Table of Contents

LoKi-based Decay Finders

Introduction

New LoKi-based Decay Finders, being developed by Alexander Mazurov are based on "tree/node" concept, which allows to describe/construct the decay descriptions of the arbitrary complexity

See interesting presentation by Patrick Spradlin, at T-Rec October 21, 2k+13.

Warning

In DaVinci versions prior to v33r6p1 CombineParticles did not use the Loki based decay finder, but a different tool. This tool is now unmaintained, so as of v33r6p1 the LoKi tool is the default decay finder used by this algorithm. To configure CombineParticles in old DaVinci releases, see DaVinciTutorial4.

The basic usage

DaVinci

The usage in DaVinci can be illustrated by four-step procedure:

Locate the tool ("factory")

```
1000/// locate the tool
1010  SmartIF<Decays::IMCDecay> decay =  tool<Decays::IMCDecay> ( "LoKi::MCDecay" , this ) ;
1020  if ( !decay ) { return Error ( "Unable to locate MC-decay finder" ) ; }
1030
```

Parse decay descriptor

```
1000// parse/decode the decay descriptor
1010  Decays::IMCDecay::Tree tree = decay->tree ( " [ [B0]nos => pi+ pi- ]CC" ) ;
1020  if ( !tree  ) { return Error ( "Unable to decode/parse decay descriptor" ) ; }
1030
```

Create the decay finder

```
1000// create the decay finder
1010  Decays::IMCDecay::Finder finder ( tree ) ;
1020  if ( !finder  ) { return Error ( "Unable to create decay finder'" ) ; }
1030
```

Use it!

```
1000// get all MC-particles:
1010  const LHCb::MCParticle::Container* mcparticles = get<LHCb::MCParticle::Container>( LHCb::M
1020
1030  // prepare the output container for "good" decays:
1040  typedef LHCb::MCParticle::ConstVector OUTPUT ;
1050  OUTPUT  output ;
1060
1070  // collect the decays
1080  finder.findDecay
1090    ( mcparticles -> begin() ,   // begin of input
1100      mcparticles -> end  () ,   // end of the input
1110      output                 ) ; // the output container
```

```
1120
1130  MsgStream& log = info() ;
1140  log << " found #" << output.size() << " decays" ;
1150  for ( OUTPUT::const_iterator idec = output.begin() ; output.end() != idec ; ++idec )
1160  {
1170    const LHCb::MCParticle* dec = *idec ;
1180    if ( 0 == dec ) { continue ; }
1190    log << std::endl << " " << (idec-output.begin()+1) << " \t" ;
1200    LoKi::PrintMC::printDecay ( dec , log ) ;
1210  }
1220  log << endmsg ;
1230
```

The complete DaVinci algorithm is available here:

The corresponding python configuration file is :

```
1000#!/usr/env gaudirun.py
1010##############################################################################
1020# $Id: LoKiNewDecayFinders.txt,v 1.24 2018/07/09 21:17:36 joroth Exp $
1030#
1040# Configuration file to run MCDecayFinder example
1050#
1060# @author Vanya BELYAEV Ivan.Belyaev@nikhef.nl
1070# @date 2009-06-28
1080#
1090##############################################################################
1100from Configurables import DaVinci, EventSelector
1110from Configurables import DaVinci__Examples__MCDecayFinderEx as MCDecayFinderEx
1120
1130## search for exclusive  decay, but allow indefinite amount of photons for sub-decays
1140alg1 = MCDecayFinderEx (
1150    "PsiPhi_finder" ,
1160    MCDecay = " [B_s0 -> (J/psi(1S) => mu+ mu- ) ( phi(1020)=> K+ K-)]CC"
1170    )
1180
1190## seach for (deep) inclusive semileptonic decays:
1200alg2 = MCDecayFinderEx (
1210    "B->ell_finder" ,
1220    MCDecay = " Xb --> l Neutrino ... "
1230    )
1240
1250DaVinci (
1260    EvtMax         = 20     ,
1270    DataType       = 'DC06' ,
1280    Simulation     = True   ,
1290    UserAlgorithms = [ alg1 , alg2 ]
1300    )
1310
1320EventSelector (
1330    PrintFreq = 1 ,
1340    Input = [
1350    "DATAFILE='PFN:/afs/cern.ch/lhcb/group/calo/ecal/vol10/DATA/Bs2PsiPhi.dst' TYPE='POOL_RO
1360    ]
1370    )
```

Grammar in short

Arrows

Arrow	Reco-particles	Monte Carlo	Generator particles	Description

		particles		
==X>	-	+	-	search in decay "sections" including "non-decay" daughters(interactions?). allowing arbitrary number of additional photons
->	+	+	+	search within the direct daughters
-->	+	+	+	search in decay "sections" (ignore intermediate resonances)
=>	-	+	+	search within the direct daughters, allowing arbitrary number of additional photons
==>	-	+	+	search in decay "sections". allowing arbitrary number of additional photons (ignore intermediate resonances)
-X>	-	+	-	search within the direct daughters, including "non-decay" daughters (interactions?)
--X>	-	+	-	search in decay "sections" including "non-decay" daughters(interactions?)
=X>	-	+	-	search within the direct daughters including "non-decay" daughters(interactions?), allowing arbitrary number of additional photons

Optional components

The optional decay components are specified using braces:

```
1000"( B0 -> pi+ pi- { pi0 }  ) "
1010
1020"( B0 -> pi+ pi- { pi0 -> gamma gamma  } { pi0 }  ) "
```

Inclusive components

The inclusive decay components are specified using ellipses:

```
1000"( B0 -> pi+ pi-  ....  ) "
1010
1020"( B0 -> pi+ pi-  ( pi0 -> gamma gamma  )  ...  ) "
```

Please note that optional and inclusive components are mutually exclusive.

Mixing

Mixing can be specified for Monte Carlo and Generator decays using `[...]os` and `[...]nos` directives:

```
1000"(  [B0]os -> pi+ pi-  ....  ) "
1010
1020"( [B0]nos -> pi+ pi-  ( pi0 -> gamma gamma  )  ...  ) "
1030
1040"(  [  Xb & Meson ]os -> pi+ pi-  ....  ) "
```

Logical operations

The logical operations are specified using `&&`, `||` and `~` operators (exclamation mark also acts as negation):

```
1000" ( (  B0 -> pi+ pi-  ) || ( B+ -> pi+ pi0   ) ) "
1010
1020" ( (  Beauty --> pi+ pi-  ... ) && ( X0 ==> pi+ pi- {pi0} {pi0} {pi0} ) ) "
1030
1040" ( (  B0 -> pi+ pi-  ) &&  !( B+ -> pi+ pi0   ) ) "
1050
1060 " (  B0 -> pi+ pi-  ) &&  ~( B+ -> pi+ pi0   ) ) "
```

Also the list operation (equivalent to `||`) is allowed:

```
1000"[ (  B0 -> pi+ pi-  ) , ( B+ -> pi+ pi0   )  , ( B- -> l- ... ) ] "
```

Marked components

The marked components of the decay descriptor are indicated using `^`-symbol:

```
1000" (  B0 -> ^pi+ ^pi-  ) "
```

Note: Negation disables the marked elements

For composite decay nodes (like the sub-decays) , the caret symbol should be added before the whole node:

```
1000" (  D0 ->  pi+  pi-   ^( KS0 ->  pi+ pi- )  ) "
```

In other words: the head of the decays can't be marked explicitely

Ignore indefinite number of photons

PHOTOS generator used in LHCb is able to produce an indefinite amount (up to 6 for the current version) of soft bremstrahlung photons for decay leg. One can ignore these photons in the decay descriptor by specification of "thick" arrow (note: this is inclusive, any decay with extra photons will be matched):

```
1000"(  B0 => pi+ pi-  ) "
1010
1020" ( B+ ==> pi+ pi+ pi- ) "
```

How to specify charge conjugated decay?

For the top-level decay (as well as any intermediate decays), the charge conjugation can be specified using the `[...]CC` directive:

```
1000" [( B+ ==> pi+ pi+ pi+ ) ]CC "
```

Such expression is just a shortcut for :

```
1000" [  ( B+ ==> pi+ pi+ pi+ ) ,  ( B- ==> pi- pi- pi+) ]"
```

Moreover such substitution is performed on "pre-parse" phase.

For individual decay 'nodes', the charge conjugation can be specified using `[...]cc` directive:

```
1000"[ [B0]cc => K+ pi- ]CC"
```

Such expression is just a shortcut for :

```
1000" ( ( [B0]cc =>  K+   pi- ) || ( [B~0]cc =>  K-   pi+ ) )"
```

How to get the full list of particle names, known for Particle Properties Service?

In C++

```
1000#include "Kernel/IParticlePropertySvc.h"
1010#include "Kernel/Symbols.h"
1020
1030const LHCb::IParticlePropertySvc* service = ....;
1040
1050Decays::Symbols::Names names ;
1060const Decays::Symbols& s = Decays::Symbols::instance() ;
1070s.particles ( service , names ) ;
1080
1090for ( Decays::Symbols::Names::const_iterator iname = names.begin () ; names.end() != iname ;
1100{
1110    std::out << " Particle name: " <<  (*Iname) << std::endl ;
1120}
```

In Python

```
1000import PartProp.PartPropSvc
1010
1020from GaudiPython.Bindings import AppMgr
1030
1040gaudi = AppMgr()
1050
1060# get the service from Gaudi:
1070ppSvc = gaudi.ppSvc()
1080
1090# loop over all particle properties:
1100for pp in ppSvc :  print pp.particle()
1110
1120# print them as table:
1130print ppSvc.all()
```

Via shell

One also can use the special script that dumps the table of particle properties:

```
1000lb-run DaVinci/v42r4 dump_ParticleProperties
```

How to get the full list of *symbols* ?

In C++

```
1000#include "Kernel/Symbols.h"
1010
1020Decays::Symbols::Names names ;
1030const Decays::Symbols& s = Decays::Symbols::instance() ;
1040s.symbols( names ) ;
1050
1060for ( Decays::Symbols::Names::const_iterator iname = names.begin () ; names.end() != iname ;
1070{
1080    std::out << " Symbol: " <<  (*Iname) << std::endl ;
1090}
```

In Python

```
1000from PartProp.decorators import Symbols
1010
1020for s in Symbols :  print s
1030
1040# get the embedded help for symbol:
1050
1060s = ...
1070
1080print Symbols.symbol ( s  )
```

Current (July 2k+9) list of known symbols

Symbol	Description
StableCharged	match any charged particle with nominal proper lifetime (in `c*tau` units) in excess of 1 meter
X	Matches **any** particle
Hadron	match **any** hadron, according to `LHCb::ParticleID::isHadron()`
Meson	match **any** meson, according to `LHCb::ParticleID::isMeson()`
Baryon	match **any** baryon, according to `LHCb::ParticleID::isBaryon()`
Nucleus	match **any** nucleus, according to `LHCb::ParticleID::isNucleus()`
Lepton	match **any** lepton, according to `LHCb::ParticleID::isLepton()`
l	match any charged lepton, according to `LHCb::ParticleID::isLepton()` and `LHCb::ParticleID::threeCharge()`
l+	match any positively charged lepton
l-	match any negatively charged lepton
Nu	match any neutrino (neutral lepton)
Neutrino	match neutrino (neutral lepton)
X0	match any neutral particle according to `LHCb::ParticleID::threeCharge()`
X+	match any positively charged particle
X-	match any negatively charged particle
Xd	match any particle with `down` quark, according to `LHCb::ParticleID::hasQuark( down )`
Xu	match any particle with `up` quark, according to `LHCb::ParticleID::hasQuark( up )`
Xs	match any particle with `strange` quark, according to `LHCb::ParticleID::hasQuark( strange )`
Xc	match any particle with `charm` quark, according to `LHCb::ParticleID::hasQuark( charm )`
Xb	match any particle with `bottom` quark, according to `LHCb::ParticleID::hasQuark( bottom )`
Xt	match any particle with `top` quark, according to `LHCb::ParticleID::hasQuark( top )`
Down	match any particle with `down` quark, according to `LHCb::ParticleID::hasQuark( down )`

Up	match any particle with `up` quark, according to `LHCb::ParticleID::hasQuark( up )`
Strange	match any particle with `strange` quark, according to `LHCb::ParticleID::hasQuark( strange )`
Charm	match any particle with `charm` quark, according to `LHCb::ParticleID::hasQuark( charm )`
Beauty	match any particle with `bottom` quark, according to `LHCb::ParticleID::hasQuark( bottom )`
Bottom	match any particle with `bottom` quark, according to `LHCb::ParticleID::hasQuark( bottom )`
Top	match any particle with `top` quark, according to `LHCb::ParticleID::hasQuark( top )`
Scalar	match any scalar particle, according to `LHCb::ParticleID::jSpin`
Spinor	match any spin-1/2 particle, according to `LHCb::ParticleID::jSpin`
OneHalf	match any spin-1/2 particle, according to `LHCb::ParticleID::jSpin`
Vector	match any vector (spin-1) particle, according to `LHCb::ParticleID::jSpin`
ThreeHalf	match any spin-3/2 particle, according to `LHCb::ParticleID::jSpin`
Tensor	match any tensor (spin-2) particle, according to `LHCb::ParticleID::jSpin`
FiveHalf	match any spin-5/2 particle, according to `LHCb::ParticleID::jSpin`
ShortLived	match any particle with nominal proper lifetime (in `c*tau` units) less than 0.1 micrometer
LongLived	match any particle with nominal proper lifetime (in `c*tau` units) in excess of 0.1 micrometer
Stable	match any particle with nominal proper lifetime (in `c*tau` units) in excess of 1 meter

Derived symbols

Symbol	Description
HasQuark (top)	match any particle with `t`-quark
ShortLived_ (double)	match any particle with nominal proper lifetime (in `c*tau` units) less than the specified threshold
LongLived_ (double)	match any particle with nominal proper lifetime (in `c*tau` units) in excess of the specified threshold
Light (double)	match any particle with nominal mass less than the specified threshold
Heavy (double)	match any particle with nominal mass in excess of the specified threshold
JSpin (positive int)	match any particle with total spin `j` such as `2j+1` is equal to the specified value
LSpin (positive int)	match any particle with `l`-spin `l` such as `2l+1` is equal to the specified value
SSpin (positive int)	match any particle with `s`-spin `l` such as `2s+1` is equal to the specified value
HasQuark (down)	match any particle with `d`-quark
HasQuark (up)	match any particle with `u`-quark
HasQuark (strange)	match any particle with `s`-quark
HasQuark (charm)	match any particle with `c`-quark

HasQuark (beauty)	match any particle with `b`-quark
HasQuark (bottom)	match any particle with `b`-quark

Nodes

EBNF grammar for nodes

The EBNF grammar for valid nodes can be expressed as:

```
1000spin   =
1010            ( "JSpin" >> '(' >> uint  >> ')' ) |
1020            ( "LSpin" >> '(' >> uint >> ')' ) |
1030            ( "SSpin" >> '(' >> uint >> ')' ) ;
1040
1050          tm =
1060            ( "LongLived_"   >> '(' >> real >> ')') |
1070            ( "ShortLived_"  >> '(' >> real >> ')') |
1080            ( "Light"             >> '(' >> real >> ')') |
1090            ( "Heavy"          >> '(' >> real >> ')') ;
1100
1110          quark  = "HasQuark"  >> '(' >> (  "up"  | "down"  |  "strange" | "charm" | "beauty
1120
1130          parts =  list of all valid particle names
1140
1150          symbs = list of all valid symbols
1160
1170          atomic =  spin | tm    | quark |   (  '[' >> parts >> ']cc' ) | parts | symbs    ;
1180
1190           expression  = atomic | operation | (  (  "~" | "!" ) >> expression ) ;
1200
1210          operation  = "("  >> expression >> +(  ( "&"  >> expression ) | ( "|"  >> expressi
1220
1230          result = expression
```

Decay Trees

EBNF grammar for decay trees:

The EBNF grammar for valid decay trees can be expressed as (see also EBNF-grammar for nodes):

```
1000head =  ( "[" >> node >> "]nos" ) |  ( "[" >> node >> "]os"  ) | node
1010
1020          expression = tree | operation  | head | (   "^" >> expression )  | (  ( "~"|"!")
1030
1040          tree = "(" >> head
1050            >> (  ( "==x>" | "=x>"  | "--x>" | "-x>"  | "==>" | "=>"  | "-->" | "->"   )
1060            >>   expression >> *( expression | ( "{" >> expression >> '}' ) )   >> !("...")
1070
1080          operation  = ( "("  >> expression >> +( (  "&&"  >> expression ) | ( "||"   >> exp
1090                               ( "["  >> expression >> + (  "," >> expression ) >> "]" )
1100
1110          result = expression;
1120
```

Implementation of charge-conjugation for decay trees

- The construction `cc( childtree )` (see lines 1380-1400) acts as charge conjugated for `childtree`

- The construction `[childtree]cc` (see lines 1380-1400) is equivalent to (and implemented as) `childtree | cc(childtree)`

The implementation of `cc` should be done in following way:

```
1000const LHCb::IParticlePropertySvc* service = ... ;
1010
1020// "cc(...)" or "[..]cc" descriptor:
1030const std::string&  descriptor = ...;
1040
1050// get the decay descriptor with removed "[...]cc" and "cc(...)" :
1060const std::string childtree = ... ;
1070
1080// get its charge conjugated:
1090std::string cctree = service->cc ( childtree ) ;
1100
1110typedef Decays::Tree_<PARTICLE> Tree ;
1120
1130// for  "[ ...]cc" case:
1140{
1150  Tree orig      = some_way_to_create_tree ( childtree )  ; // create the the tree using "ch
1160  Tree orig_cc = some_way_to_create_tree ( cctree ) ; // create the tree using "cctree"
1170  Tree result_1 = orig || orig_cc ;
1180}
1190
1200// for "cc(...)" case:
1210{
1220  Tree result_2 = some_way_to_create_tree ( cctree ) ; // create the tree using "cctree"
1230}
```

Helper Factories for creation of decay trees

There are helper factories for creation of decay trees:

```
1000/// The file $LOKIMCROOT/LoKi/MCTreeFactory.h
1010
1020namespace Decays
1030{
1040  namespace Trees
1050  {
1060    // ========================================================================
1070    /** "Factory" to create the proper Tree from the full description
1080     *   @param tree       (OUTPUT) the constructed tree
1090     *   @param mother     (INPUT)  the mother particle
1100     *   @param oscillated (INPUT)  the oscillation flag
1110     *   @param arrow      (INPUT)  the type of arrow
1120     *   @param daughters  (INPUT)  the list of daughter trees
1130     *   @param inclusive  (INPUT)  the flag for inclusive
1140     *   @param optional   (INPUT)  the list of "optional" particles
1150     *   @return StatusCode
1160     *   @author Vanya BELYAEV Ivan.Belyaev@nikhef.nl
1170     *   @date   2009-05-06
1180     */
1190    StatusCode factory
1200    ( Decays::Trees::Types_<const LHCb::MCParticle*>::Tree&           tree       ,
1210      const Decays::iNode&                                            mother     ,
1220      const Decays::Trees::Oscillation&                               oscillated ,
1230      const Decays::Trees::Arrow&                                     arrow      ,
1240      const Decays::Trees::Types_<const LHCb::MCParticle*>::SubTrees& daughters  ,
1250      const bool                                                      inclusive  ,
1260      const Decays::Trees::Types_<const LHCb::MCParticle*>::SubTrees& optional   ) ;
1270    // ========================================================================
1280    /** "Factory" to create the proper Tree from "short" descriptor
1290     *   @param tree       (OUTPUT) the constructed tree
```

```
1300     *   @param mother     (INPUT)  the mother particle
1310     *   @return status code
1320     *   @author Vanya BELYAEV Ivan.Belyaev@nikhef.nl
1330     *   @date   2009-05-06
1340     */
1350    StatusCode factory
1360    ( Decays::Trees::Types_<const LHCb::MCParticle*>::Tree& tree       ,
1370      const Decays::iNode&                                  mother     ) ;
1380    // ============================================================================
1390  } // end of namespace Decays::Trees::
1400  // ============================================================================
1410} // end of namespace Decays
1420// ============================================================================
1430
1440/// The file $LOKIGENROOT/LoKi/GenTreesFactory.h
1450
1460namespace Decays
1470{
1480  // ============================================================================
1490  namespace Trees
1500  {
1510    // ============================================================================
1520    /** "Factory" to create the proper Tree from the full description
1530     *   @param tree       (OUTPUT) the constructed tree
1540     *   @param mother     (INPUT)  the mother particle
1550     *   @param oscillated (INPUT)  the oscillation flag
1560     *   @param arrow      (INPUT)  the type of arrow
1570     *   @param daughters  (INPUT)  the list of daughter trees
1580     *   @param inclusive  (INPUT)  the flag for inclusive
1590     *   @param optional   (INPUT)  the list of "optional" particles
1600     *   @return StatusCode
1610     *   @author Vanya BELYAEV Ivan.Belyaev@nikhef.nl
1620     *   @date   2009-05-06
1630     */
1640    StatusCode factory
1650    ( Decays::Trees::Types_<const HepMC::GenParticle*>::Tree&           tree       ,
1660      const Decays::iNode&                                              mother     ,
1670      const Decays::Trees::Oscillation&                                 oscillated ,
1680      const Decays::Trees::Arrow&                                       arrow      ,
1690      const Decays::Trees::Types_<const HepMC::GenParticle*>::SubTrees& daughters  ,
1700      const bool                                                        inclusive  ,
1710      const Decays::Trees::Types_<const HepMC::GenParticle*>::SubTrees& optional   ) ;
1720    // ============================================================================
1730    /** "Factory" to create the proper Tree from "short" descriptor
1740     *   @param tree       (OUTPUT) the constructed tree
1750     *   @param mother     (INPUT)  the mother particle
1760     *   @return status code
1770     *   @author Vanya BELYAEV Ivan.Belyaev@nikhef.nl
1780     *   @date   2009-05-06
1790     */
1800    StatusCode factory
1810    ( Decays::Trees::Types_<const HepMC::GenParticle*>::Tree& tree       ,
1820      const Decays::iNode&                                    mother     ) ;
1830    // ============================================================================
1840  } // end of namespace Decays::Trees::
1850  // ============================================================================
1860} // end of namespace Decays
1870// ============================================================================
1880
1890/// The file $LOKIPHYSROOT/LoKi/TreeFactory.h
1900
1910namespace Decays
1920{
1930  namespace Trees
1940  {
1950    // ============================================================================
1960    /** "Factory" to create the proper Tree from the full description
```

```
1970      *   @param tree       (OUTPUT) the constructed tree
1980      *   @param mother     (INPUT)  the mother particle
1990      *   @param oscillated (INPUT)  the oscillation flag
2000      *   @param arrow      (INPUT)  the type of arrow
2010      *   @param daughters  (INPUT)  the list of daughter trees
2020      *   @param inclusive  (INPUT)  the flag for inclusive
2030      *   @param optional   (INPUT)  the list of "optional" particles
2040      *   @return StatusCode
2050      *   @author Vanya BELYAEV Ivan.Belyaev@nikhef.nl
2060      *   @date   2009-05-06
2070      */
2080     StatusCode factory
2090     ( Decays::Trees::Types_<const LHCb::Particle*>::Tree&           tree       ,
2100       const Decays::iNode&                                         mother     ,
2110       const Decays::Trees::Oscillation&                            oscillated ,
2120       const Decays::Trees::Arrow&                                  arrow      ,
2130       const Decays::Trees::Types_<const LHCb::Particle*>::SubTrees& daughters  ,
2140       const bool                                                   inclusive  ,
2150       const Decays::Trees::Types_<const LHCb::Particle*>::SubTrees& optional   ) ;
2160     // ============================================================================
2170     /** "Factory" to create the proper Tree from "short" descriptor
2180      *   @param tree       (OUTPUT) the constructed tree
2190      *   @param mother     (INPUT)  the mother particle
2200      *   @return status code
2210      *   @author Vanya BELYAEV Ivan.Belyaev@nikhef.nl
2220      *   @date   2009-05-06
2230      */
2240     StatusCode factory
2250     ( Decays::Trees::Types_<const LHCb::Particle*>::Tree& tree       ,
2260       const Decays::iNode&                                mother     ) ;
2270     // ============================================================================
2280   } // end of namespace Decays::Trees
2290   // ============================================================================
2300 } // end of namespace Decays
2310 // ============================================================================
2320 // The END
2330 // ============================================================================
2340
```

How to use the factory?

The file `$LOKICOREROOT/LoKi/Trees.h` defines many useful utilities to obtain the proper types, associated with the decay tree:

```
1000namespace Decays
1010{
1020   // ============================================================================
1030   namespace Trees
1040   {
1050     // ============================================================================
1060     /** @struct Types
1070      *  Helper structure to define the proper types
1080      */
1090     template <class PARTICLE>
1100     struct Types_
1110     {
1120       // ============================================================================
1130       /// the actual particle type
1140       typedef PARTICLE                                    Type     ;
1150       /// the actual particle type
1160       typedef PARTICLE                                    Particle ;
1170       /// the actual type of interface
1180       typedef Decays::iTree_<PARTICLE>                    iTree    ;
1190       /// the actual type of assignable
1200       typedef Decays::Tree_<PARTICLE>                     Tree     ;
```

```
1210       /// the actual type of vectior of assignables
1220       typedef typename Decays::Trees::_Tree_<PARTICLE>::SubTrees SubTrees ;
1230       // the acutal type of "Any"
1240       typedef Decays::Trees::Any_<PARTICLE>                      Any      ;
1250       // the acutal type of "Marked"
1260       typedef Decays::Trees::Marked_<PARTICLE>                   Marked   ;
1270       // ========================================================================
1280     } ;
1290     // ============================================================================
1300   } // end of namespace Decays::Trees
1310   // ================================================================================
1320} // end of namespace Decays
1330// ================================================================================
1340
```

The typical code fragment for using factories is:

```
1000#include "LoKi/Trees.h"
1010#include "LoKi/<.....>Factory.h" // get the corresponding factory
1020...
1030
1040typedef Decays::Trees::Types_<PARTICLE> Types ;
1050
1060// prepare the placeholder for the decay tree:
1070Types::Tree tree = Types::Any() ;
1080
1090// get the oscillation flag (e.g. form the parser):
1100const Decays::Trees::Oscilaltion& oscillated = ... ;
1110
1120// get the arrow type (e.g. form the parser):
1130const Decays::Trees::Arrow& arrow = .... ;
1140
1150// get "inclusive" flag (e.g. form the parser)
1160const bool inclusive = .... ;
1170
1180// get the mother/dead node (e.g. from the parser):
1190const Decays::iNode& mother = .... ;
1200
1210// get the list of child trees (e.g. from parser):
1220const Types::SubTrees& daughters = ... ;
1230
1240// get the list of "optional" children (e.g. from parser):
1250const Types::SubTrees& optional = ... ;
1260
1270// Use the factory:
1280StatusCode sc = Decays::Trees::factory ( tree , mother , oscillated , arrow , daughters , in
1290if ( sc.isFailure() ) {  .. error here ... }
1300
1310// validate the tree:
1320const LHCb::IParticlePropertySvc* service = ... ;
1330sc = tree.validate ( service ) ;
1340if ( sc.isFailure() ) { ... error here ... }
1350
```

-- Vanya BELYAEV - 28 June 2009

This topic: LHCb/FAQ > LoKiNewDecayFinders
Topic revision: r24 - 2018-07-09 - JordanDanielRoth

Copyright &© 2008-2020 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding TWiki? Send feedback