News on Rho

PANDA Collaboration Meeting , Bochum

11. 9. 2013

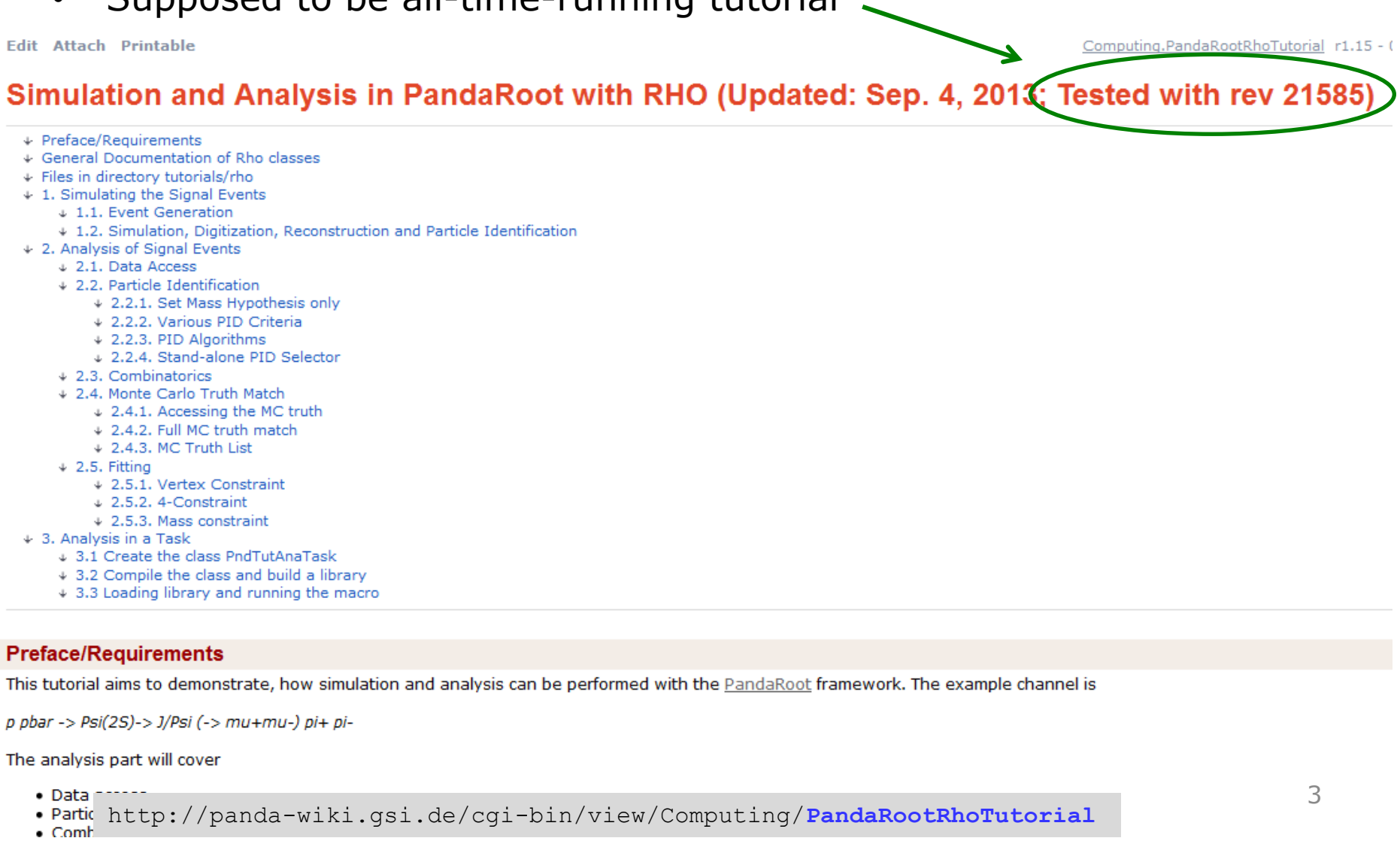
K. Götzen, GSI

Major Changes (by Ralf Kliemt)

- Rename all Rho classes to Rho*
- RhoCandidate handled fully with pointers
- Unclutter Rho classes → delete obsolete/unnecessary stuff
- Remove unnecessary virtual or semivirtual layers
- Structure fitter interfaces
- Put data objects from pid to pnddata & rename
- Reorganize PndAnalysis
- New Tutorial + Rho class Docu in PANDA Wiki

New Tutorial – August 2013

- Available in PANDA-Wiki
- Supposed to be all-time-running tutorial



Edit Attach Printable

Computing.PandaRootRhoTutorial r1.15 - (

Simulation and Analysis in PandaRoot with RHO (Updated: Sep. 4, 2013; Tested with rev 21585)

Preface/Requirements

This tutorial aims to demonstrate, how simulation and analysis can be performed with the PandaRoot framework. The example channel is

p pbar -> Psi(2S)-> J/Psi (-> mu+mu-) pi+ pi-

The analysis part will cover

- Data
- Partic
- Comb

http://panda-wiki.gsi.de/cgi-bin/view/Computing/**PandaRootRhoTutorial**

… and additional Documentation!

Edit Attach Printable Computing.PandaRootAnalysisJuly13

Documentation of Rho Classes

Index

Here you can find information about:

- **PndAnalysis** - Interface - Lists and Keys - PID Algo Names - MC Truth Match
- **RhoCandidate** - Interface
- **RhoCandList** - Interface
- **Particle Selectors** - Kinematic Selectors - PID Selectors
- **Fitters** - Vertex Fitting - 4C Fitting - Kinematic Fitting

A tutorial for Rho with an example simulation and analysis can be found on this Wiki page.

http://panda-wiki.gsi.de/cgi-bin/view/Computing/**PandaRootAnalysisJuly13**

PndAnalysis: Data Access/PID

- Data access via PndAnalysis

```
FairRunAna *fRun = new FairRunAna();
fRun->SetInputFile("pid_complete.root");

PndAnalysis *ana = new PndAnalysis();
RhoCandList eplus, eminus, muplus, muminus, mct;
TString myPidAlgosElectron = "PidAlgoEmcBayes;PidAlgoDrc"
TString myPidAlgosMuon     = "PidAlgoMdtHardCuts"
...
while ( ana->GetEvent() )
{
  ana->FillList( eplus,   "ElectronTightPlus",  myPidAlgosElectron );
  ana->FillList( eminus,  "ElectronTightMinus", myPidAlgosElectron );
  ana->FillList( muplus,  "MuonTightPlus",      myPidAlgosMuon );
  ana->FillList( muminus, "MuonTightMinus",     myPidAlgosMuon );
  ana->FillList( mct,     "McTruth" );
  ...
}
```

With standard pid macros only pid file needed as input, also including MC info now!

- Features:
  - Simple access to reco candidates and McTruth objects
  - Various PID algorithms directly accessible

Particle Identification Concept

- PID probabilities are computed by various algorithms *PidAlgoEmcBayes*, *PidAlgoDrc*, *PidAlgoMvd*, ...

- PndAnaPidCombiner
  - combines *on demand* probabilities from various algo's by computing product of all P_i *(i=algo's)*
  - copy probabilities to RhoCandidate/RhoCandList

- PndAnaPidSelector / RhoSimple...Selector
  - selects particles based on these probabilities

- PndAnalysis::FillList is a short-cut to this funtionallity via

```
ana->FillList(..., "PidAlgoEmcBayes;PidAlgoDrc");
```

Particle Identification Concept

	Event 1			*Event 2*			...
	TRK 1	**...**	**TRK m1**	**TRK 1**	**...**	**TRK m2**	
PidAlgoEmcBayes	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	
PidAlgoDrc	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	
...	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	
PidAlgoStt	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	

Read

PndAnaPidCombiner
Multiply & Normalize

Write

	Event 1			*Event 2*			...
	TRK 1	**...**	**TRK m1**	**TRK 1**	**...**	**TRK m2**	
RhoCandidate	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	$P_e, P_\mu, \dots, P_p$	...	$P_e, P_\mu, \dots, P_p$	

PndAnaPidSelector / RhoSimple...Selector
Cut on

Stand-alone usage of PID-Selector

- Selector can be used independent of PndAnalysis::FillList
- Say, you have a list of charged particles and want to identify loose kaons with PidAlgoDrc & PidAlgoMvd

```
...
PndAnalysis *ana = new PndAnalysis();

RhoCandList charged, kaonLoose;

PndAnaPidSelector kaonSel("KaonSelector");
kaonSel.SetSelection("KaonLoose");              // set selection criterion

PndAnaPidCombiner pidComb("PidCombiner");
pidComb.SetTcaNames("PidAlgoDrc;PidAlgoMvd"); // set algo‘s

while ( ana->GetEvent() )
{
  ana->FillList(charged, "Charged");          // start w/ charged candidates

  pidComb.Apply(charged);                     // copy P to candidates
  kaonSelector.Select(charged, kaonLoose);    // select kaons from charged
...
}
```

Possible selection keywords are described in Rho class documentation wiki!

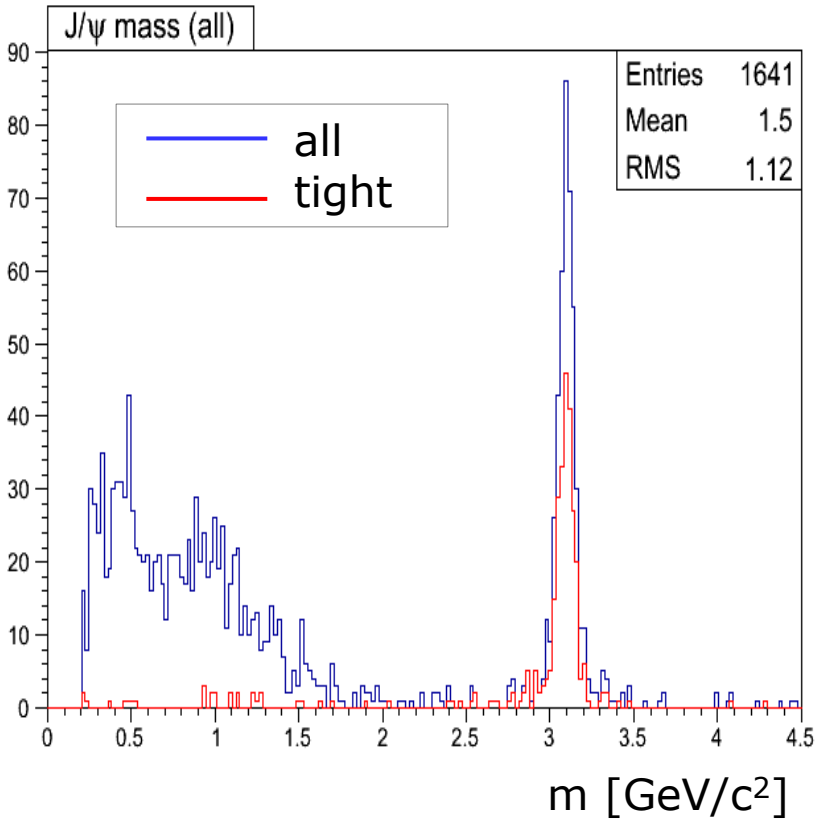
PID: Additional Notes

- There are some default PID-criteria (*[ANAPidSelections]* set in `ana.par` in `macro/params/`)
  - "VeryLoose" : $P_i \geq 0$
  - "Loose" : $P_i \geq 0.25$ *(was previously 0.2)*
  - "Tight" : $P_i \geq 0.5$
  - "VeryTight" : $P_i \geq 0.9$
  - "Variable" : $P_i \geq 0.5$ (user value)
  - "Best" : $P_i \geq P_j, \forall\ j \neq i$

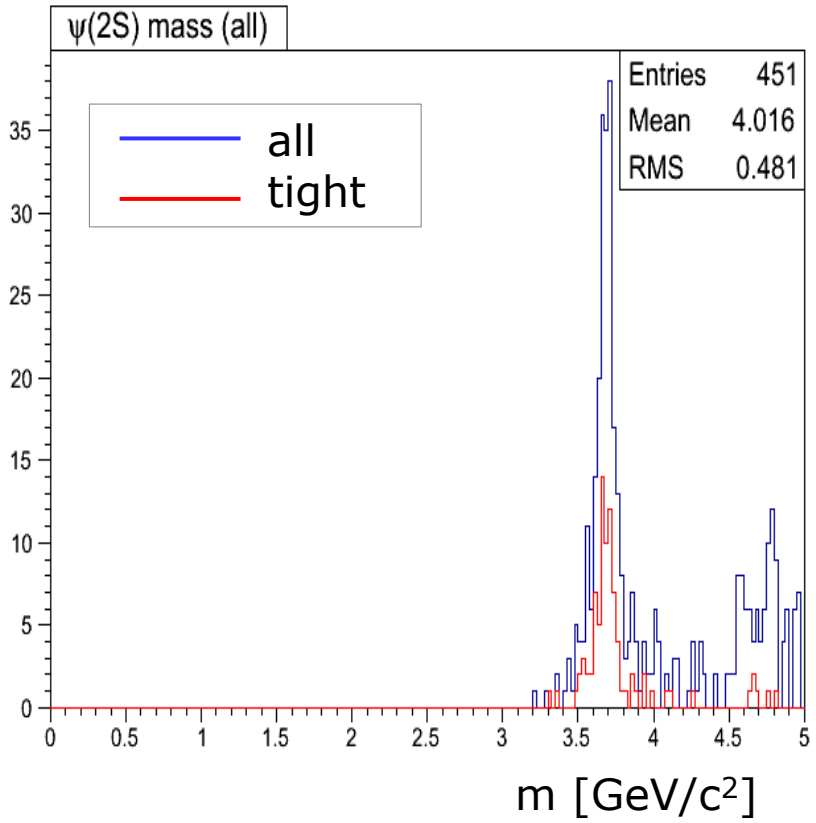
- **Ideal PID algorithm**: PidAlgoIdealCharged
  - During reconstruction looks up true MC particle
  - Sets P = 1 for correct species, for all other P = 0
  - *Not equivalent to MC truth match!*

Example: PID

$J/\psi \rightarrow \mu^+\mu^-$



$\psi(2S) \rightarrow J/\psi\ \pi^+\pi^-$



$\mu^{\pm}$: "PidAlgoMdtHardCuts"
$\pi^{\pm}$: "PidAlgoMvd;PidAlgoStt;PidAlgoDrc"

MC Truth Access

- For final state particles via RhoCandidate::GetMcTruth

```
ana->FillList( eplus,  "ElectronTightPlus",  "PidAlgoEmcBayes" );
RhoCandidate *truth = eplus[0]->GetMcTruth();
```

- The MC truth particles have the complete genealogy

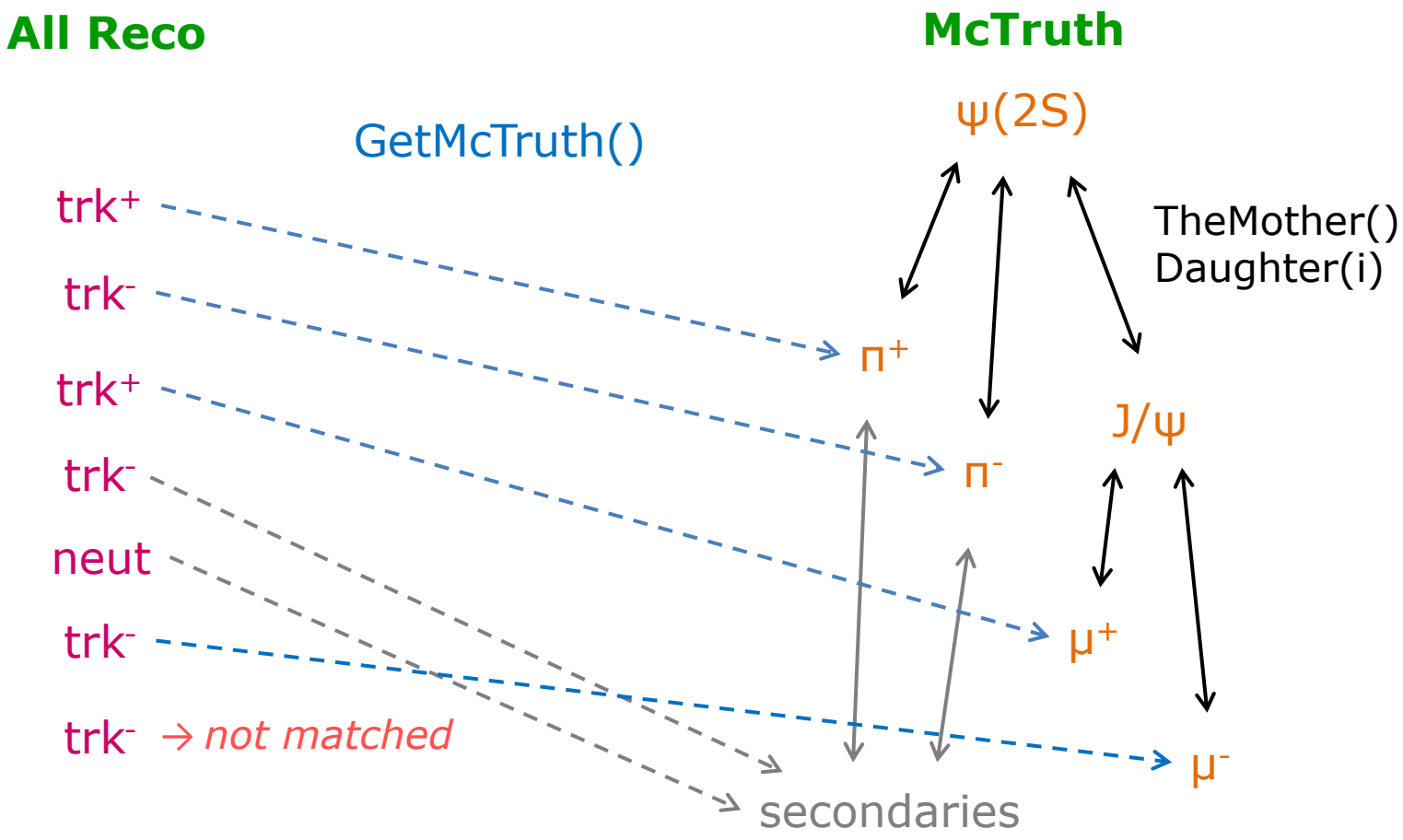
```
if ( truth != 0x0 ) {
   RhoCandidate *mother = truth->TheMother();
   if ( truth->NDaughters() > 1 ) {
      RhoCandidate *firstDaughter  = truth->Daughter(0);
      RhoCandidate *secondDaughter = truth->Daughter(1);
   }
}
```

- Also complete MC truth list available

```
ana->FillList( mct,  "McTruth" );
...
RhoCandidate *firstDaughter = mct[0]->Daughter(0);
```

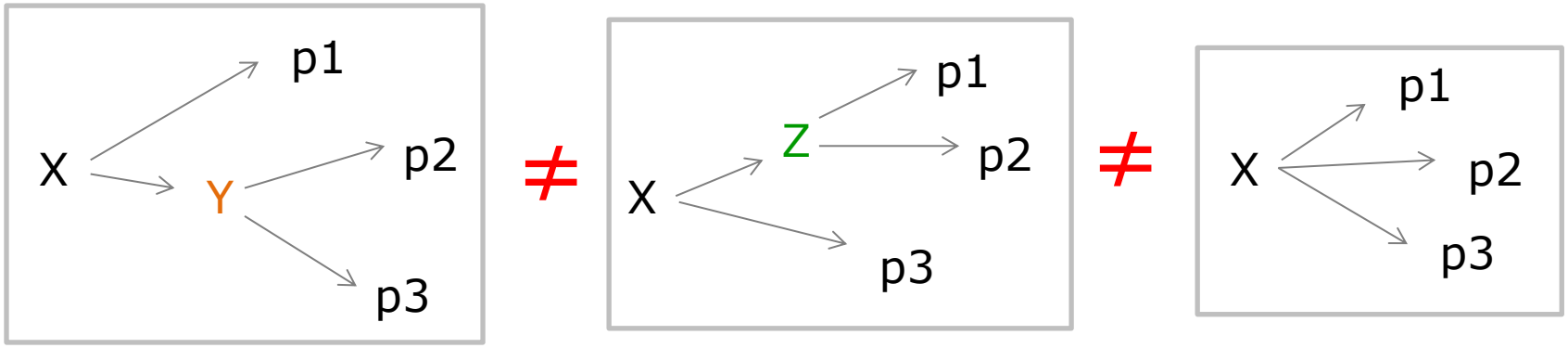
Mc Truth Genealogy

- **Idea**: Each Reco points to an McTruth object, from which the full true tree can be accessed



PndAnalysis: MC Truth Match of Composites

- Physics analysis might require a full mc truth match for composite particles for efficiency calculation, because



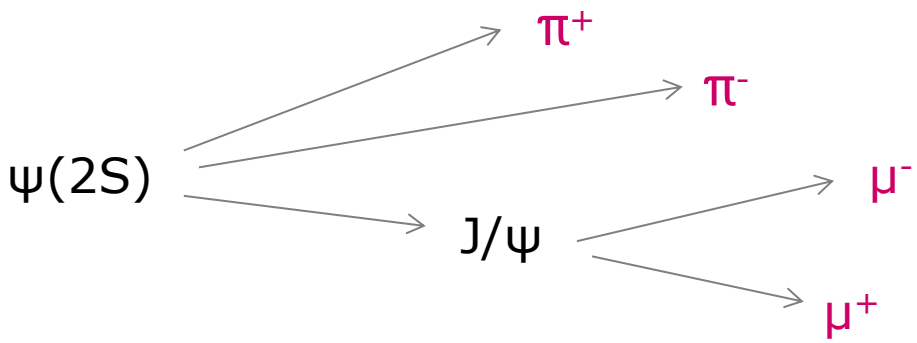
- Functionality, formerly hosted in PndMcTruthMatch now in

```
PndAnalysis::McTruthMatch( RhoCandidate* );
```

Requires PndEvtGenDirect with SetStoreTree!

Example: ψ(2S) → J/ψ ($\mu^+\mu^-$) $\pi^+\pi^-$

- We want to reconstruct the following decay



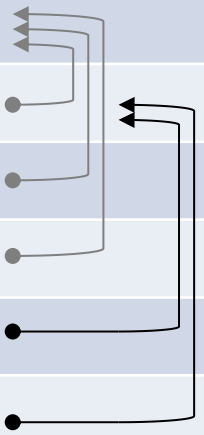

```
RhoCandList muplus, muminus, piplus, piminus, jpsi, psi2s;

ana->FillList( muplus,  "MuonLoosePlus",  myPidAlgos );
ana->FillList( muminus, "MuonLooseMinus", myPidAlgos );
ana->FillList( piplus,  "PionLoosePlus",  myPidAlgos );
ana->FillList( piminus, "PionLooseMinus", myPidAlgos );

jpsi.Combine( muplus, muminus );
psi2s.Combine( jpsi, piplus, piminus );
```

Example: ψ(2S) → J/ψ ($\mu^+\mu^-$) $\pi^+\pi^-$

#	reco
1	trk+
2	trk+
3	trk+
4	trk-
5	trk-
6	trk-
7	trk-
8	neut
9	neut
10	neut
11	neut

GetMcTruth()

#	truth	TheMother()
1	psi(2S)	
2	J/psi	
3	pi+	
4	pi-	
5	mu+	
6	mu-	
7	gamma	
8	gamma	
9	e+	
10	mu-	
11	gamma	
12	gamma	
13	pi-	

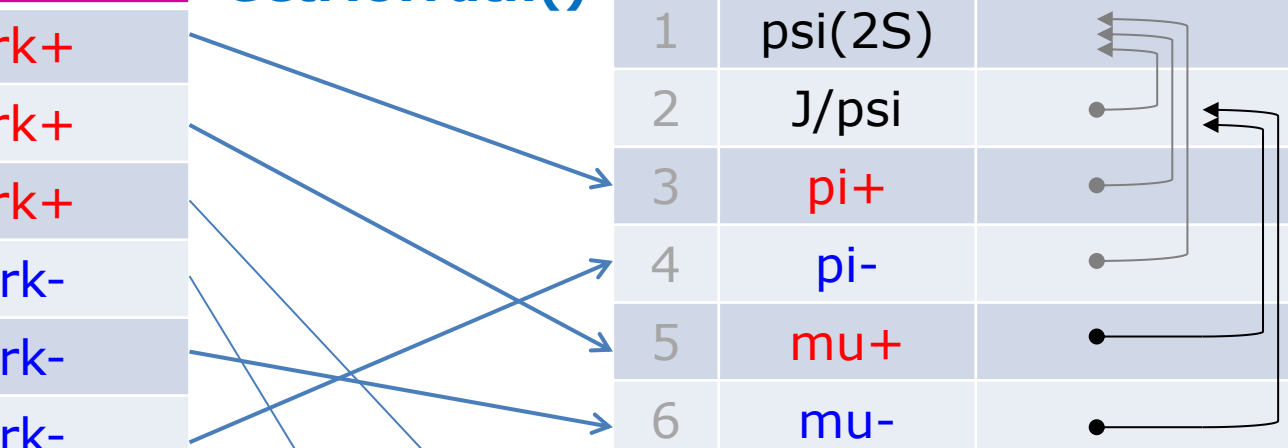


Example: ψ(2S) → J/ψ ($\mu^+\mu^-$) $\pi^+\pi^-$

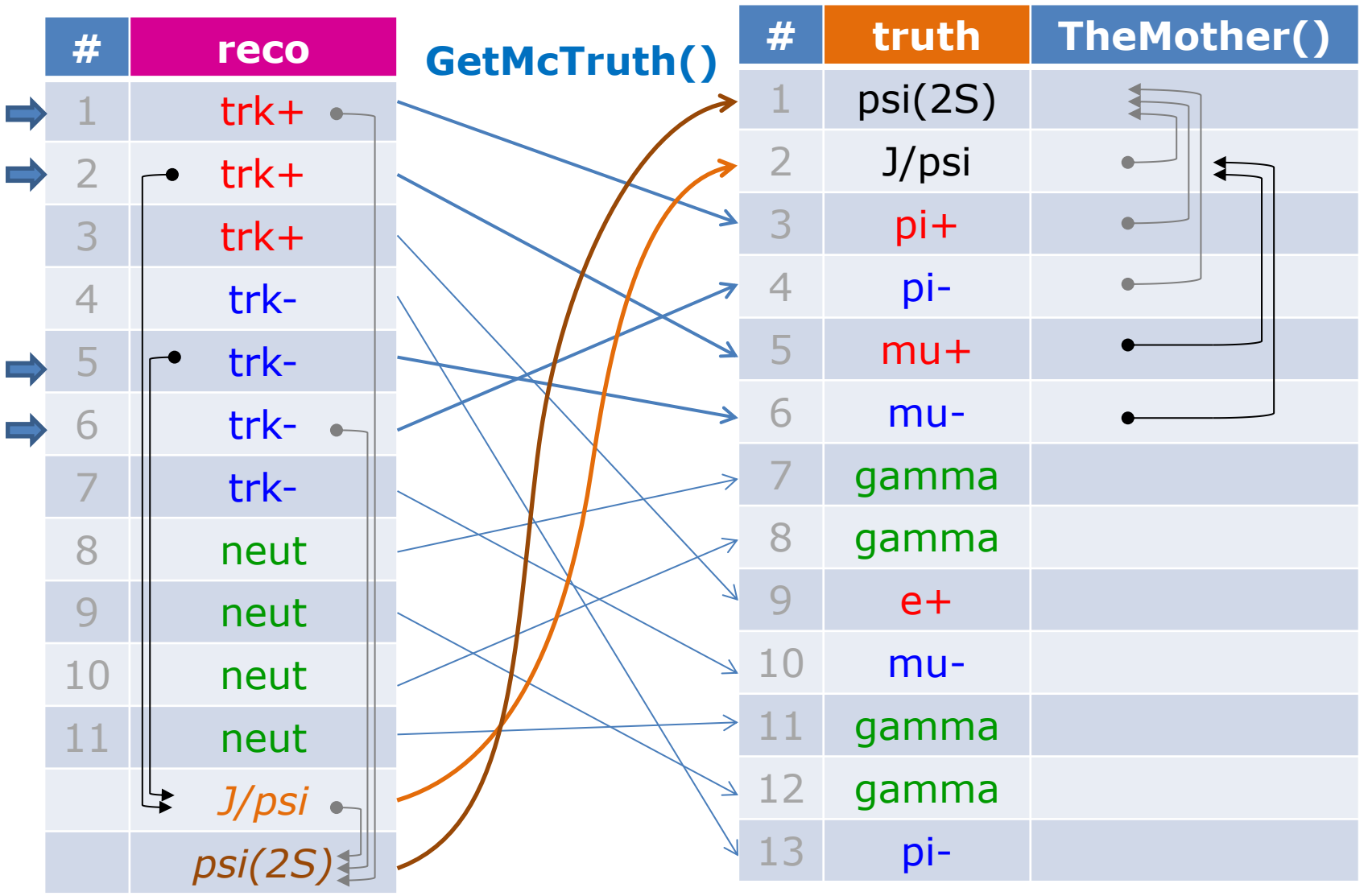
#	reco
1	trk+
2	trk+
3	trk+
4	trk-
5	trk-
6	trk-
7	trk-
8	neut
9	neut
10	neut
11	neut
	J/psi
	psi(2S)

GetMcTruth()

#	truth	TheMother()
1	psi(2S)	
2	J/psi	
3	pi+	
4	pi-	
5	mu+	
6	mu-	
7	gamma	
8	gamma	
9	e+	
10	mu-	
11	gamma	
12	gamma	
13	pi-	



Example: ψ(2S) → J/ψ ($\mu^+\mu^-$) $\pi^+\pi^-$



Usage of PndAnalysis::McTruthMatch

- For matching, composite candidates have to have type set

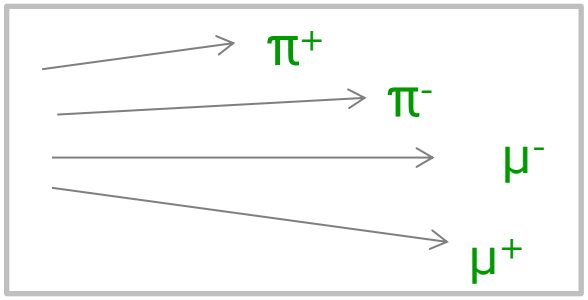
```
PndAnalysis *ana = new PndAnalysis();

while ( ana->GetEvent() )
{
  ana->FillList(muplus, "MuonPlus");
  ...
  jpsi.Combine(muplus, muminus);
  jpsi.SetType( "J/psi" );     // set type for J/psi (names like TDatabasePDG)

  psi2s.Combine(jpsi, piplus, piminus);
  psi2s.SetType( "psi(2S)" );  // set type for psi(2S)

  bool match  = ana->McTruthMatch( psi2s[0] );  // match for RhoCandidate

  int nmatch  = ana->McTruthMatch( psi2s );     // match complete RhoCandList

  for (int j=0; j<psi2s.GetLength(); ++j)
  {
    RhoCandidate *truth = psi2s[j].GetMcTruth(); // access truth of composites
    ...
```

still some inconsistency with TDatabasePDG names
psi(2S) <-> psi'
30443 <-> 100443

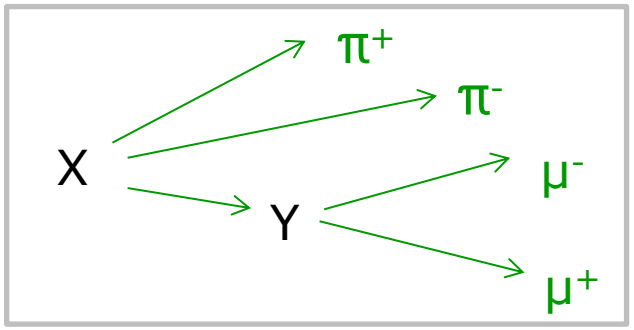
Different levels of matching

- Three different match levels are available via
  `ana->McTruthMatch( psi[0], level );`

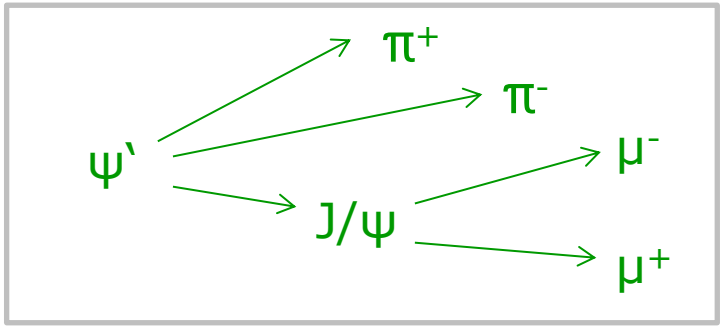
- Level = 0 only looks for correct PID



- Level = 1 matches topology in addition

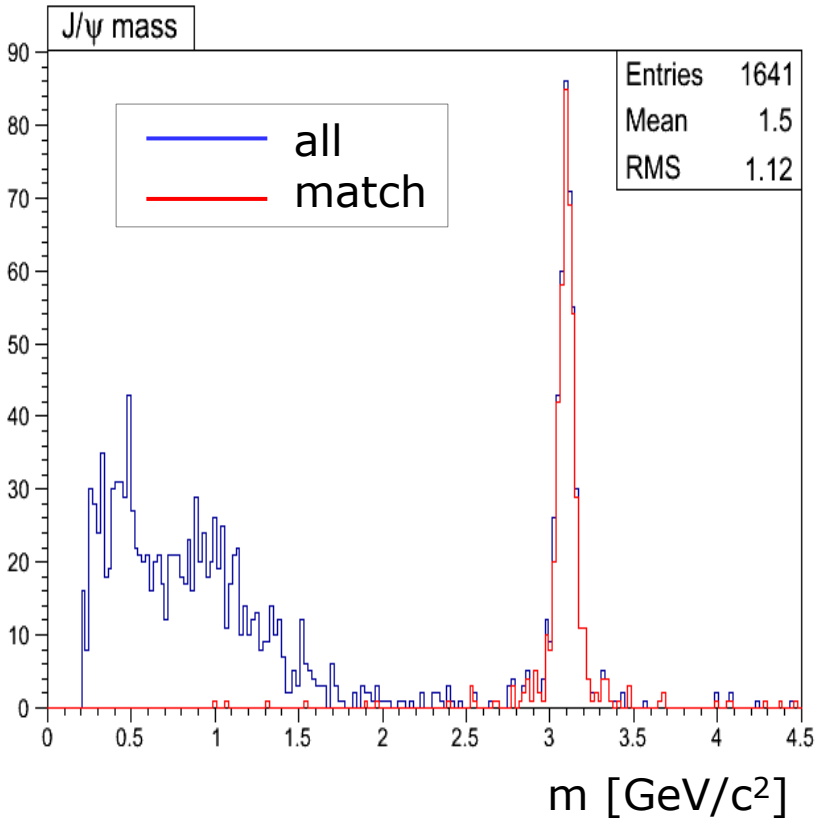


- Level = 2 (default) is full match

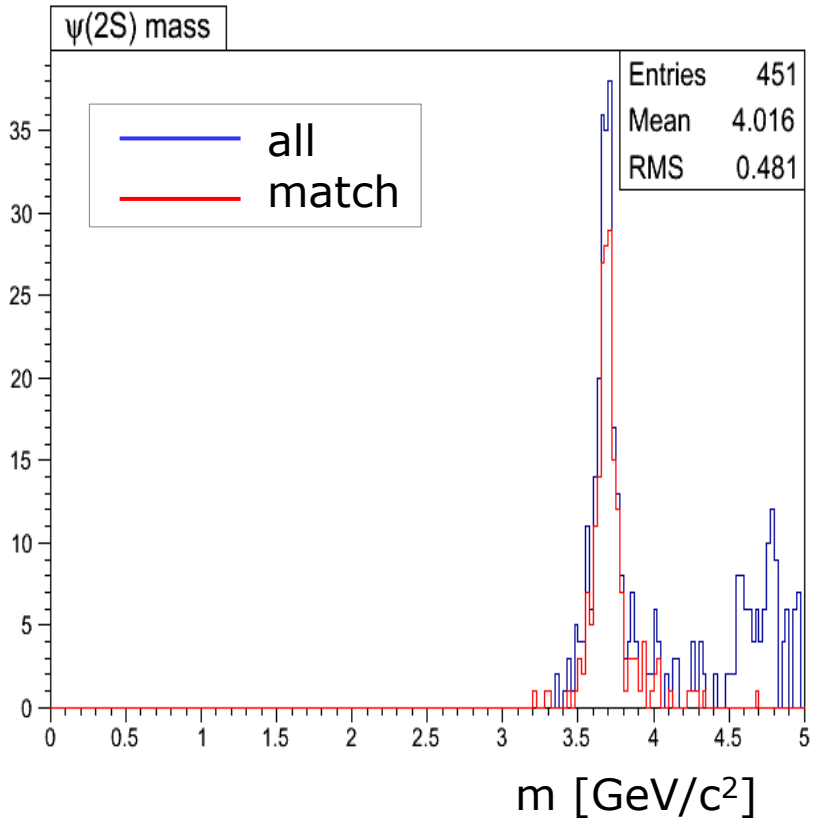


Example: Mc Truth Match

$J/\psi \rightarrow \mu^+\mu^-$



$\psi(2S) \rightarrow J/\psi\ \pi^+\pi^-$



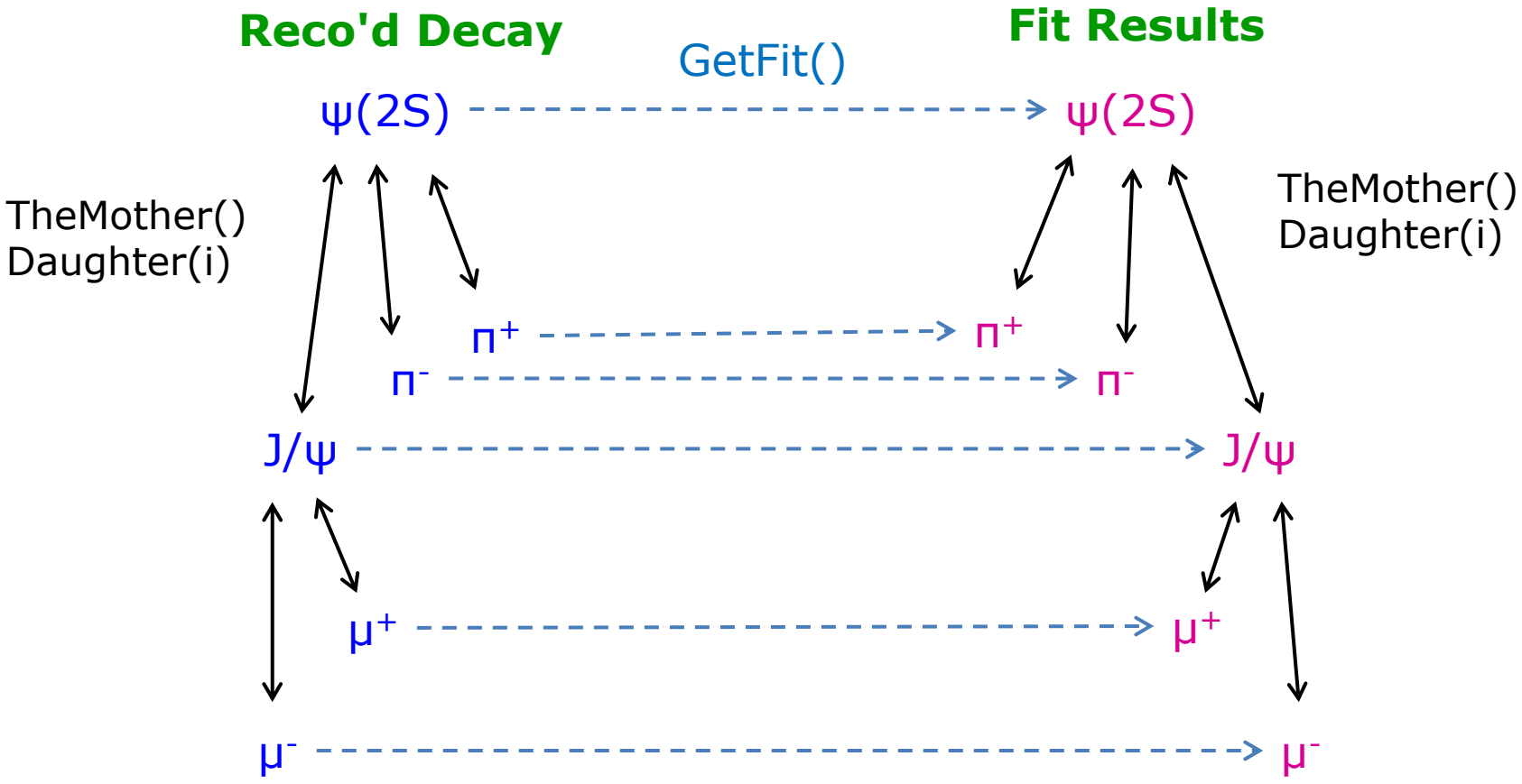
Fitting

- Fitters basically all have a similar interface
- Fit results are now attached to RhoCandidates
- Can be accessed as full tree, allows cascaded fitting!
- E.g. vertex fitting + mass fitting might look like this:

```
RhoCandidate *lambda = pplus->Combine( piminus );
TVector3 IP( 0,0,0 );
  ...
PndKinVtxFitter fitvtx( lambda );                  // setup vertex fitter
fitvtx.AddPointingConstraint( IP );                // add pointing constraint
fitvtx.FitAll();                                   // perform fit

RhoCandidate *lambda_vtx = lambda->GetFit();       // access fit results

PndKinFitter fitmass( lambda_vtx );                // setup mass fitter
fitmass.SetMassConstraint( 1.115 );                // set mass constraint
fitmass.Fit();                                     // perform fit

RhoCandidate *lambda_mass = lambda_vtx->GetFit(); // access cascaded fit results

RhoCandidate *fit_pplus   = lambda_mass->Daughter( 0 );
RhoCandidate *fit_piminus = lambda_mass->Daughter( 1 );
```

Fitting: Access to Results

- After fit → full fitted tree is attached to the reco object



Analysis Code in a Task

- Problem:
  - Interpreted code is harder to be debugged
  - It runs slower when using many loops
  - Beyond certain macro-complexity
    → *CINT starts getting slower from event to event*

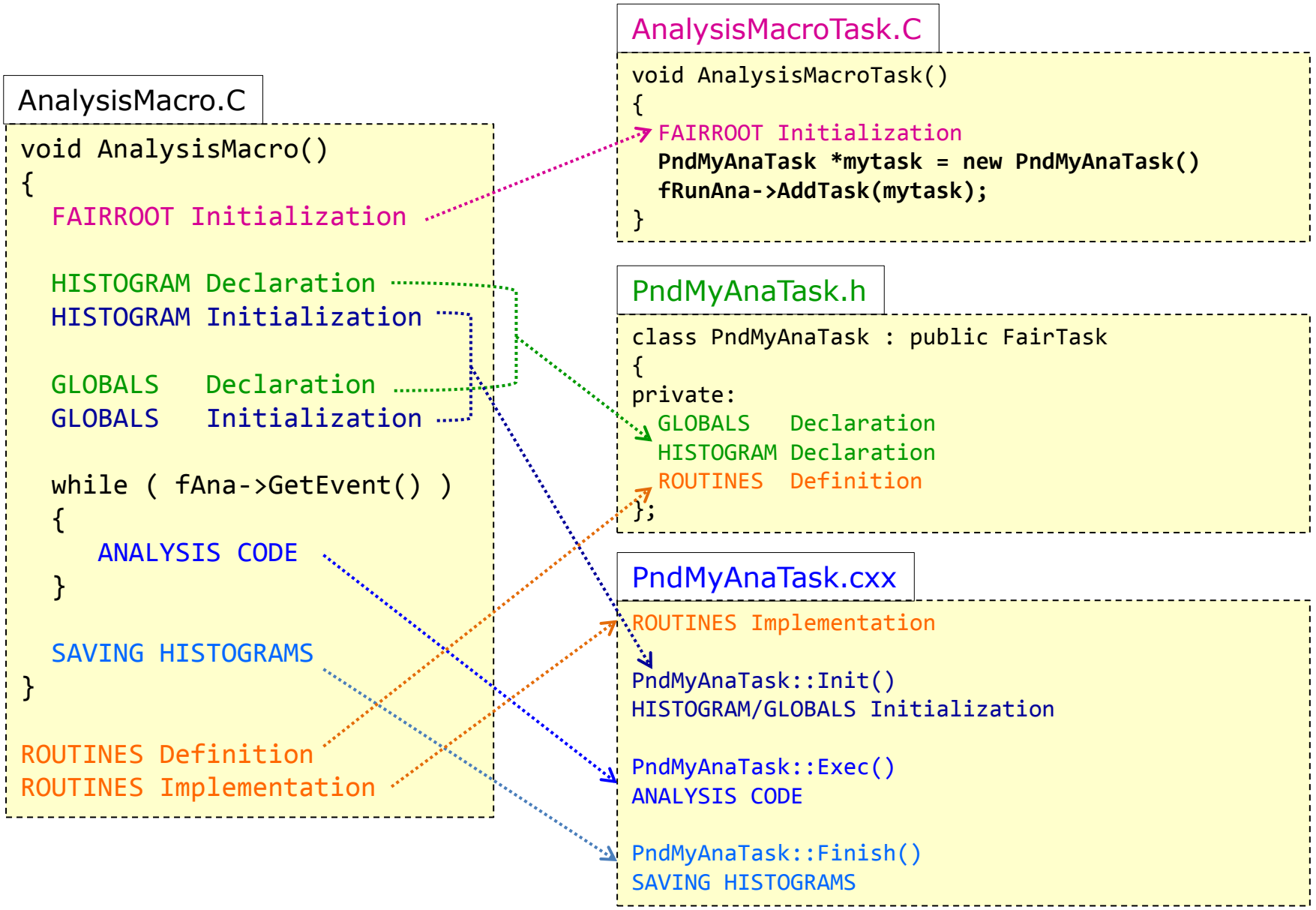
- Solution:
  - Transformation of the macro code into task

    → Code is compiled and thus more stable/reliable
    → Analysis runs (usually) much faster
    → Doesn't have the problem from above

- Much more simple to achieve than you might think

- How-to is given in the new tutorial wiki!

Analysis Code in a Task



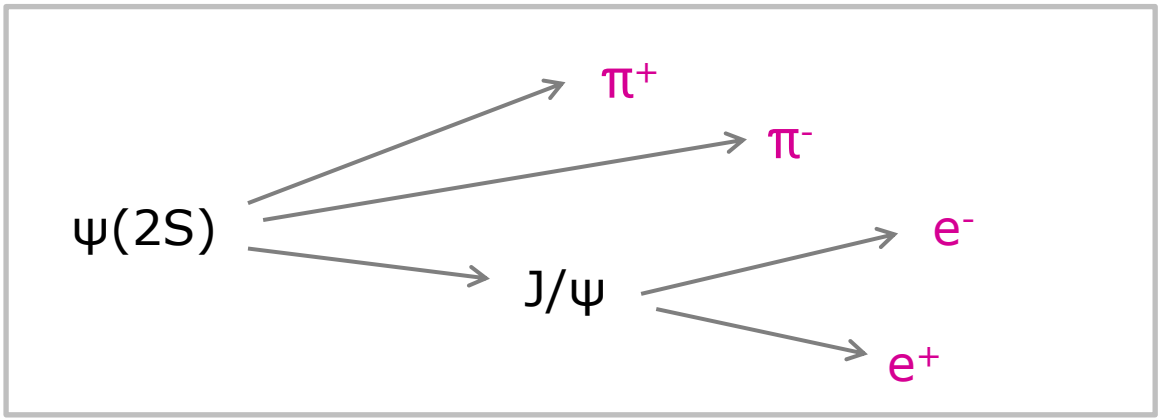
Summary

- New Tutorial is available
  - Plan: Keep functioning (+ extend it) as standard tutorial
  - Rho Class Docu available in addition
- Extended PID concept
  - Easy use with PndAnalysis::FillList
  - Stand-alone use possible
- Monte Carlo Truth Match
  - Should work for all kinds of complicated decay trees
  - Allows access to non-final-state MC truth resonances
- Fitting
  - Full fitted tree created and attached to reco object
  - Cascaded fitting possible on fit result
- Analysis in Task
  - Documented in Tutorial → more stable running

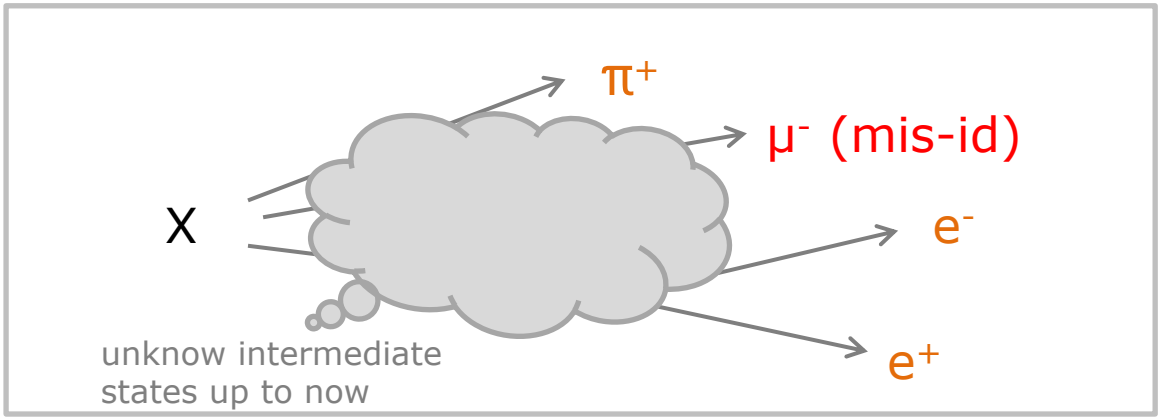
BACKUP

PndMcTruthMatch

- **Checklist** for full tree match:
  1. truth objects of final states have the correct PID types

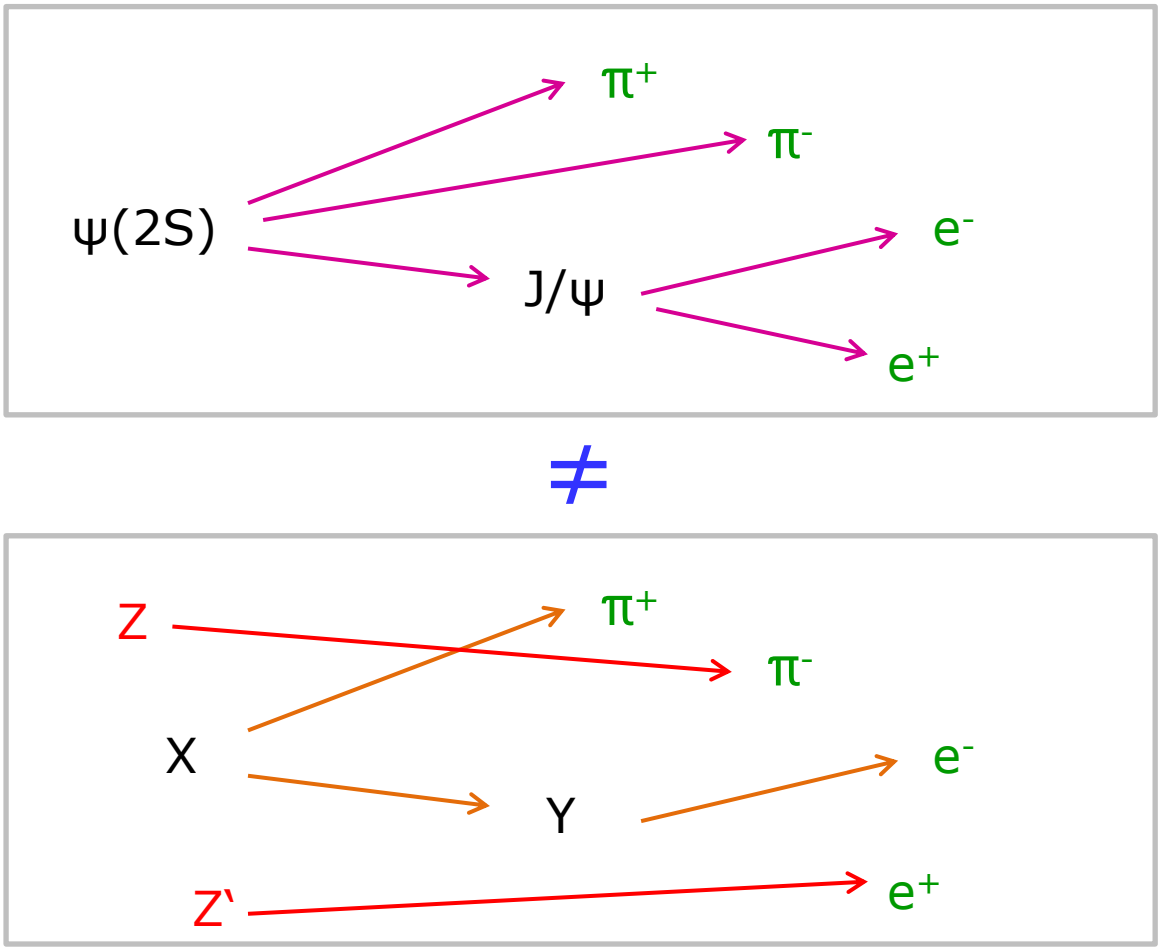


≠



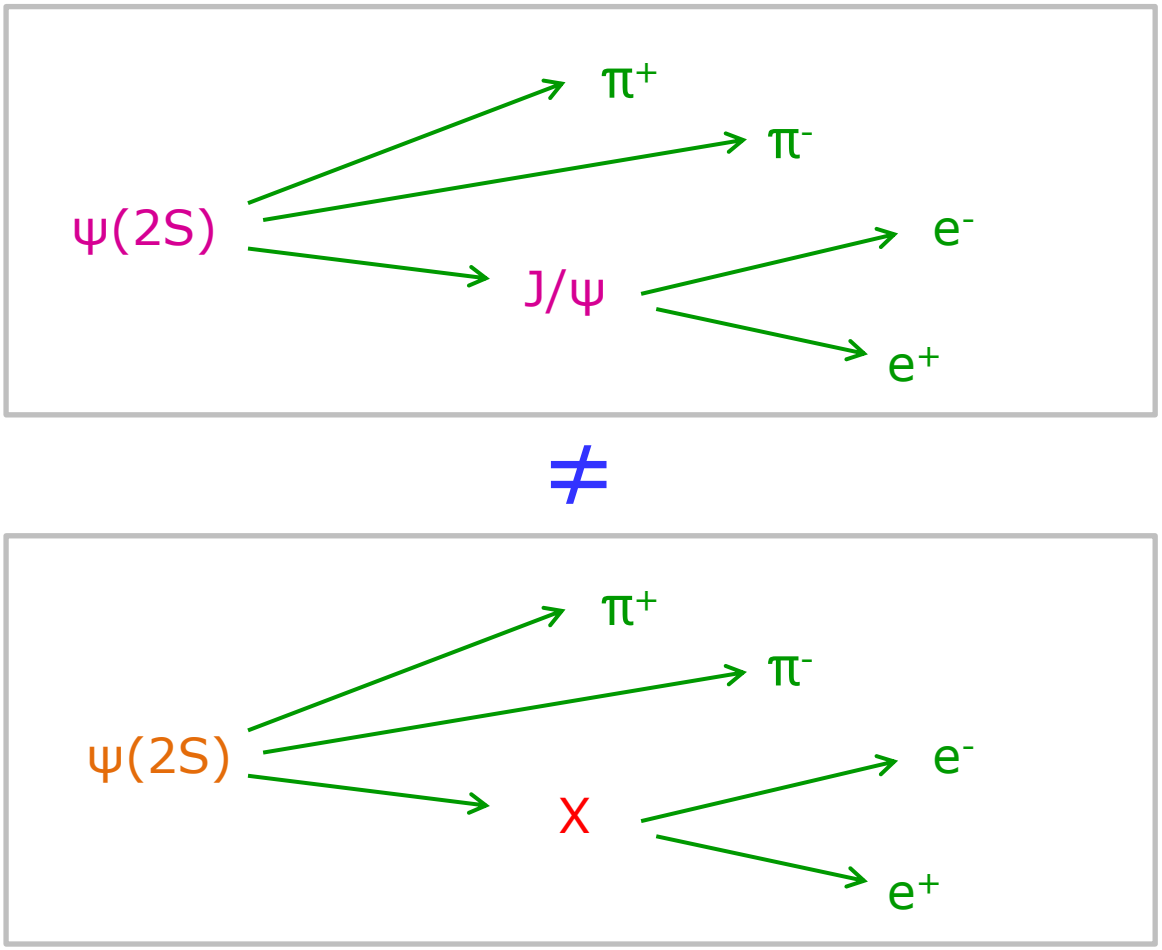
PndMcTruthMatch

- **Checklist** for full tree match:
  2. truth object of final states have the same mother



PndMcTruthMatch

- **Checklist** for full tree match:
  3. mother has required type



PndMcTruthMatch

- **Checklist** for full tree match:
  4. mother has correct number of daughters