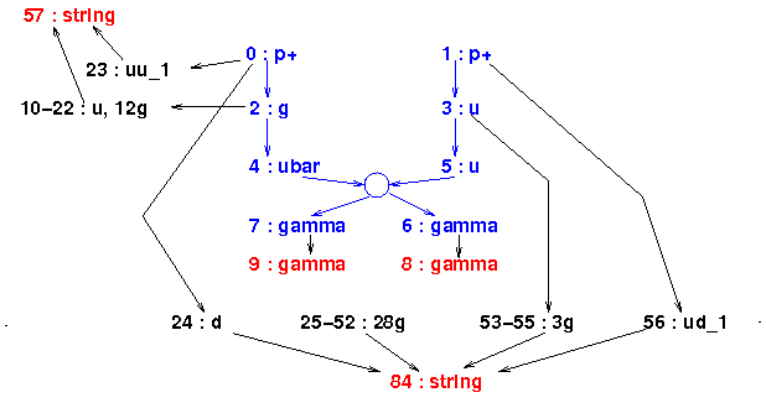
ISUB = 18: More... Close **Event 1:** More... Close



Partonic flow for a pythia6 event, subprocess 18

blue : "hard interaction"

Event 2: More... Close



Partonic flow for a pythia6 event, subprocess 18

blue : "hard interaction"

Original files: **gif:** More... Close ISUB=29: 1 , 2 , 3 , 4 , ISUB=14: 1 , 2 , 3 , ISUB=18: 1 , 2 , **fig:** More... Close ISUB=29: 1 , 2 , 3 , 4 , ISUB=14: 1 , 2 , 3 , ISUB=18: 1 , 2

This topic: Main > AVFedotovLogA008
Topic revision: r33 - 2013-11-01 - AlexanderFedotov