LoKi & Bender
Smart & transparent physics analysis

Vanya Belyaev
CERN & ITEP/Moscow
Outline

• The easy and friendly environment for physics analysis
  • The dream or reality?

• Code complexity, readability, easiness etc is not a property of language
  • C++ itself is not a shit

LoKi

• Set of C++ utilities atop of DaVinci to perform easy (‘1 line’) and readable physics analysis

Bender

• Set of Python’s utilities for interactive physics analysis with full access to LoKi’s functionality
Tool for physics analysis

- Selection/filtering of particles with certain criteria
- Looping over the various combinations
- Creation of composed particles
- Kinematical/topological constraints
- Access to MC truth
- Histograms & N-Tuples

The major design criteria
- Compact & Readable code
  - At most 1 line per each task
- Hide all technical details
- Concentrate on physical contents
- Friendly semantics
Is the goal achievable?

The best example - **KAL** by genius Hartwig Albrecht

- Script-like file
- All technical details are well hidden from end-users
- Transparent physical content of the code
- Looping, histograms, N-tuples, MC truth - at most 1 line!
- Typical analysis program ~ 50-70 lines
- All senior person, including the spokesman successfully participated in physics analysis

```
HYPOTH E+ MU+ PI+ 5 K+ PROTON
IDENT PI+ PI+
IDENT K+ K+
IDENT E+ E+
IDENT PROTON PROTON
IDENT MU+ MU+
SELECT K- pi+
  IF P > 2 THEN
    SAVEMFIT D0 DMASS 0.040 CHI2 4
  ENDIF
ENDSEL
SELECT D0 pi+
  PLOT MASS L 2.0 H 2.1 NB 100 @
    TEXT ' mass D0 pi+ '
ENDSEL
GO 1000
```
Is the goal achievable with OO?

- Majority (but me) is convinced that C++ features (verbosity, static nature etc) do not allow to use it as friendly language for physics analysis

Pattern package by T. Glebe (HERA-B)
- Native C++
- Easy, readable and very efficient

TrackPattern PiMinus = pi_minus.with ( pt > 0.1 & p > 1 );
TrackPattern PiPlus = pi_plus.with ( pt > 0.1 & p > 1 );
TwoProngDecay kShort = K0S.decaysTo ( PiMinus & PiPlus );
kShort.with ( vz > 0 );
kShort.with ( pt > 0.1 );
Try to merge the best ideas: LoKi

- KAL by Hartwig Albrecht
  - ‘1-line’ semantics
  - Predefined variables
- Pattern and GCombiner by Thorsten Glebe
  - Cuts and patterns
- HepChooser and HepCombiner from obsolete CLHEP
  - Combinations, loops
- Loki by Andrei Alexandresku
  - Functions, name and spirit

```c
select ("K-", ID == "K-" && CL > 0.01 && P > 5 * GeV );
select ("Pi+", ID == "pi+" && CL > 0.01 && P > 5 * GeV );

for ( Loop D0 = loop("K- Pi+", "D0" ); D0 ; ++D0 )
{
    if ( P(D0) > 10 * GeV ) { D0->save("D0"); }
}

for ( Loop Dstar = loop("D0 Pi+", "D*+"); Dstar; ++Dstar )
{
    plot ("Mass of D0 pi+", M(Dstar) / GeV, 2.0, 2.1, 100);
}
```
**LoKi: major design ideas**

- Compact, easy to read and transparent code
- Hide all technicalities
- Implement all 'everyday idioms' as 1-line functions
- Locality:
  - Declare, create and use the objects only 'locally'
  - 1 analysis = 1 short file
- High CPU performance
  - Reuse of the most modern C++ techniques
  - Paradigm of templated compile time metaprogramming
- Implement everything as reusable components
  - LoKi functions are compatible with Loki, STL, boost, CLHEP
  - LoKi functions are used with cuts, other functions, histograms, tuples, MC truth, etc
- Weak coupling with concrete Event model, tools, etc
- Extendable
### LoKi versus native DaVinci

<table>
<thead>
<tr>
<th>Selection</th>
<th>SLOC</th>
<th>Person-month</th>
<th>Cost [k$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2DD</td>
<td>2.6 k</td>
<td>6.5</td>
<td>73</td>
</tr>
<tr>
<td>B2HH</td>
<td>362</td>
<td>0.8</td>
<td>9</td>
</tr>
<tr>
<td>Bd2D0Kstar</td>
<td>1.1 k</td>
<td>2.3</td>
<td>30</td>
</tr>
<tr>
<td>Bd2JpsiKstar</td>
<td>1.5 k</td>
<td>3.6</td>
<td>40</td>
</tr>
<tr>
<td>Bd2MuMuKstar</td>
<td>1.4 k</td>
<td>3.4</td>
<td>38</td>
</tr>
<tr>
<td>Bs2DsH</td>
<td>3.2 k</td>
<td>8.0</td>
<td>91</td>
</tr>
<tr>
<td>Bs2MuMu</td>
<td>530</td>
<td>1.2</td>
<td>14</td>
</tr>
<tr>
<td>Bd2JpsiKs</td>
<td>1.0 k</td>
<td>2.3</td>
<td>30</td>
</tr>
<tr>
<td><strong>Bd2KstarGamma</strong></td>
<td><strong>128</strong></td>
<td><strong>0.3</strong></td>
<td><strong>2</strong></td>
</tr>
</tbody>
</table>
LoKi: Status of current version v1r2

- **LoKi** is used by Galina, Andrey, Sergey and Benoit for their studies of radiative and gluon penguins
- Tools/LoKi v1r1 is semi-officially released as part of DaVinci v8r0
- Few released physics (pre)selection packages use LoKi
- Package of examples is provided: LoKiExamples
- Detailed documentation (65 pages) is available
  ~ibelyaev/doc/GaudiDoc/LoKi.ps
**LoKi: selection of particles**

- Simple selection of particles & vertices from TES, LoKi internal storages, already selected particles or any other sources according to kinematical and/or topological criteria

```cpp
select("Kaon" , ID=="K-" || ID=="K+" ) ;

Positive pions with Confidence Level in excess of 1% and $p_T > 100$ MeV/c$^2$
select("Pi+" , ID=="pi+" && CL>0.01 && PT>100*MeV ) ;

Positive muons with $\chi^2_{IP}$ with respect to the primary vertex in excess of 4
const Vertex* pv = ... ;
select("MyMu" , ID=="mu+" && IPCHI2(point(pv))>4) ;
```
LoKi: functions and cuts

Large set (>50) of predefined functions

- Simple properties of particles
  - $P, PT, PX, M, CL, ID, Q, LV01, M12, DMASS, DMCHI2, \ldots$

- Simple properties of Vertices
  - $VCHI2, VTYPE, VX, VZ, VDOF, VPRONGS, VTRACKS, \ldots$

- Topological properties of Particles and Vertices
  - $IP, IPCHI2, VDCHI2, VDTIME, VDSIGN, DDANG, \ldots$

- Operations with Functions - other Functions
  - $+ - * / \ sin \ cos \ tan \ abs \ pow \ min \ max \ \ldots$

Cuts/predicates are formed from functions
LoKi: multiparticle loops

Loops over particle combinations, selects combinations according to kinematical and topological criteria

simple loop over all K^- π+ π+ π- combinations

for (Loop D0 = loop( "K^- pi+ pi+ pi-" , "D0" ) ; D0 ; ++D0 )
{
  if ( PT( D0 ) > 1 * GeV && VCHI2( D0 ) < 49 )
  {
    plot( "K^- pi+ pi+ pi- mass", M(D0)/GeV , 1.5 , 2.0 , 200 );
    Cut dm = abs( DMASS("D0") ) < 30 * MeV;
    if ( dm( D0 ) ) { D0->save("D0"); }
  }
}
LoKi: Histograms

- Histograms are local & booked on-demand
- No need for pre-booking!
- Include variants for effective implicit loops

```cpp
for( Loop D0 = loop( "K- pi+" , "D0") ; D0 ; ++D0 )
{
    plot( "K- pi+ mass", M(D0)/GeV , 1.7 , 2.0 , 150 );
}
```

- Book and fill the histogram

```cpp
plot( loop( "K- pi+"", "D0" ) , "(2)K-pi+ mass" , M12 / GeV ,
1.7 , 2.0 , 150 );
```

- Make a loop, book and fill the histogram

```cpp
plot( select("Kaons", ID == "K-" ) , "PT of kaons ", PT /GeV ,
0 , 5 , 100 );
```

- Select particle, make a loop, book and fill the histogram
LoKi: N-tuples

- N-Tuples are local & booked on-demand
- No need for pre-booking of N-Tuple and its items
- Include variants for effective implicit loops

```cpp
Tuple tuple = ntuple("My N-Tuple for K- pi+ combinations");
for( Loop D0 = loop("K- pi+", "D0"); D0; ++D0 )
{
  tuple -> column("M", M(D0)/GeV);
  tuple -> column("PT", PT(D0)/GeV);
  tuple ->fill("PX, PY, PZ", PX(D0)/GeV, PY(D0)/GeV, PZ(D0)/GeV);
  tuple->write();
}
```

- Book N-tuple
- Fill columns one-by-one
- Fill few columns at once
- Commit N-Tuple row
The simplest basic formal question:

Does this reconstructed Particle originates from this MCParticle?

const Particle*   p = ... ; 
const MCParticle*  mcp = ... ;

MCMatch mcmatch = mctruth();

bool match = mcmatch( p , mcp ) ;
LoKi: MC matching II

*Question 2*

**What MCParticle from the list correspond to this Particle?**

```cpp
const Particle* p = ... ;
MCSEQ mcps = ... ;

MCMatch mcmatch = mctruth();

MCSEQ::iterator mcp =
    mcmatch->match( p , mcps.begin() , mcps.end() ) ;
if ( mcps.end() != mcp )
    {  
        const MCParticle* mc =  *mcp ;
    }
```

*Arbitrary sequence of MCParticle objects*

*Use MC match object*

*MCParticle is found!*
LoKi: MC matching III

**Question 3**
- **What** Particle from the list correspond to **this** MCParticle?

```cpp
SEQ ps = ...;
const MCParticle* mcp = ...;

MCMatch mcmatch = mc_truth();

SEQ::iterator ip =
    mcmatch->match( ps.begin() , ps.end() , mcp );
if ( ps.end() != ip )
{
    const Particle* particle = *ip ;
}
```

*Arbitrary sequence of Particle objects*

*Use MC match object*

*Particle is found!*
LoKi: MC matching IV

• Easy to combine with Olivier Dormond’s beautiful tool

MCMatch finder = mctruth();
MCRange mcD0s = finder->findDecays("D0 -> K- pi+");

Cut mccut = MCTRUTH( mctruth(), mcD0s );
for( Loop D0 = loop( "K- pi+", "D0" ); D0 ; ++D0 )
{
    if( mccut( D0 ) )
    {
        plot("mass of true D0->K- pi+",
             M(D0)/GeV,1.7,2.0,150);
    }
}
LoKi: other utilities

- Event tag collections
  - Almost no difference to Tuples

- Expansion of decay trees (both MC and Reco)
- Extraction of ProtoParticles
- Easy extraction to decay tree products with indices:
  - child ( B0 , 1 )
  - child ( B0 , 2 , 1 )
  - child ( child ( B0 , 1 ) , 4 )

- Other utilities & tools beyond this presentation
LoKi + Python = Bender

- **Python** allows to make the code even more compact and readable
- **Python** allows to keep the code and the options together in one file
  - Improved locality
- **Python** allows to make analysis interactive
  - **Invoke** Bender from Panoramix prompt ?
- **The only one executable for all persons and all their jobs**
  - No private libraries, no compiler, linker etc
- 'Platform independent' (to some extent)
  - Develop and test algorithms on laptop (Win) and then send the script to 'large' center (Linux)
- Each separate analysis - **1 self-contained python file with code and options**
```python
from BenderModule import *

class PhiPhi(Bender):
    " My own analysis algorithm"
    def analyse(self):
        kplus = self.select(tag="K+", cuts=ID="K+")
        kminus = self.select(tag="K-", cuts=ID="K-")
        primaries = self.oselect(tag='FVs', cuts=VTYPE='Vertex_Type.Primary')
        if primaries.empty(): return SUCCESS

        tuple = self.ntuple(name='myTuple')
        phis = self.loop(formula='K+ K-', pid=333)

        # MC truth information
        mc = self.mctruth()
        mcphi0 = mc.finddecay(decay='phi(1020) -> K+ K-')
        mcphi12 = MCUTRUTH(self.mctruth('jshrai'), mcphi0)

        cm = abs(DMASS(phi(1020))) < 10.
        for phi in phis:
            if M12(phi) > 1050: continue
            if not mcphi0: continue
            self.plot(title='phi mass', value=M(phi), low=1000, high=1050)
            tuple.column(name='mass', value=M(phi))
            tuple.column(name='m12', value=M12(phi))
            tuple.column(name='p', value=P(phi))
            tuple.column(name='pt', value=PT(phi))
            tuple.column(name='c11', value=CL(phi(1)))
            tuple.column(name='c12', value=CL(phi(2)))
            tuple.write()
            if cm: phi.save('phi')
```

(1) \( \phi \) selection part of Analysis.py file
Bender: \( B_s \to \phi\phi \) 

(II)

tuple2 = self.ntuple( name='tuple for Bs' )  
allBs = self.loop(formula='\text{phi phi}', pid=531)  
for Bs in allBs:
    if M12(Bs) < 3000: continue
    tuple2.column( name='m12', value=M12(Bs) )
    tuple2.column( name='p', value=P(Bs) )
    tuple2.column( name='pt', value=PT(Bs) )
    tuple2.column( name='lv01', value=LVO1(Bs) )
    tuple2.write()

phis = self.selected('phi')
print 'Number of selected phis is ', phis.size()

return SUCCESS
Bender: $B_s \rightarrow \phi \phi$ (III)

```python
def property(tag):  # temporary trick!
    return Service(tag)

# create my algorithm
myAlg = PhiPhi('MyAnalysis')

# configure my algorithm
p1 = property('MyAnalysis')
p1.OutputLevel = 4
p1.TupleLUN = 'TUPLES'
p1.TupleOffSet = 2000

# configure the desktop
desktop = property('MyAnalysis.PhysDesktop')
desktop.InputLocations = ['/Event/Phys/Photons', '/Event/Phys/Charged']
desktop.OutputLocation = '/Event/Phys/MyAlg'

# initialize the algorithms
myAlg.initialize()

# append it to the list of top level algorithms
g.topAlg = g.topAlg + [ 'MyAnalysis' ]

g.execute(400)
g.exit()
```
• Loki is a god of wit and mischief in Norse mythology

• Loops & Kinematics
LoKi II
LoKi III

28 May'2k+3

Vanya Belyaev CERN & ITEP/Moscow
Bender