Analysis Description Languages for LHC & CutLang
Gokhan UNEL, UC Irvine
Welcome to the LHC analysis jungle

Inclusive analyses with hundreds of selection regions

Overlaps between different analyses?

Multiple analyses exploring similar final states

Is my control region your signal region???

Many alternative definitions for one object

Many variables, ambiguous definitions

...time to get better organized to work more efficiently!
Maximize the scientific Impact of the LHC analyses

- Accessible data
- Accessible simulation
- Accessible results
- Accessible analysis information

Analysis description languages
An Analysis Description Language (ADL) for the LHC is:

- A domain specific language capable of describing the contents of an LHC analysis in a standard and unambiguous way.
  - Customized to express analysis-specific concepts.
- Designed for use by anyone with an interest in, and knowledge of, LHC physics: experimentalists, phenomenologists, other enthusiasts...
- Earlier HEP formats/languages proved successful and useful:
  - SUSY Les Houches Accord
  - Les Houches Event Accord
Principles for an LHC ADL

The principles of an analysis description language were defined in the Les Houches 2015 new physics WG report (arXiv:1605.02684)

Towards an analysis description accord for the LHC


Abstract: We discuss the concept of an “analysis description accord" for LHC analyses, a format capable of describing the contents of an analysis in a standard and unambiguous way. We present the motivation for such an accord, the requirements upon it, and an initial discussion of the merits of several implementation approaches. With this, we hope to initiate a community-wide discussion that will yield, in due course, an actual accord.
By construction, an ADL is not designed to be general purpose; therefore, getting the right scope is key.

The core of ADL for the LHC should include

- simple and composite object definitions (jets, muons, Ws, RPV stops, …)
- event variable definitions ($M_{T2}$, angular variables, BDTs…)
- event selection definitions (signal, control, validation regions, …)

(optional: standard reports, visualizations)

Further operations with selected events (background estimation methods, scale factor derivations, etc.) can vary greatly, and thus may not easily be considered within the ADL scope.
ADLs would help everyone

<table>
<thead>
<tr>
<th>Motivation / use case</th>
<th>Exp</th>
<th>TH/Pheno</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis abstraction, design, implementation</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Analysis communication, clarification, synchronization, visualization</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Analysis review by internal or external referees</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Easier comparison/combination of analyses</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Interpretation studies, analysis reimplementation</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Analysis preservation (ongoing discussions with CERN Analysis Preservation Group)</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Improve our way of thinking about our analyses modelling and structure</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Framework independence highly desirable

Coding analyses in different frameworks takes too much time!

LHC physicist

LHC physics

hard to maintain

ever-changing frameworks

complex software

physics information scattered

everyone writes code differently

takes time to learn
Framework independence highly desirable

Coding analyses in different frameworks takes too much time!

LHC physicist

hard to maintain

A D L

ever-changing frameworks
complex software

physics information scattered

everyone writes code differently
takes time to learn

LHC physics
Features of an ADL for the LHC

Basic requirements:

- Public: Belongs to everyone
- Can describe the complete analysis
- Easily learned
- Demonstrably correct
- Human readable

Desirable features:

- Self-contained
- Domain specific language (not a general purpose language)
- Analysis framework-independent
What could be a good way to systematically organize the components of an analysis?

Use “blocks” as in SLHA or LHE.
A Proposal for a Les Houches Analysis Description Accord


Abstract: We present the first draft of a proposal for “a Les Houches Analysis Description Accord" for LHC analyses, a formalism that is capable of describing the contents of an analysis in a standard and unambiguous way independent of any computing framework. This proposal serves as a starting point for discussions among LHC physicists towards an actual analysis description accord for use by the LHC community.

CutLang: A particle physics ADL and runtime interpreter

S. Sekmen, G. Ünel

Abstract: This note introduces CutLang, a domain specific language that aims to provide a clear, human readable way to define analyses in high energy particle physics (HEP) along with an interpretation framework of that language. A proof of principle (PoP) implementation of the CutLang interpreter, achieved using C++ as a layer over the CERN data analysis framework ROOT, is presently available. This PoP implementation permits writing HEP analyses in an unobfuscated manner, as a set of commands in human readable text files, which are interpreted by the framework at runtime. We describe the main features of CutLang and illustrate its usage with two analysis examples. Initial experience with CutLang has shown that a just-in-time interpretation of a human readable HEP specific language is a practical alternative to analysis writing using compiled languages such as C++.

→ ADL design driven by runtime interpretability.

CutLang and LHADA follow same principles but slightly differ in syntax.
The ADL consists of

- a plain text file describing the analysis using a HEP specific language with syntax rules that include standard mathematical and logical operations and 4-vector algebra.
- a library of self-contained functions encapsulating variables that are non-trivial to express with the ADL syntax.

The ADL is analysis framework independent so that it can offer a standard input to analysis frameworks, just like an SLHA file offers standard input to SUSY calculators.

Both ADL files and external functions can be eventually hosted at central databases for LHC analyses. Discussions ongoing with CERN Analysis Preservation Group.
Examples: object definitions

LHADA ADL style

# AK4 jets
object AK4jets
take Jet
select pt > 30
select |eta| < 2.4

# b-tagged jets - loose
object bjetsLoose
take AK4jets
select btagDeepB > 0.152

# b-tagged jets - medium
object bjetsMedium
take AK4jets
select btagDeepB > 0.4941

CutLang style

# AK4 jets
object AK4jets using JET
select {JET_}Pt > 30
select abs({JET_}Eta) < 2.4

# b-tagged jets - loose
object bjetsLoose : AK4jets
select {AK4jets}_btagDeepB > 0.152

# b-tagged jets - medium
object bjetsMedium : AK4jets
select {AK4jets}_btagDeepB > 0.4941


LHADA style full implementation link
CutLang style full implementation link
Examples: variable definitions

LHADA style

define MR = fMR(megajets)
define Rsq = sqrt(fMTR(megajets, met) / MR)
define dphimegajets = dPhi(megajets[0], megajets[1])
define METl = met + leptonsVeto[0]
define Rsqm = sqrt(fMTR(megajets, METl) / MR)
define MT = fMT(leptonsVeto[0], met)
define Mll = fMll(leptonsTight[0], leptonsTight[1])

CutLang style

define MR = fMR(megajets)
define Rsq = sqrt(fMTR(megajets, MET) / MR)
define dphimegajets = dPhi(megajets[0], megajets[1])
define METLVm = METLV[0] + muonsVeto[0]
define Rsqm = sqrt(fMTR(megajets, METLVm) / MR)
define MTm = sqrt(2*{muonsVeto[0]}Pt*MET*(1-cos({METLV[0]}Phi - {muonsVeto[0]}Phi)))
define Mll = { muonsTight[0] muonsTight[1] }m
Examples: event selection

**LHADA style**

# preselection region
region preselection
    select size(AK4jets) >= 3
    select size(AK8jets) >= 1
    select MR > 800
    select Rsq > 0.08

# control region for tt+jets
region ttjetsCR
    select preselection
    select size(leptonsVeto) == 1
    select size(WjetsMasstag) >= 1
    select dphimegajets < 2.8
    select MT [] 100
    # or select fMT(leptonsVeto[0], met) [] 30 100
    # or select 30 < MT < 100
    select size(bjetsLoose) == 0

**CutLang style**

# p# preselection region
region preselection
    select ALL # count all events
    select Size(AK4jets) >= 3
    select Size(AK8jets) >= 1
    select Size(megajets) == 2
    select MR > 800
    select Rsq > 0.08

# control region for W+jets
region WjetsCR
    preselection
    select Size(muonsVeto)+Size(electronsVeto) == 1
    select Size(WjetsMasstag) >= 1
    select dphimegajets < 2.8
    select Size(muonsVeto) == 1 ? MTm [] 30 100
    # select MT > 300
    # MTm [30 100]
    # MT [30 100]
    # MTm [30 100]
    select Size(bjetsLoose) == 0
## ADL block types and keywords

<table>
<thead>
<tr>
<th>Block Type</th>
<th>LHADA $\rightarrow$ ADL</th>
<th>CutLang $\rightarrow$ ADL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object definition blocks</strong></td>
<td>object</td>
<td>obj / object</td>
</tr>
<tr>
<td><strong>Event selection blocks</strong></td>
<td>region</td>
<td>algo / region</td>
</tr>
<tr>
<td><strong>Analysis information</strong></td>
<td>info</td>
<td>info</td>
</tr>
<tr>
<td><strong>Tables of results, etc.</strong></td>
<td>table</td>
<td>—</td>
</tr>
<tr>
<td><strong>Define variables, constants</strong></td>
<td>define</td>
<td>def / define</td>
</tr>
<tr>
<td><strong>Select object or event</strong></td>
<td>select</td>
<td>select / cmd</td>
</tr>
<tr>
<td><strong>Reject object or event</strong></td>
<td>reject</td>
<td>—</td>
</tr>
<tr>
<td><strong>Define the mother object</strong></td>
<td>take</td>
<td>: / take / using</td>
</tr>
<tr>
<td><strong>Define histograms</strong></td>
<td>—</td>
<td>histo</td>
</tr>
<tr>
<td><strong>Applies object/event weights</strong></td>
<td>weight</td>
<td>—</td>
</tr>
<tr>
<td><strong>Bins events in regions</strong></td>
<td>bin</td>
<td>—</td>
</tr>
</tbody>
</table>

**Green:** Implemented in (some) parser/interpreter tools  
**Black:** Implementation in progress
## ADL operators

<table>
<thead>
<tr>
<th></th>
<th>LHADA $\rightarrow$ ADL</th>
<th>CutLang $\rightarrow$ ADL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparison operators</strong></td>
<td>$&gt; \ &lt; \ =&gt; \ &lt;= \ ==&gt; \ &lt;=\ $ (include) $\overline{[]}$ (exclude)</td>
<td>$&gt; \ &lt; \ =&gt; \ &lt;= \ ==&gt; \ &lt;=\ $ (include) $\overline{[]}$ (exclude)</td>
</tr>
<tr>
<td><strong>Mathematical operators</strong></td>
<td>+ - * / ^</td>
<td>+ - * / ^</td>
</tr>
<tr>
<td><strong>Logical operators</strong></td>
<td>and or</td>
<td>AND/&amp;&amp; OR/</td>
</tr>
<tr>
<td><strong>Ternary operator</strong></td>
<td>condition ? true-case : false-case</td>
<td>condition ? truecase : falsecase</td>
</tr>
<tr>
<td><strong>Optimization operators</strong></td>
<td>$-$</td>
<td>$~=$ (closest to) $!=$ (furthest from) (optimal particle sets are assigned negative indices)</td>
</tr>
<tr>
<td><strong>Lorentz vector addition</strong></td>
<td>LV1 + LV2</td>
<td>LV1 LV2 / LV1 + LV2</td>
</tr>
</tbody>
</table>

**Green:** Implemented in (some) parser/interpreter tools

**Black:** Implementation in progress
ADL functions

Standard/internal functions: Sufficiently generic math and HEP operations would be a part of the language and any tool that interprets it

• Math functions: $\text{abs}()$, $\text{sin}()$, $\text{cos}()$, $\text{tan}()$, $\text{log}()$, $\text{sqrt}()$, … (mostly implemented in CutLang)
• Reducers: $\text{size}()$, $\text{sum}()$, $\text{min}()$, $\text{max}()$, $\text{any}()$, $\text{all}()$, …
• HEP-specific functions: $\text{dR}()$, $\text{dphi}()$, $\text{m}()$, …. (exist in CutLang)
  • CutLang treats object attributes like pT, eta, … as functions

External/user functions: Variables that cannot be expressed using the available operators or standard functions would be encapsulated in self-contained functions that would be addressed from the ADL file

• Variables with non-trivial algorithms: MT2, aplanarity, razor variables, …
• Non-analytic variables: Object/trigger efficiencies, variables computed with MVAs, …

Green: Implemented in CutLang and partially in other tools,
Black: Implementation in progress
adl2tnm (Harrison Prosper)

- Python script converts ADL to c++ code.
- c++ code executed within the generic TNM (TheNtupleMaker) generic ntuple analysis framework. Only depends on ROOT.
- Can work with any simple ntuple format. Automatically incorporates the input event format into the c++ code:
  ADL + input ROOT files $\rightarrow$ adl2tnm.py $\rightarrow$ c++ analysis code
- Assumes that a standard extensible type is available to model all analysis objects. Uses adapters to translate input to standard types.
- Can be used for experimental or phenomenological analyses.
- Upcoming version will include formal grammar building and parsing.

GitHub link: https://github.com/hbprosper/adl2tnm
Transpilers for LHADA style ADL - II

**lhada2tivet** (Philippe Gras)
- Python script converts LHADA to **c++ code for Rivet**.
- Particles and jets are implemented using **Rivet-specific truth level objects**. Smearing added in Rivet.
- For phenomenological analyses.

GitHub link: https://github.com/lhada-hep/lhada/tree/master/lhada2rivet.d

**lhada2checkmate** (Daniel Dercks)
- Python script converts from early LHADA to **CheckMate c++ code**.
- Works with **Delphes** objects
- Tested a simple version of **automatic function download**, and confirmed feasibility of a function database for the future.
- For phenomenological analyses
CutLang runtime interpreter & framework
GitHub link: https://github.com/unelg/CutLang

CutLang runtime interpreter:
• No compilation. Directly runs on the ADL file.
  • ADL: [initializations] [definitions] [objects] [definitions] commands
• Written in C++, works in any modern Unix environment.
• Based on ROOT classes for Lorentz vector operations and histograms
• ADL parsing by Lex & Yacc: relies on automatically generated dictionaries and grammar.

CutLang framework: CutLang interpreter + tools and facilities
• Reads events from ROOT files, from multiple input formats like Delphes, ATLAS & CMS open data, LVL0, CMSnanoAOD, FCC. More can be easily added.
• All event types converted into predefined particle object types.
• Includes many internal functions.
• Output in ROOT files. Analysis algorithms, cutflows and histograms for each region in a separate directory.
particle notation

• On the blackboard, we write
  • When you type it in latex it is \textit{jet}_1
  • CL understands \textit{particleName\_index} notation:

<table>
<thead>
<tr>
<th>Highest Pt object</th>
<th>Second Highest Pt object</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELE_0</td>
<td>ELE_1</td>
</tr>
<tr>
<td>MUO_0</td>
<td>MUO_1</td>
</tr>
</tbody>
</table>

• On the computer, we write
  • CL understands \textit{particleName}[index] notation:
functions & attributes

- Is pseudo rapidity or transverse momentum a property of a particle? of the addition of many particles? is it an attribute? is it a function?

- DO I CARE? no.
  - I only care about the result of my analysis

- However, when I speak or write I might say either of
  - “the mass of a particle set”   \( m(\ ) \)
  - “the particle set’s mass”            \( \{\ }m \)

- CL understands both notations

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Operator</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass of</td>
<td>( m(\ ) )</td>
<td>( {\ }m )</td>
</tr>
<tr>
<td>Charge of</td>
<td>( q(\ ) )</td>
<td>( {\ }q )</td>
</tr>
<tr>
<td>Phi of</td>
<td>( \phi(\ ) )</td>
<td>( {\ }\phi )</td>
</tr>
<tr>
<td>Eta of</td>
<td>( \eta(\ ) )</td>
<td>( {\ }\eta )</td>
</tr>
<tr>
<td>Absolute value of Eta of</td>
<td>( \text{Abs}\eta(\ ) )</td>
<td>( {\ }\text{Abs}\eta )</td>
</tr>
<tr>
<td>Pt of</td>
<td>( \text{Pt}(\ ) )</td>
<td>( {\ }\text{Pt} )</td>
</tr>
</tbody>
</table>
A very simple example

- Reconstruct Z from 2 electrons

- the Z candidate should be neutral (q=0)

\[ Z \rightarrow \ell\ell \quad \ell = e, \mu \]

### User’s ADL File

```adl
region test
select ALL # to count all events
select Size (ELE) >= 2 # events with 2 or more electrons
histo h1mReco, "Z candidate mass (GeV)", 100, 0, 200, {ELE_0 ELE_1}m
select {ELE[0] ELE[1]}q == 0 # Z is neutral
histo h2mReco, "Z candidate mass (GeV)", 100, 0, 200, {ELE_0 ELE_1}m
```

### CL Output

```
test Based on 125000 events:
  ALL :  1 +-  0 evt: 125000
Size (ELE) >= 2 :  0.284 +- 0.00128 evt: 35501
[Histo] Z candidate mass (GeV) :  1 +-  0 evt: 35501
{ELE[0] ELE[1]}q == 0 :  0.9595 +- 0.00105 evt: 34063
[Histo] Z candidate mass (GeV) :  1 +-  0 evt: 34063
--> Overall efficiency = 27.3 % +- 0.126 %
```
A very simple example

• Reconstruct Z from 2 electrons
  \( Z \rightarrow \ell\ell \quad \ell = e, \mu \)

• the Z candidate should be neutral (q=0)

region test
select ALL # to count all events
select Size (ELE) >= 2 # events with 2 or more electrons
histo h1mReco, "Z candidate mass (GeV)", 100, 0, 200, {ELE_0 ELE_1}m
select {ELE[0] ELE[1] }q == 0 # Z is neutral
histo h2mReco, "Z candidate mass (GeV)", 100, 0, 200, {ELE_0 ELE_1}m

CL output

<table>
<thead>
<tr>
<th>test</th>
<th>Based on 125000 events:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>1 ++ 0 evt: 125000</td>
</tr>
<tr>
<td>Size (ELE) &gt;= 2</td>
<td>0.284 ++ 0.00128 evt: 35501</td>
</tr>
<tr>
<td>[Histo] Z candidate mass (GeV)</td>
<td>1 ++ 0 evt: 35501</td>
</tr>
<tr>
<td>{ELE[0] ELE[1] }q == 0</td>
<td>0.9595 ++ 0.00105 evt: 34063</td>
</tr>
<tr>
<td>[Histo] Z candidate mass (GeV)</td>
<td>1 ++ 0 evt: 34063</td>
</tr>
</tbody>
</table>

--> Overall efficiency = 27.3 % ++ 0.126 %
A very simple example

- introducing definitions

\[ Z \rightarrow \ell\ell \quad \ell = e, \mu \]

user's ADL file

```plaintext
define Zreco : ELE[0] ELE[1]
region    test
select    ALL    # to count all events
select    Size (ELE) >= 2  # events with 2 or more electrons
histo     h1mReco, "Z candidate mass (GeV)", 100, 0, 200, {Zreco}m
select    {Zreco}q == 0    # Z is neutral
histo     h2mReco, "Z candidate mass (GeV)", 100, 0, 200, m(Zreco)
```

CL output

```
    test Based on 125000 events:
       ALL :  1 ++  0 evt:  125000
       Size (ELE) >= 2 :  0.284 ++  0.00128 evt:  35501
   [Histo] Z candidate mass (GeV) :  1 ++  0 evt:  35501
       {Zreco}q == 0 :  0.9595 ++  0.00105 evt:  34063
   [Histo] Z candidate mass (GeV) :  1 ++  0 evt:  34063
    -- Overall efficiency =  27.3 % ++  0.126 %
```

Are these electrons inside the inner tracker?
# A simple example

- introducing derived objects

- Electron —> goodElectron

```plaintext
define Zreco : ELE[0] ELE[1]

object goodEle : ELE
    select  Pt(ELE_)    >  10
    select  abs(ELE_.Eta) <  2.4
    select  {ELE_.AbsEta}[ 1.442 1.556

define goodZreco : goodEle[0] goodEle[1]

region    test
    select  ALL                        # to count all events
    select  Size(ELE) >= 2            # events with 2 or more electrons
    select  Size(goodEle) >= 2        # events with 2 or more electrons
h histo   h1mReco, "Z candidate mass (GeV)", 100, 0, 200, {Zreco}m
h histo   h1mgoodReco, "Z candidate mass (GeV)", 100, 0, 200, {goodZreco}m
    select  {Zreco}q == 0            # Z is neutral
    select  {goodZreco}q == 0        # Z is neutral
h histo   h2mReco, "Z candidate mass (GeV)", 100, 0, 200, m(Zreco)
h histo   h2mgoodReco, "Z candidate mass (GeV)", 100, 0, 200, m(goodZreco)
```

\[
Z \rightarrow \ell\ell \quad \ell = e, \mu
\]
A simple example

• introducing multiple regions or algorithms

• A user defined region can contain another one
  • e.g. SignalRegion containing preselection

```plaintext
define Zreco : ELE[0] ELE[1]

object goodEle : ELE
    select Pt(ELE_) > 10
    select {ELE_}AbsEta < 2.4
    select {ELE_}AbsEta ][ 1.442 1.556

define goodZreco : goodEle[0] goodEle[1]

algo preselection
    select ALL # to count all events
    select Size(ELE) >= 2 # events with 2 or more electrons

algo testA

# histo h1mReco, "Z candidate mass (GeV)", 100, 0, 200, {Zreco}m
select {Zreco}q == 0 # Z is neutral
histo h2mReco, "Z candidate mass (GeV)", 100, 0, 200, m(Zreco)

algo testB

select Size(goodEle) >= 2 # events with 2 or more electrons
# histo h1mgoodReco, "Z candidate mass (GeV)", 100, 0, 200, {goodZreco}m
select {goodZreco}q == 0 # Z is neutral
histo h2mgoodReco, "Z candidate mass (GeV)", 100, 0, 200, m(goodZreco)
```

\[ Z \rightarrow \ell\ell \quad \ell = e, \mu \]
All regions are processed in parallel and saved as TDirectories in the output ROOT file.
A search example $Z \rightarrow \ell\ell \quad \ell = e, \mu$

- **Introducing optimizers**
  - if there are more than 2 electrons, search all possible combinations to find the “best” candidate
  - use negative indices to defer the identification

```plaintext
define Zreco : ELE[0] ELE[1]

object goodEle : ELE
    select Pt(ELE_) > 10
    select {ELE_}AbsEta < 2.4
    select {ELE_}AbsEta ][ 1.442 1.556

define goodZreco : goodEle[-1] goodEle[-2]

algo BestZ
    select ALL # to count all events
    select Size(goodEle) >= 2 # events with 2 or more electrons
    select {goodZreco}m =~ 91.2 # find the pair yielding mass closest to Z
    select {goodZreco}q == 0 # Z is neutral
    histo hZRecoB, "Z candidate mass (GeV)", 100, 0, 200, m(goodZreco)
```

Negative indices are to be determined at run time, using a criterion, such as: $\sim$
A search example $Z \rightarrow \ell\ell$ $\ell = e, \mu$

- Taking a short cut
  - $e_1 + e_2 = e_2 + e_1 \rightarrow$ same $Z$, no need to calculate both
  - repeating the same negative index (-1) tells CutLang to compute only one
    - compute time reduced by 50%

```python
define Zreco : ELE[0] ELE[1]

object goodEle : ELE
  select Pt(ELE_) > 10
  select {ELE_}.AbsEta < 2.4
  select {ELE_}.AbsEta ][ 1.442 1.556

define goodZreco : goodEle[-1] goodEle[-1]

algo BestZ
  select ALL # to count all events
  select Size(goodEle) >= 2 # events with 2 or more electrons
  select {goodZreco}m ~= 91.2 # find the pair yielding mass closest to $Z$
  select {goodZreco}q == 0 # $Z$ is neutral
  histo hZRecoB, "Z candidate mass (GeV)", 100, 0, 200, m(goodZreco)
```

if we have 3 electrons in an event

<table>
<thead>
<tr>
<th>e1</th>
<th>e2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
User functions

• User defined selection functions are somewhat difficult to incorporate into an interpreter

• Currently we define a user function type and compile it in.
  • CLv2 will provide the means to do this automatically
  • Currently Razor functions are pre-integrated:

```
std::vector<TLorentzVector> fmegajets(std::vector<TLorentzVector> myjets);
double fMR(std::vector<TLorentzVector> j);
double fMTR(std::vector<TLorentzVector> j, TVector2 amet);
double fMTR2(std::vector<TLorentzVector> j, TLorentzVector amet);
```

• Simple functions can be interpreted using CL math functions

```
return sqrt( 2 * lepton.Pt() * pfmet.Pt() * ( 1 - cos( pfmet.Phi() - lepton.Phi() ) ));
```

define MTe : sqrt( 2*{electronsVeto_0}Pt *MET*(1-cos( {METLV_0}Phi - {electronsVeto_0}Phi )))
define MTm : sqrt( 2*{muonsVeto_0}Pt *MET*(1-cos( {METLV_0}Phi - {muonsVeto_0}Phi )))
define mZ : 91.187
To conclude

• An ADL would greatly facilitate analyses for the whole LHC community. First target is the BSM studies type.

• Several prototypes have proven the feasibility of ADLs.

• CutLang is an ADL interpreter with additional features

• you can test is on JuPyter

• Work in progress. Still many intriguing problems to solve! New Gitter forum open to all for discussions: https://gitter.im/HSF/ADL

• This is a community effort. Please join!
Recent workshop to seriously start community-wide discussions.

Participation by experimentalists, phenomenologists, computer scientists.

Learned about other ADL efforts:
- Query ADLs (G. Watts)
- YAML as ADL (B. Krikkler)
- NAIL (A. Rizzi)
- TTreeFormula / RDataFrame (P. Canal)
- AEACUS & RHADAMANTUS (J. Walker - talk in this session)

Extensive discussions towards a unified ADL. Extensive notes and video recordings on indico:
https://indico.cern.ch/event/769263/
thank you for your attention

backup slides

- reference guide
- ttbar reconstruction
- example analyses
- speed issues
The Objects

<table>
<thead>
<tr>
<th>Name</th>
<th>Keyword</th>
<th>Highest Pt object</th>
<th>Second Highest Pt object</th>
<th>$j+1^{th}$Highest Pt object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electron</td>
<td>ELE</td>
<td>ELE_0</td>
<td>ELE_1</td>
<td>ELE_j</td>
</tr>
<tr>
<td>Muon</td>
<td>MUO</td>
<td>MUO_0</td>
<td>MUO_1</td>
<td>MUO_j</td>
</tr>
<tr>
<td>Tau</td>
<td>TAU</td>
<td>TAU_0</td>
<td>TAU_1</td>
<td>TAU_j</td>
</tr>
<tr>
<td>Lepton</td>
<td>LEP</td>
<td>LEP_0</td>
<td>LEP_1</td>
<td>LEP_j</td>
</tr>
<tr>
<td>Photon</td>
<td>PHO</td>
<td>PHO_0</td>
<td>PHO_1</td>
<td>PHO_j</td>
</tr>
<tr>
<td>Jet</td>
<td>JET</td>
<td>JET_0</td>
<td>JET_1</td>
<td>JET_j</td>
</tr>
<tr>
<td>Fat Jet</td>
<td>FJET</td>
<td>FJET_0</td>
<td>FJET_1</td>
<td>FJET_j</td>
</tr>
<tr>
<td>b-tagged Jet</td>
<td>BJET</td>
<td>BJET_0</td>
<td>BJET_1</td>
<td>BJET_j</td>
</tr>
<tr>
<td>light Jet</td>
<td>QGJET</td>
<td>QGJET_0</td>
<td>QGJET_1</td>
<td>QGJET_j</td>
</tr>
<tr>
<td>neutrino</td>
<td>METV</td>
<td>METV_0</td>
<td>METV_1</td>
<td>METV_j</td>
</tr>
</tbody>
</table>

Functions

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of</td>
<td>Size( )</td>
</tr>
<tr>
<td>tangent</td>
<td>tan()</td>
</tr>
<tr>
<td>sinus</td>
<td>sin()</td>
</tr>
<tr>
<td>cosinus</td>
<td>cos()</td>
</tr>
<tr>
<td>absolute value</td>
<td>abs()</td>
</tr>
<tr>
<td>square root</td>
<td>sqrt()</td>
</tr>
<tr>
<td>in the interval</td>
<td>[ ]</td>
</tr>
<tr>
<td>not in the interval</td>
<td>![ ]</td>
</tr>
<tr>
<td>as close as possible</td>
<td>~=</td>
</tr>
<tr>
<td>as far away as possible</td>
<td>!=</td>
</tr>
<tr>
<td>usual meaning</td>
<td>+/*</td>
</tr>
<tr>
<td>to the power</td>
<td>^</td>
</tr>
</tbody>
</table>

Meaning | Operator | Operator
---|---------|----------------|
Mass of | m( )    | { }m
Charge of | q( )    | { }q
Phi of | Phi( )  | { }Phi
Eta of | Eta( )  | { }Eta
Absolute value of Eta of | AbsEta( ) | { }AbsEta
Pt of | Pt( )   | { }Pt
Pz of | Pz( )   | { }Pz
Energy of | E( )    | { }E
Momentum of | P( )    | { }P
Angular distance between | dR( )    | { }dR
Phi difference between | dPhi( )  | { }dPhi
Eta difference between | dEta( )  | { }dEta

The ternary function in C notation

```
false_part : true_part ? ternary_expression
```
$t\bar{t}$ Reconstruction example
Reconstruction example

$tt \rightarrow \bar{t} t \rightarrow Wb \rightarrow jjj_b$
There are 6 jets in the event of which 2 can be b-tagged

+ LOTS of other jets from spectator quarks and QCD effects
There are 6 jets in the event of which 2 can be b-tagged
+ LOTS of other jets from spectator quarks and QCD effects

Which one is which?
Reconstruction example

There are 6 jets in the event of which 2 can be b-tagged
+ LOTS of other jets from spectator quarks and QCD effects

Which one is which?

with the $\chi^2$ defined as:

$$\chi^2 = \frac{(m_{b,1j_1j_2} - m_{b,2j_3j_4})^2}{\sigma^2_{\Delta m_{b,j}}^2} + \frac{(m_{j_1j_2} - m_{W}^{MC})^2}{\sigma^2_{m_{W}^{MC}}^2} + \frac{(m_{j_3j_4} - m_{W}^{MC})^2}{\sigma^2_{m_{W}^{MC}}^2}.$$
ttbar Reconstruction example

```plaintext
define WH1 : JET[-1] JET[-1]
define WH2 : JET[-3] JET[-3]
### chi2 for W finder
define Wchi2 : (((WH1)m - 80.4)/2.1)^2 + (((WH2)m - 80.4)/2.1)^2

### top quarks without b tagging
define Top1 : WH1 JET[-2]
define Top2 : WH1 JET[-4]
define mTop1 : m(Top1)
define mTop2 : m(Top2)
### chi2 for top finder
define topchi2 : ((mTop1 - mTop2)/4.2)^2

algo besttop
  select ALL # to count all events
  select Size(JET) >= 6 # at least 6 jets
  select MET < 100 # no large MET
  select Wchi2 + topchi2 ~= 0 # find the tops and ws
histo hmWH1, "Hadronic W reco (GeV)", 50, 50, 150, m(WH1)
histo hmWH2, "Hadronic W reco (GeV)", 50, 50, 150, m(WH2)
histo hmTop1, "Hadronic top reco (GeV)", 70, 0, 700, mTop1
histo hmTop2, "Hadronic top reco (GeV)", 70, 0, 700, mTop2
```

with the $\chi^2$ defined as:

$$
\chi^2 = \frac{(m_{b_1j_1j_2} - m_{b_2j_3j_4})^2}{\sigma_{\Delta m_{bJJ}}^2} + \frac{(m_{j_1j_2} - m_W^{MC})^2}{\sigma_{m_W^{MC}}^2} + \frac{(m_{j_3j_4} - m_W^{MC})^2}{\sigma_{m_W^{MC}}^2}
$$
“Hadronic W reco (GeV)”

```
Entries  4800
Mean    81.71
Std Dev 11.91
```

“Hadronic top reco (GeV)”

```
Entries  4800
Mean    198.5
Std Dev 81.32
```

reconstructed W bosons

reconstructed top quarks
Razor boost example 1/2
### Compatibility

<table>
<thead>
<tr>
<th>Description</th>
<th>#evt</th>
<th>tot.eff</th>
<th>rel.eff</th>
<th>#evt</th>
<th>tot.eff</th>
<th>rel.eff</th>
<th>tot.eff</th>
</tr>
</thead>
<tbody>
<tr>
<td>2jl cut-flow</td>
<td>31250</td>
<td>1</td>
<td>-</td>
<td>31250</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Pre-sel+MET+pT1</td>
<td>28592</td>
<td>0.91</td>
<td>0.91</td>
<td>28626</td>
<td>0.92</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>Njet</td>
<td>28592</td>
<td>0.91</td>
<td>1</td>
<td>28625</td>
<td>0.92</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Dphi_min(j,MET)</td>
<td>17297</td>
<td>0.55</td>
<td>0.6</td>
<td>17301</td>
<td>0.55</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>pT2</td>
<td>17067</td>
<td>0.55</td>
<td>0.99</td>
<td>17042</td>
<td>0.55</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>MET/sqrtHT</td>
<td>8900</td>
<td>0.28</td>
<td>0.52</td>
<td>8898</td>
<td>0.28</td>
<td>0.52</td>
<td></td>
</tr>
<tr>
<td>m_eff(incl)</td>
<td>8896</td>
<td>0.28</td>
<td>1</td>
<td>8897</td>
<td>0.28</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>#evt</th>
<th>tot.eff</th>
<th>rel.eff</th>
<th>#evt</th>
<th>tot.eff</th>
<th>rel.eff</th>
<th>tot.eff</th>
</tr>
</thead>
<tbody>
<tr>
<td>2jm cut-flow</td>
<td>31250</td>
<td>1</td>
<td>-</td>
<td>32150</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Pre-sel+MET+pT1</td>
<td>28472</td>
<td>0.91</td>
<td>0.91</td>
<td>28478</td>
<td>0.91</td>
<td>0.91</td>
<td>0.91</td>
</tr>
<tr>
<td>Njet</td>
<td>28472</td>
<td>0.91</td>
<td>1</td>
<td>28477</td>
<td>0.91</td>
<td>1</td>
<td>0.91</td>
</tr>
<tr>
<td>Dphi_min(j,MET)</td>
<td>22950</td>
<td>0.73</td>
<td>0.81</td>
<td>22889</td>
<td>0.73</td>
<td>0.8</td>
<td>0.73</td>
</tr>
<tr>
<td>pT2</td>
<td>22950</td>
<td>0.73</td>
<td>1</td>
<td>22889</td>
<td>0.73</td>
<td>1</td>
<td>0.73</td>
</tr>
<tr>
<td>MET/sqrtHT</td>
<td>10730</td>
<td>0.34</td>
<td>0.47</td>
<td>10710</td>
<td>0.34</td>
<td>0.47</td>
<td>0.33</td>
</tr>
<tr>
<td>m_eff(incl)</td>
<td>10630</td>
<td>0.34</td>
<td>0.99</td>
<td>10609</td>
<td>0.34</td>
<td>0.99</td>
<td>0.32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>#evt</th>
<th>tot.eff</th>
<th>rel.eff</th>
<th>#evt</th>
<th>tot.eff</th>
<th>rel.eff</th>
<th>tot.eff</th>
</tr>
</thead>
<tbody>
<tr>
<td>2jt cut-flow</td>
<td>31250</td>
<td>1</td>
<td>-</td>
<td>31250</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Pre-sel+MET+pT1</td>
<td>28592</td>
<td>0.91</td>
<td>0.91</td>
<td>28626</td>
<td>0.92</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>Njet</td>
<td>28592</td>
<td>0.91</td>
<td>1</td>
<td>28625</td>
<td>0.92</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Dphi_min(j,MET)</td>
<td>17297</td>
<td>0.55</td>
<td>0.6</td>
<td>17301</td>
<td>0.55</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>pT2</td>
<td>17067</td>
<td>0.55</td>
<td>0.99</td>
<td>17042</td>
<td>0.55</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>MET/sqrtHT</td>
<td>5083</td>
<td>0.16</td>
<td>0.3</td>
<td>5098</td>
<td>0.16</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Pass m_eff(incl)</td>
<td>4861</td>
<td>0.16</td>
<td>0.96</td>
<td>4889</td>
<td>0.16</td>
<td>0.96</td>
<td></td>
</tr>
</tbody>
</table>

### CutLang

<table>
<thead>
<tr>
<th>Description</th>
<th>#evt</th>
<th>total eff.</th>
<th>rel. eff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>31250</td>
<td>1.000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>28431</td>
<td>0.91</td>
<td>0.91</td>
<td></td>
</tr>
<tr>
<td>28430</td>
<td>0.91</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>16661</td>
<td>0.53</td>
<td>0.59</td>
<td></td>
</tr>
<tr>
<td>16381</td>
<td>0.52</td>
<td>0.98</td>
<td></td>
</tr>
<tr>
<td>8159</td>
<td>0.26</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>8156</td>
<td>0.26</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>31250</td>
<td>1.000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>28301</td>
<td>0.91</td>
<td>0.91</td>
<td></td>
</tr>
<tr>
<td>28300</td>
<td>0.91</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>22441</td>
<td>0.72</td>
<td>0.79</td>
<td></td>
</tr>
<tr>
<td>22441</td>
<td>0.72</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>10043</td>
<td>0.32</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>9896</td>
<td>0.32</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>31250</td>
<td>1.000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>28431</td>
<td>0.91</td>
<td>0.91</td>
<td></td>
</tr>
<tr>
<td>28430</td>
<td>0.91</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>16661</td>
<td>0.53</td>
<td>0.59</td>
<td></td>
</tr>
<tr>
<td>16381</td>
<td>0.52</td>
<td>0.98</td>
<td></td>
</tr>
<tr>
<td>4375</td>
<td>0.14</td>
<td>0.27</td>
<td></td>
</tr>
<tr>
<td>4132</td>
<td>0.13</td>
<td>0.94</td>
<td></td>
</tr>
</tbody>
</table>
Debugging & speeding

For example, the one step and two step top quark reconstructions requiring one line and two lines to implement in the CutLang language take about 40 to 70 lines of standard analysis code in C++.