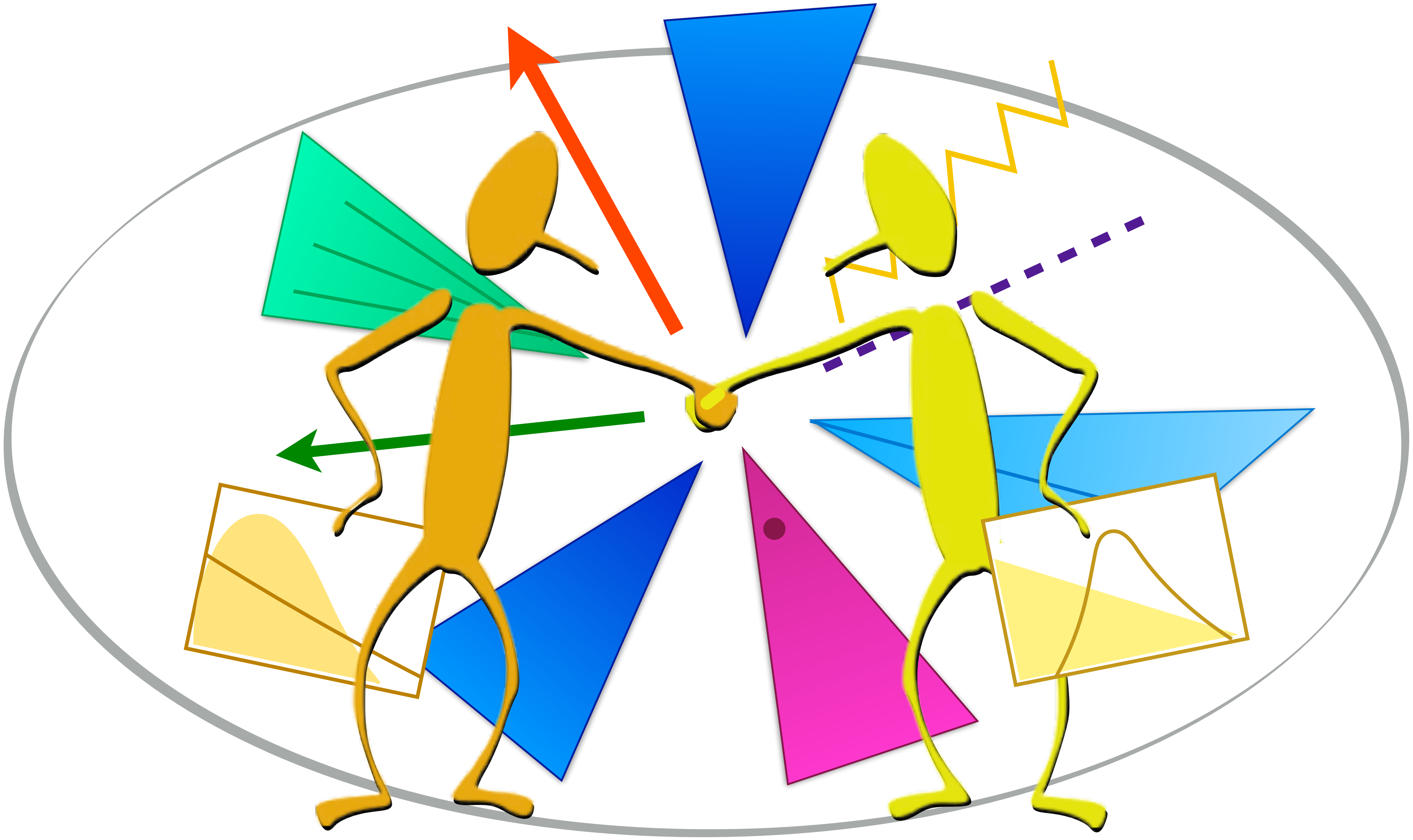
3rd International Turkey-Iran Joint Conference on LHC Physics



Analysis **D**escription **L**anguages for LHC

& *CutLang*

Gokhan UNEL, *UC Irvine*

Welcome to the LHC analysis jungle

Inclusive analyses with hundreds of selection regions

Overlaps between different analyses?

Is my control region your signal region???

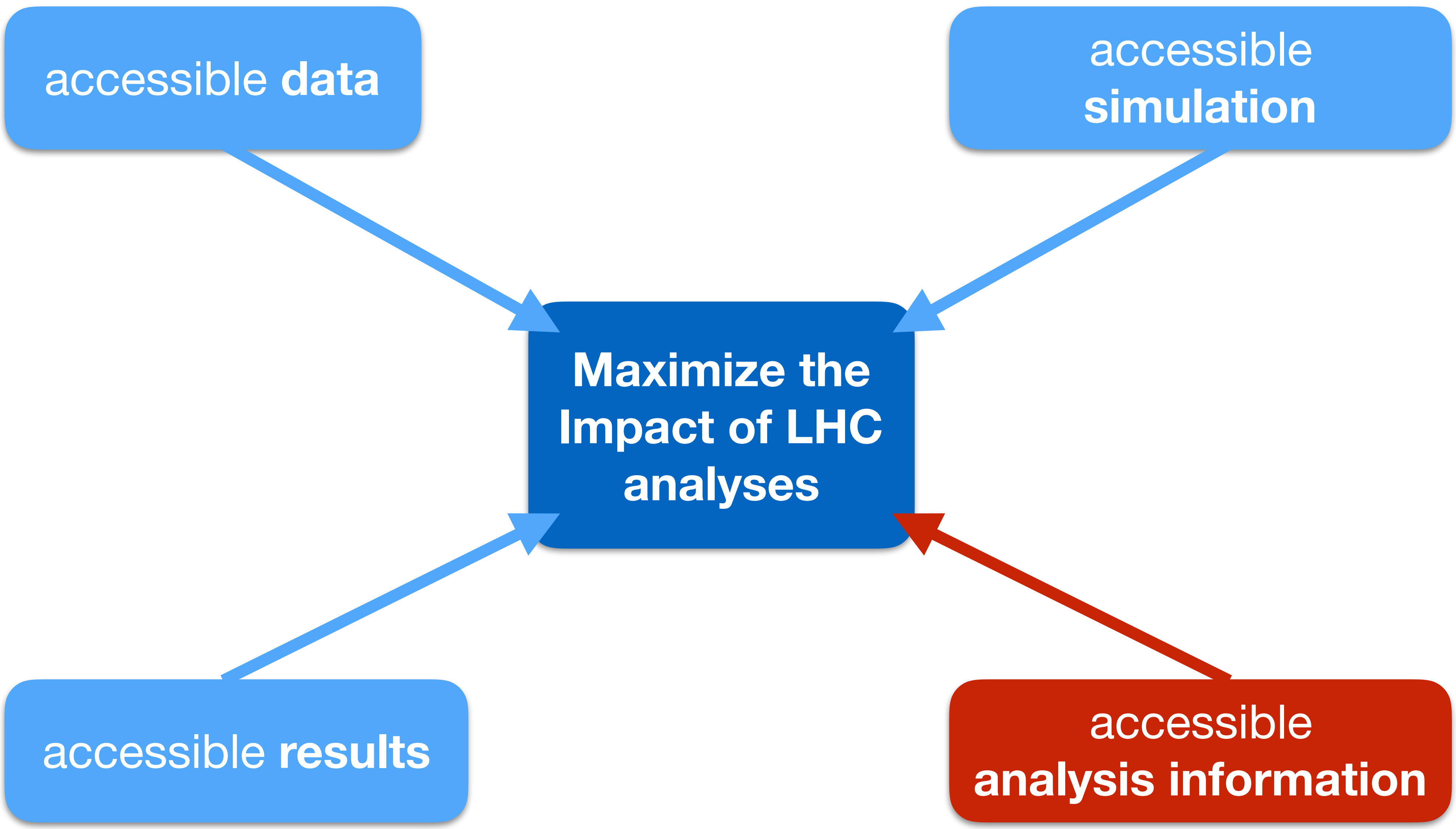
Multiple analyses exploring similar final states

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



Analysis description languages

Analysis description languages for LHC

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

D. Barducci, A. Buckley, G. Chalons, E. Conte, N. Desai, N. de Filippis, B. Fuks, P. Gras, S. Kraml, S. Kulkarni, U. Laa, M. Papucci, C. Pollard, H. B. Prosper, K. Sakurai, D. Schmeier, S. Sekmen, D. Sengupta, J. Sonneveld, J. Tattersall, G. Unel, W. Waltenberger, A. Weiler.

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.

ADL scope

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

Event processing…

input: event content → 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, …) → output: event selection

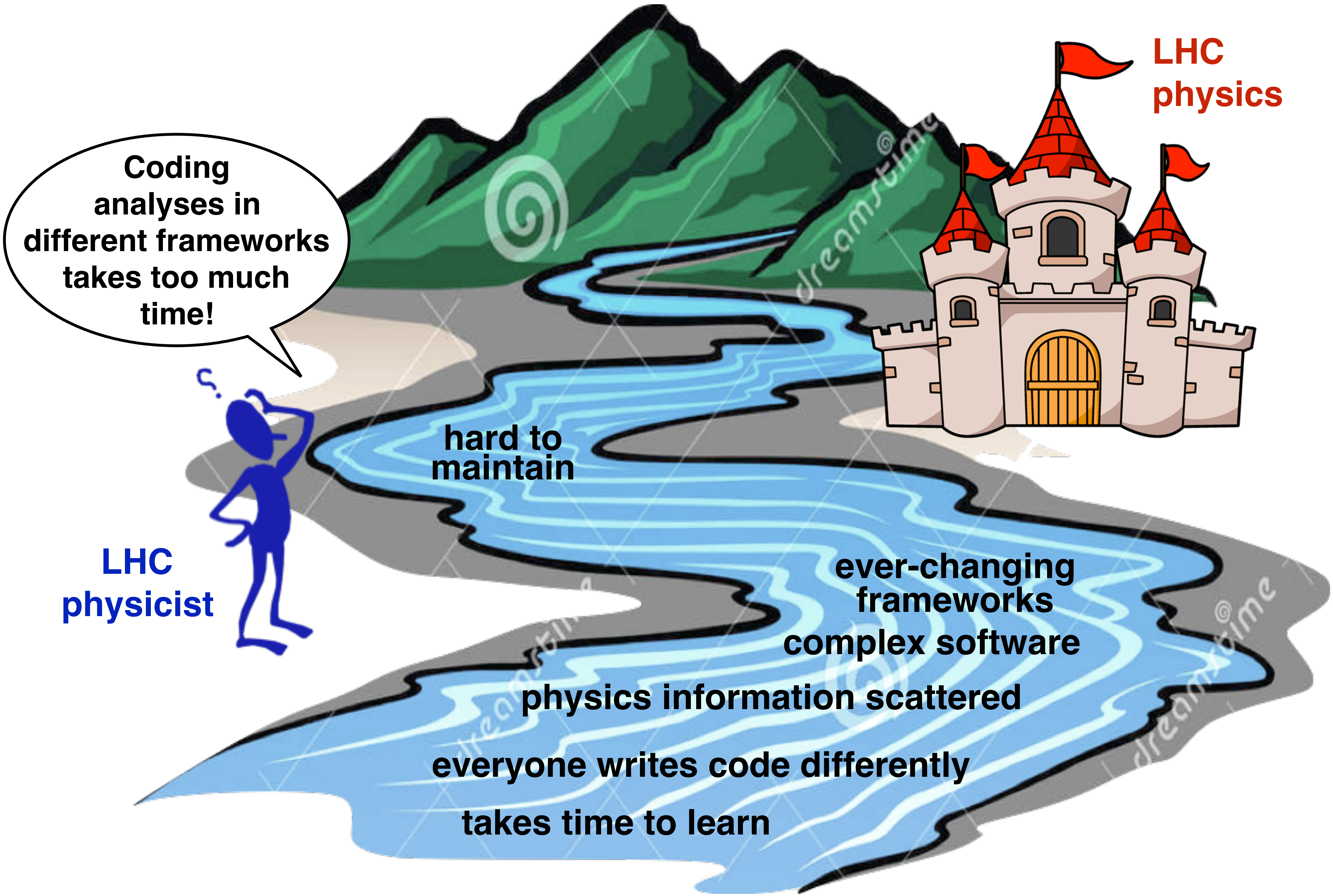
(optional: standard reports, visualizations)

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.

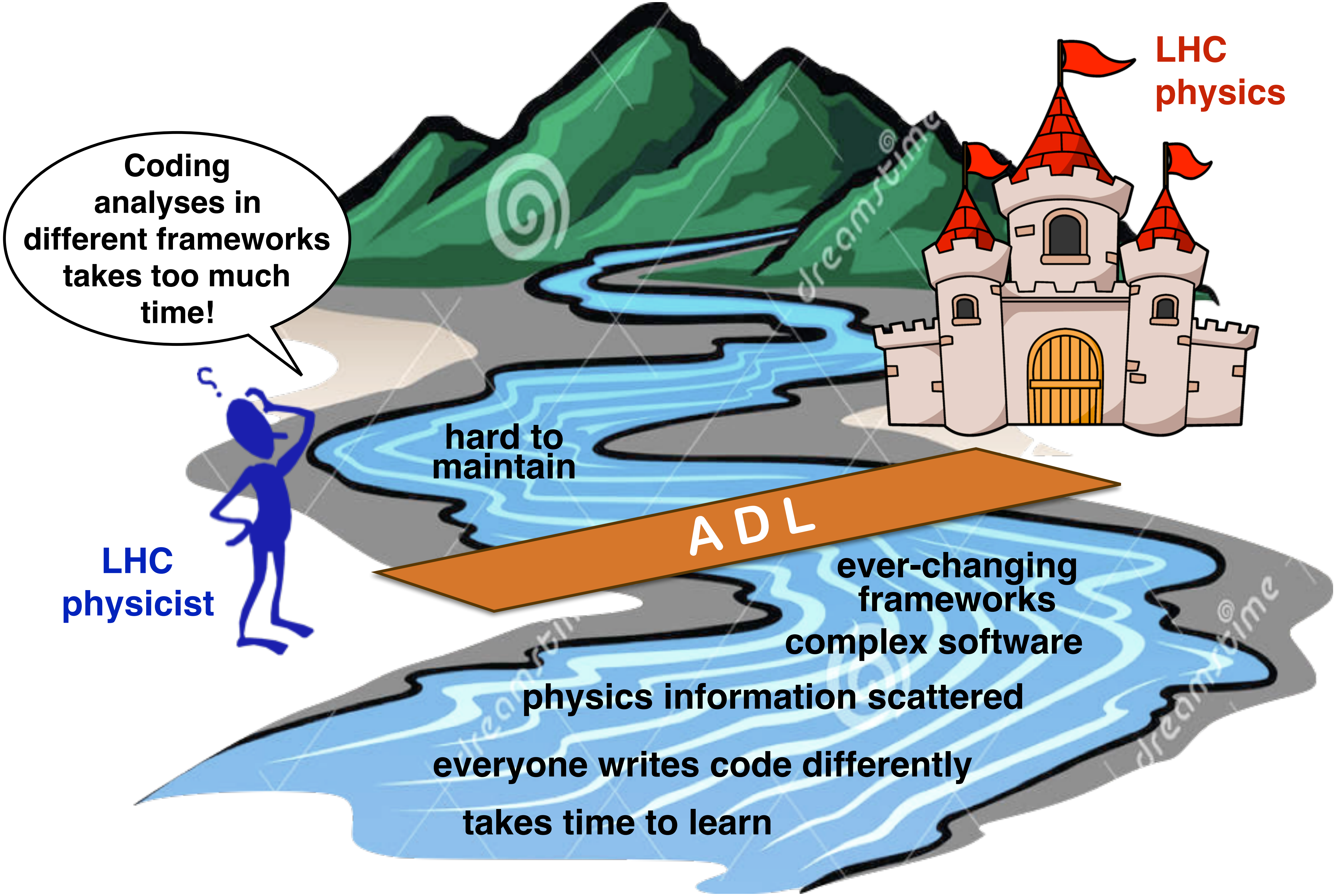
ADLs would help everyone

Motivation / use case	Exp	TH/ Pheno	Public
Analysis abstraction, design, implementation	✔	✔	✔
Analysis communication, clarification, synchronization, visualization	✔	✔	✔
Analysis review by internal or external referees	✔	✔	✔
Easier comparison/combination of analyses	✔	✔	
Interpretation studies, analysis reimplementation	✔	✔	✔
Analysis preservation (ongoing discussions with CERN Analysis Preservation Group)	✔	✔	✔
Improve our way of thinking about our analyses modelling and structure	✔	✔	✔

Framework independence highly desirable



Framework independence highly desirable



Features of an ADL for the LHC

Basic requirements:

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

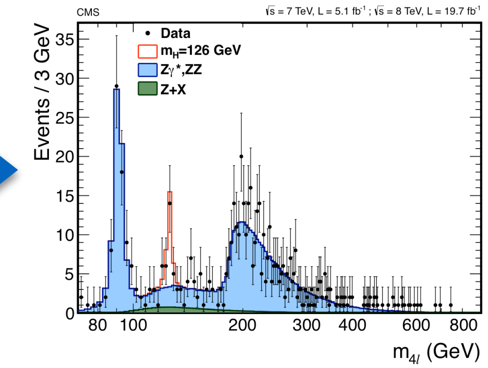
Desirable features:

- Self-contained
- Domain specific language (not a general purpose language)
- Analysis framework-independent

A specific ADL proposal

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 **L**es **H**ouches **A**nalysis **D**escription **A**ccord

D. Barducci, G. Chalons, N. Desai, N. de Filippis, P. Gras, S. Kraml, S. Kulkarni, U. Laa, M. Papucci, H. B. Prosper, K. Sakurai, D. Schmeier, S. Sekmen, D. Sengupta, J. Sonneveld, J. Tattersall, G. Unel, W. Waltenberger, A. Weiler.
LH 2015 New Phys WG report (arXiv:1605.02684), section 15

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.

—> Generic and abstract ADL design

CutLang: A particle physics ADL and runtime interpreter

Comput.Phys.Commun. 233 (2018) 215-236 (arXiv:1801.05727)

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

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

Color legend:
defined object
existing object
object attribute
internal function
selection criterion

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
```

From CMS SUSY razor analysis (Phys.Rev. D97 (2018) no.1, 012007, arxiv:1710.11188)

LHADA style full implementation link
CutLang style full implementation link

Examples: variable definitions

Color legend:

- defined variable
- existing object
- object attribute
- existing variable
- internal function
- external function

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 Rsql = 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

Color legend:
defined region
existing region
existing object
existing variable
internal function
external function
selection criterion

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
                              : MTe [] 30 100
  select Size(bjetsLoose) == 0
```

ADL block types and keywords

	LHADA—>ADL	CutLang —> ADL
object definition blocks	**object**	**obj / object**
event selection blocks	**region**	**algo / region**
analysis information	**info**	**info**
tables of results, etc.	**table**	—

	LHADA—>ADL	CutLang —> ADL
define variables, constants	**define**	**def / define**
select object or event	**select**	**select / cmd**
reject object or event	**reject**	—
define the mother object	**take**	**: / take / using**
define histograms	—	**histo**
applies object/event weights	**weight**	—
bins events in regions	**bin**	—

Green: Implemented in (some) parser/interpreter tools

Black: Implementation in progress

ADL operators

	LHADA—>ADL	CutLang —> ADL
Comparison operators	**> < => =< ==** **[]** (include) **][** (exclude)	**> < => =< ==** **[]** (include) **][** (exclude)
Mathematical operators	**+ - * / ^**	**+ - * / ^**
Logical operators	**and or**	**AND/&& OR/||**
Ternary operator	condition **?** true-case **:** false-case	condition **?** truecase **:** falsecase
Optimization operators	—	**~=** (closest to) **!=** (furthest from) (optimal particle sets are assigned negative indices)
Lorentz vector addition	LV1 **+** LV2	LV1 LV2 / LV1 **+** LV2

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: **abs()/||** , **sin()**, **cos()**, **tan()**, **log()**, **sqrt()**, … (mostly implemented in CutLang)
- Reducers: **size()**, **sum()**, **min()**, **max()**, **any()**, **all()**, …
- HEP-specific functions: **dR()**, **dphi()**, **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, vatiables computed with MVAs, …

Green: Implemented in CutLang and partially in other tools,

Black: Implementation in progress

Transpilers for LHADA style ADL - I

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 ➡ adl2tnm.py ➡ 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 jet_1
  - CL understands *particleName_index* notation:

jet$_1$

Highest Pt object	Second Highest Pt object
ELE_0	ELE_1
MUO_0	MUO_1

- On the computer, we write
  - CL understands *particleName*[*index*] notation:

jet[3]

```
muonsVeto[0]
photons[0]
```

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 ← more natural in Turkish
- CL understands both notations

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`

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)

user's ADL file

```
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 $Z \to \ell\ell \quad \ell = e, \mu$
  - the Z candidate should be neutral (q=0)

user's ADL file

```
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 %
```

2 electron combination is often used, why not to give it a name like Zreco?

A very simple example

- introducing definitions

$Z \to \ell\ell \quad \ell = e, \mu$

user's ADL file

```
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)
```

Are these electrons inside the inner tracker?

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 %
```

A simple example

- introducing derived objects

$Z \to \ell\ell \quad \ell = e, \mu$

  - Electron —> goodElectron

```
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
  histo       h1mReco,     "Z candidate mass (GeV)", 100, 0, 200, {Zreco}m
  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
  histo       h2mReco     , "Z candidate mass (GeV)", 100, 0, 200, m(Zreco)
  histo       h2mgoodReco, "Z candidate mass (GeV)", 100, 0, 200, m(goodZreco)
```

A simple example

- introducing multiple regions or algorithms
- A user defined region can contain another one
  - e.g. SignalRegion containing preselection

$$Z \rightarrow \ell\ell \quad \ell = e, \mu$$

```
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
  preselection
#  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
  preselection
  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)
```

Output file

```
TFile**         histoOut-ex5.root
 TFile*         histoOut-ex5.root
  KEY: TDirectoryFile   preselection;1  preselection
  KEY: TDirectoryFile   testA;1 testA
  KEY: TDirectoryFile   testB;1 testB
```

All regions are processed in parallel and saved as TDirectories in the output ROOT file

```
root [2] testA->cd()
(bool) true
root [3] .ls
TDirectoryFile*         testA   testA
 KEY: TText     CLA2cuts;1
  select       ALL
  select       Size(ELE)     >= 2
  select       {Zreco}q == 0
  histo        h2mReco     , "Z candidate mass (GeV)", 100, 0, 200, m(Zreco)
  select       ALL

 KEY: TText     CLA2defs;1
define Zreco : ELE[0] ELE[1]
define goodZreco : goodEle[0] goodEle[1]

 KEY: TText     CLA2Objs;1
object goodEle : ELE
  select    Pt(ELE_)        >   10
  select abs({ELE_}Eta )  <   2.4
  select     {ELE_}AbsEta ][  1.442 1.556

 KEY: TH1F      eff;1   selection efficiencies
 KEY: TNtuple   rntuple;1       run info
 KEY: TH1D      h2mReco;1       "Z candidate mass (GeV)"
```

A search example $Z \to \ell\ell \quad \ell = e, \mu$

- 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

if we have 3 electrons in an event

e1	e2
1	2
1	3
2	3
2	1
3	1
3	2

```
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: **~=**

A search example $Z \rightarrow \ell\ell \quad \ell = e, \mu$

- Taking a short cut
  - e1 + e2 = e2 + e1 —> same Z, no need to calculate both
  - repeating the same negative index (-1) tells CutLang to compute only one
    - compute time reduced by 50%

if we have 3 electrons in an event

e1	e2
1	2
1	3
2	3

```
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)
```

User (external) 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:

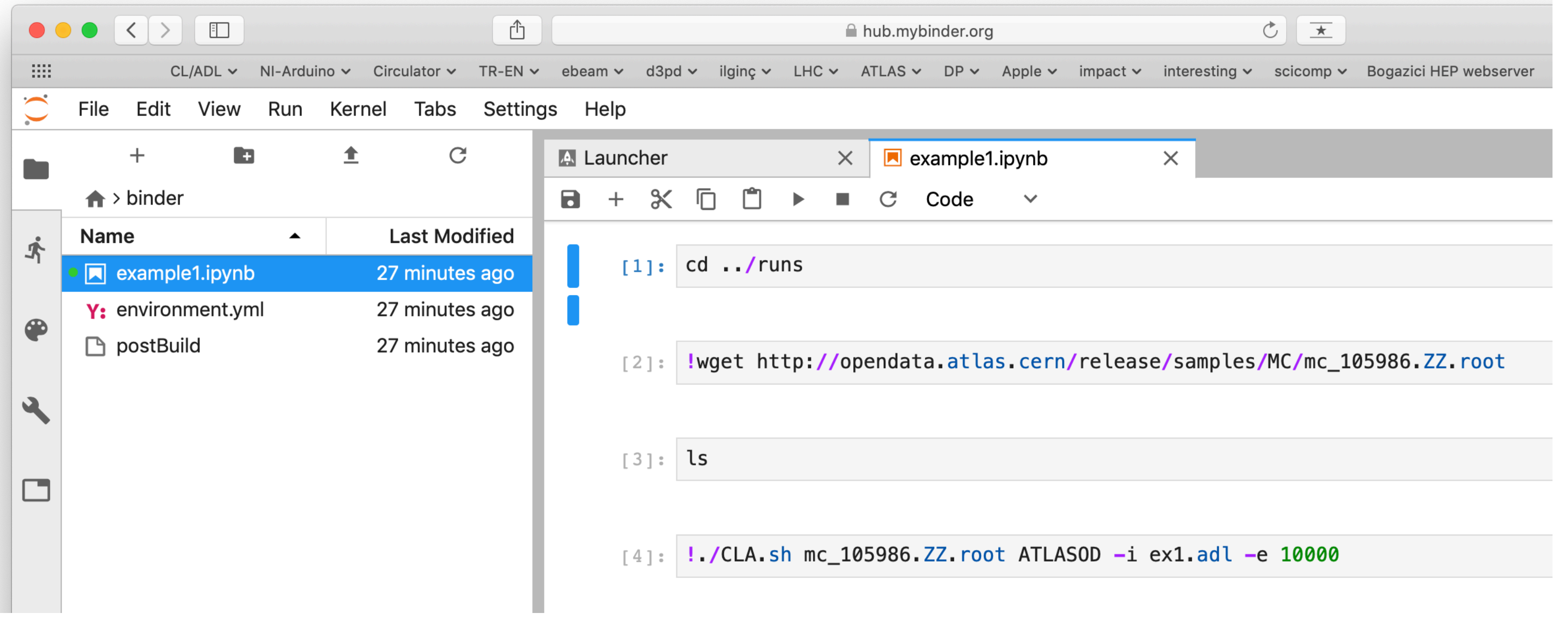
```
std::vector<TLorentzVector> fmegajets(std::vector<TLorentzVector> myjets);
double fMR(std::vector<TLorentzVector> j);
double fMTR(std::vector<TLorentzVector> j, TVector2 amet);
double fMTR2(std::vector<TLorentzVector> j, TLorentzVector amet);
```

- Simple functions can be interpreted using CL math functions

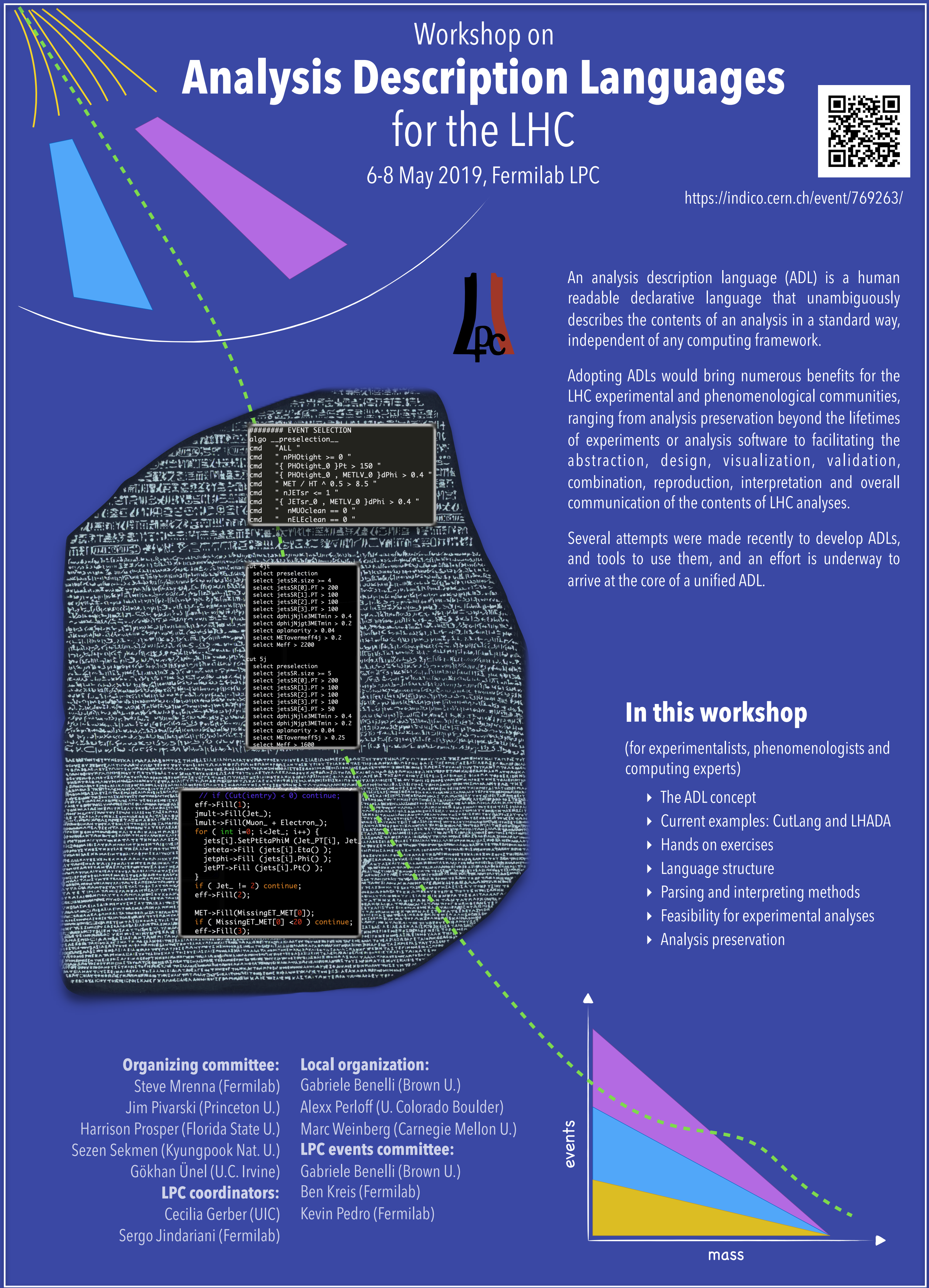
```
    return sqrt( 2 * lepton.Pt() * pfmet.Pt() * ( 1 - cos( pfmet.Phi() - lepton.Phi() )));
```

```
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
  - your 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 vidyo recordings on indico:

https://indico.cern.ch/event/769263/

thank you for your attention

backup slides

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

reference guide

- The Objects

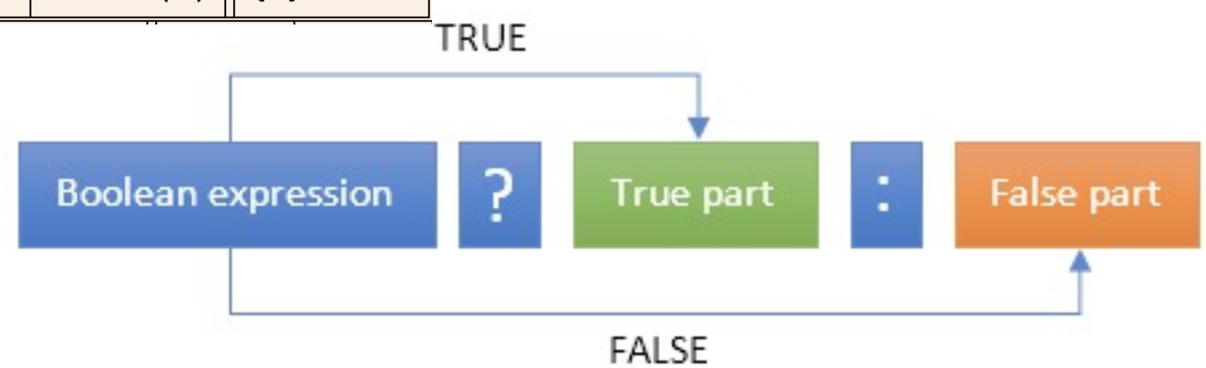
Name	Keyword	Highest Pt object	Second Highest Pt object	$j+1^{th}$Highest Pt object
Electron	ELE	ELE_0	ELE_1	ELE_j
Muon	MUO	MUO_0	MUO_1	MUO_j
Tau	TAU	TAU_0	TAU_1	TAU_j
Lepton	LEP	LEP_0	LEP_1	LEP_j
Photon	PHO	PHO_0	PHO_1	PHO_j
Jet	JET	JET_0	JET_1	JET_j
Fat Jet	FJET	FJET_0	FJET_1	FJET_j
b-tagged Jet	BJET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

- Functions

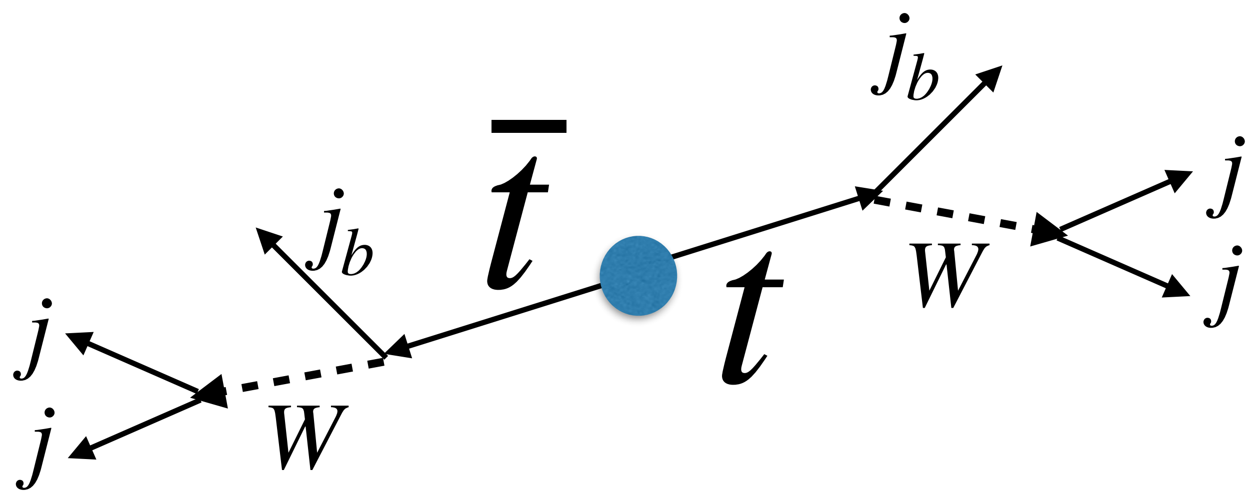
Meaning	Operator
number of	Size( )
tangent	tan()
sinus	sin()
cosinus	cos()
absolute value	abs()
square root	sqrt()
in the interval	[ ]
not in the interval	] [
as close as possible	~=
as far away as possible	!=
usual meaning	+-/*
to the power	^

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

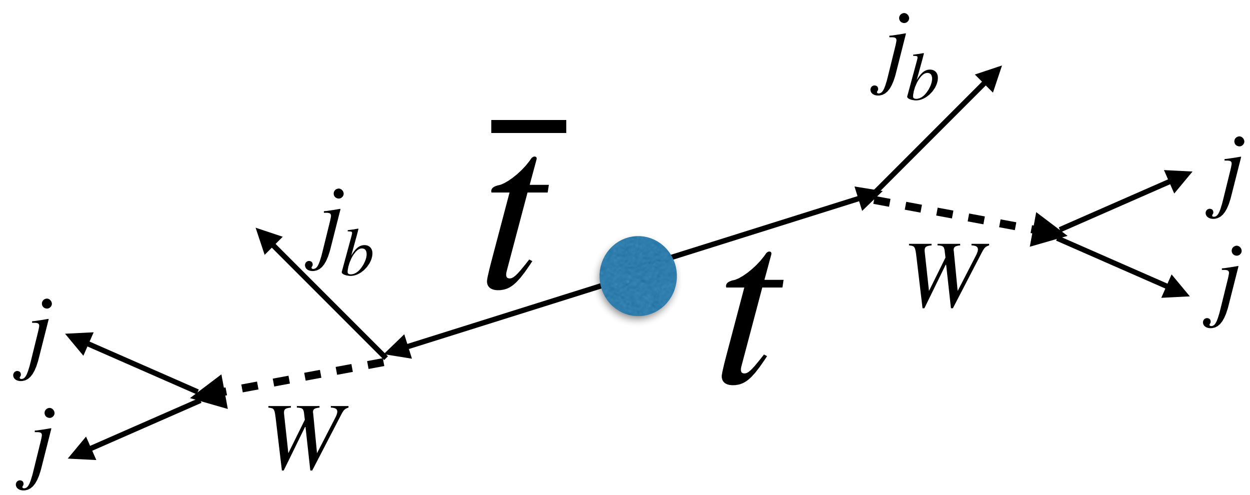
- The ternary function in C notation



$t\bar{t}$ Reconstruction example

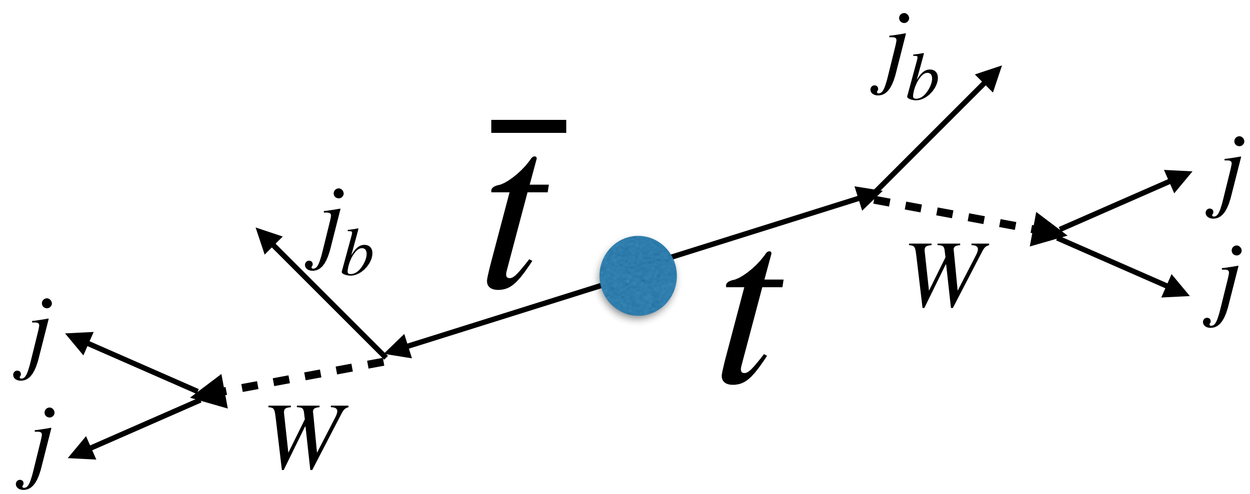


$t\bar{t}$ Reconstruction example



$$t \rightarrow Wb \rightarrow jjj_b$$

$t\bar{t}$ Reconstruction example

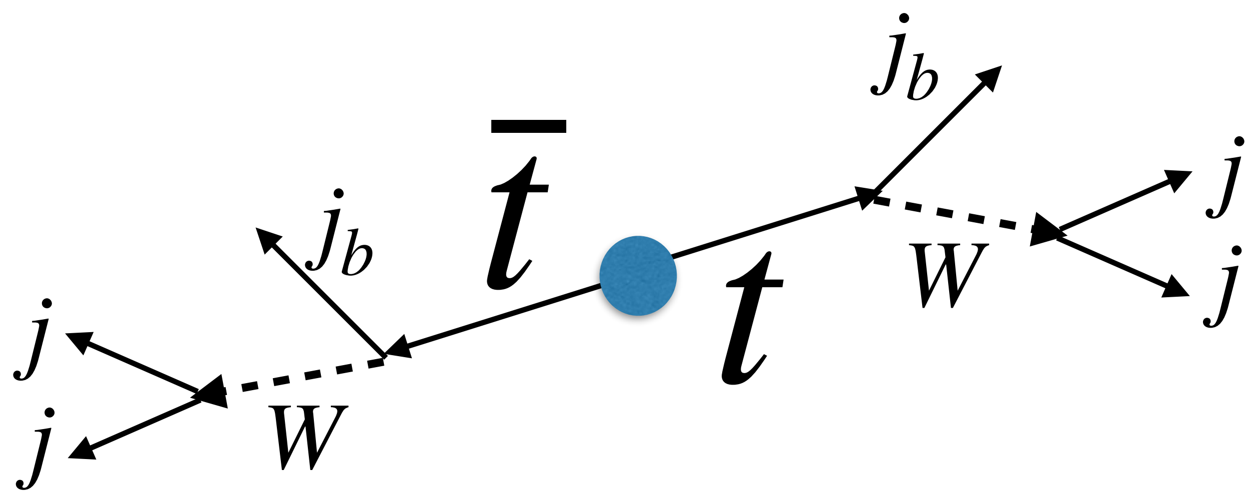


$$t \to Wb \to jjj_b$$

There are 6 jets in the event of which 2 can be b-tagged

+ LOTS of *other jets* from spectator quarks and QCD effects

$t\bar{t}$ Reconstruction example



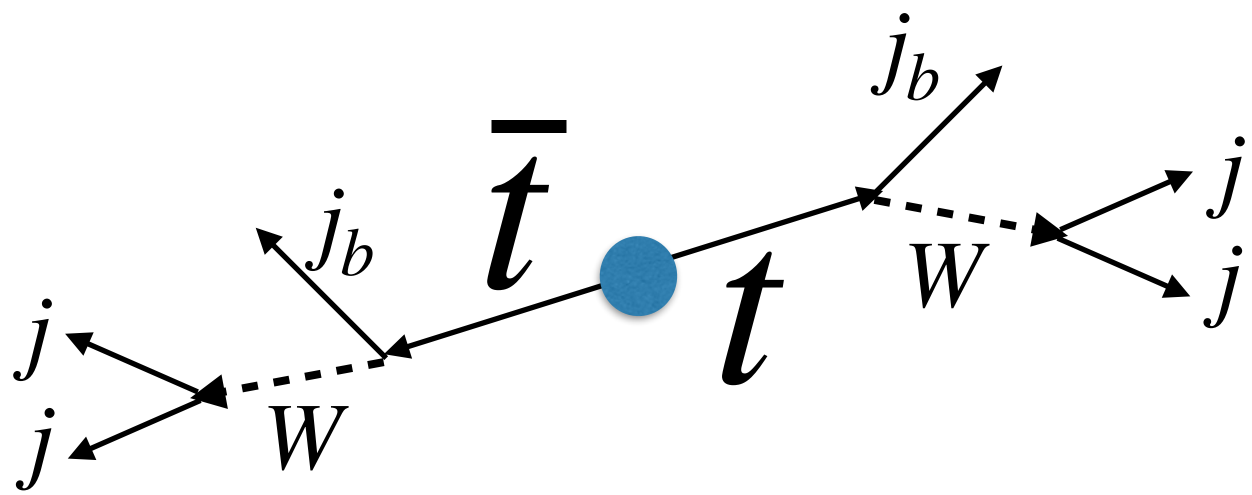
$$t \rightarrow Wb \rightarrow jjj_b$$

There are 6 jets in the event of which 2 can be b-tagged

+ LOTS of *other jets* from spectator quarks and QCD effects

Which one is which?

$t\bar{t}$ Reconstruction example



$$t \to Wb \to jjj_b$$

There are 6 jets in the event of which 2 can be b-tagged

+ LOTS of *other jets* from spectator quarks and QCD effects

Which one is which?

with the χ^2 defined as:

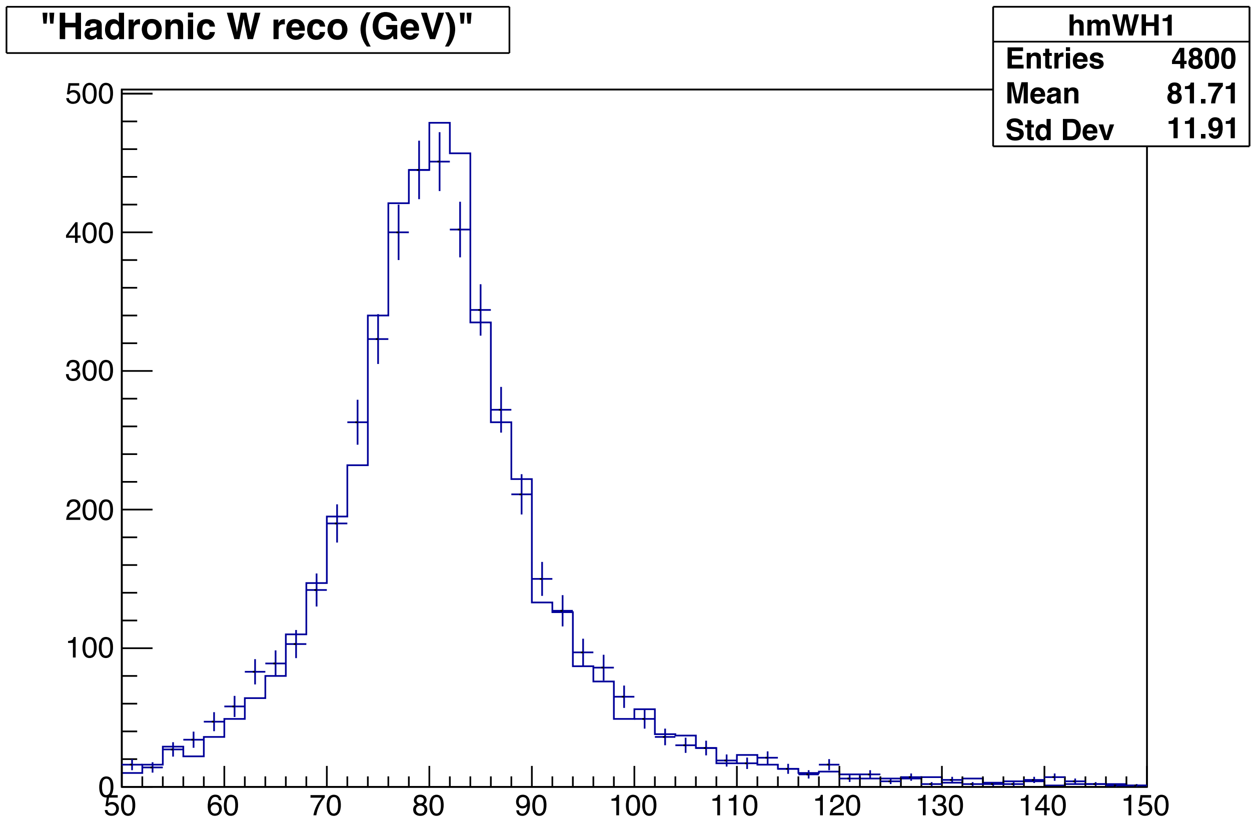
$$\chi^2 = \frac{(m_{b_1 j_1 j_2} - m_{b_2 j_3 j_4})^2}{\sigma^2_{\Delta m_{bjj}}} + \frac{(m_{j_1 j_2} - m_W^{\mathrm{MC}})^2}{\sigma^2_{m_W^{\mathrm{MC}}}} + \frac{(m_{j_3 j_4} - m_W^{\mathrm{MC}})^2}{\sigma^2_{m_W^{\mathrm{MC}}}}.$$

$t\bar{t}$ Reconstruction example

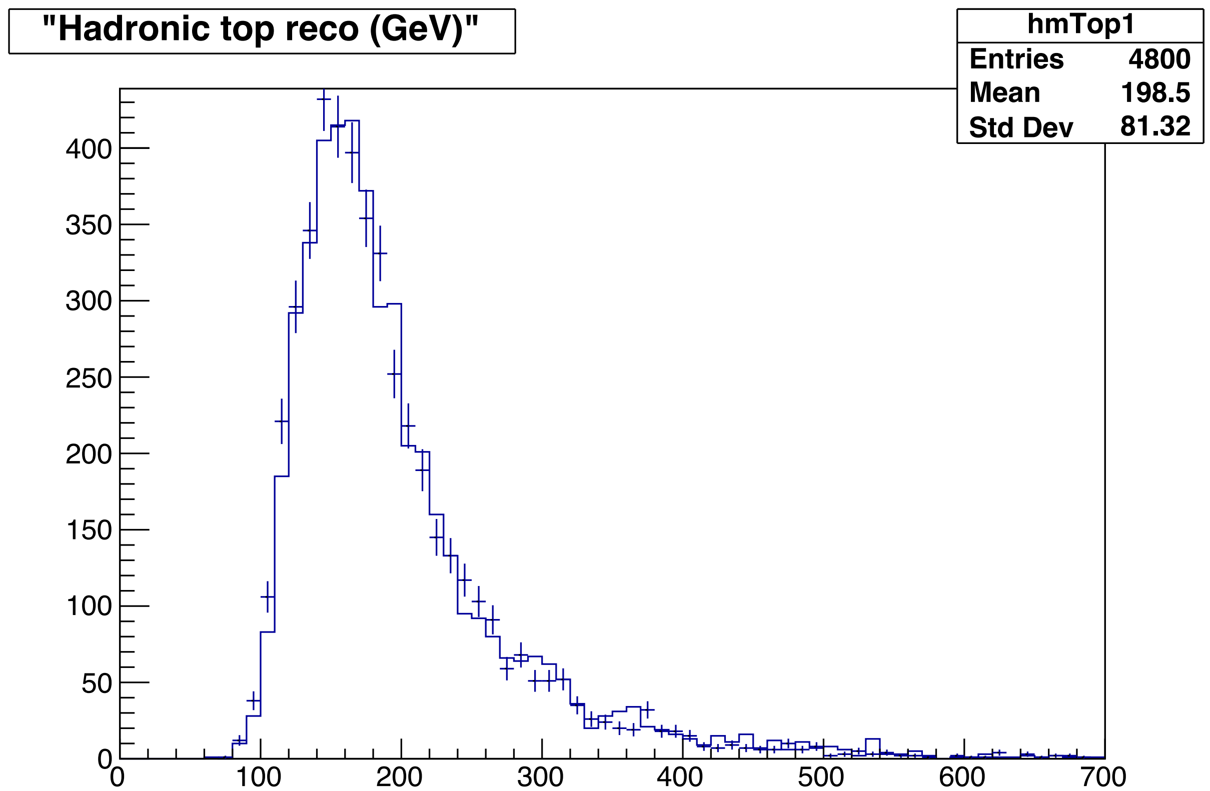
```
define   WH1  : JET[-1] JET[-1]
define   WH2  : JET[-3] JET[-3]
###      chi2 for W finder
define   Wchi2 : (({WH1}m - 80.4)/2.1)^2 + (({WH2}m - 80.4)/2.1)^2

## 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 χ^2 defined as:

$$\chi^2 = \frac{(m_{b_1j_1j_2} - m_{b_2j_3j_4})^2}{\sigma^2_{\Delta m_{bjj}}} + \frac{(m_{j_1j_2} - m_W^{MC})^2}{\sigma^2_{m_W^{MC}}} + \frac{(m_{j_3j_4} - m_W^{MC})^2}{\sigma^2_{m_W^{MC}}}.$$



reconstructed W bosons



reconstructed top quarks

Razor boost example 1/2

```
# arxiv:1710.11188, CMS SUSY stop (resolved and boosted)

#info analysis
# Details about experiment
#  experiment CMS
#  id SUS-16-050
#  publication Phys.Rev. D97 (2018) no.1, 012007
#  sqrtS 13.0
#  lumi 35.9
#  arXiv 1710.11188
#  hepdata https://www.hepdata.net/record/ins1633588
#  doi 10.1103/PhysRevD.97.012007

### OBJECT SELECTIONS

# AK4 jets
object AK4jets : JET
  select {JET_}Pt > 30
  select {JET_}AbsEta < 2.4

# AK8 jets
object AK8jets : FJET
  select {FJET_}Pt > 200
  select {FJET_}AbsEta < 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

# b-tagged jets - tight
object bjetsTight : AK4jets
  select {AK4jets_}btagDeepB > 0.8001

# W jets - mass-tagged
object WjetsMasstag : AK8jets
  select {AK8jets_}msoftdrop [] 65 105

# W jets - W-tagged
object Wjets : WjetsMasstag
  select {WjetsMasstag_}tau2  <= 0.2
  select {WjetsMasstag_}tau2 / {WjetsMasstag_}tau1 <= 0.4

# W jets - anti-tagged
object WjetsAntitag : WjetsMasstag
  select {WjetsMasstag_}tau2 / {WjetsMasstag_}tau1 > 0.4

# top jets - mass-tagged
object topjetsMasstag : AK8jets
  select {AK8jets_}Pt > 400
  select {AK8jets_}msoftdrop [] 105 210

# top jets - mass-tagged, subjet b-antitagged
object topjetsMasstag0b : topjetsMasstag
  select {topjetsMasstag_}btagDeepB < 0.1522

# top jets - top-tagged, subjet b-tagged
object topjets : topjetsMasstag
  select {topjetsMasstag_}btagDeepB >= 0.1522
  select {topjetsMasstag_}tau3 / {topjetsMasstag_}tau2 < 0.46

# top jets - anti-tagged
object topjetsAntitag : topjetsMasstag
  select {topjetsMasstag_}btagDeepB < 0.1522
  select {topjetsMasstag_}tau3 / {topjetsMasstag_}tau2 >= 0.46

# muons - veto
object muonsVeto : MUO
  select {MUO_}Pt > 5
  select {MUO_}AbsEta < 2.4
  select {MUO_}softId == 1
  select {MUO_}miniPFRelIsoAll < 0.2
  select abs({MUO_}dxy) < 0.2  ## how to take the abs of this
  select abs({MUO_}dz) < 0.5   ## and this?

# muons - select
object muonsSel : MUO
  select {MUO_}Pt > 10
  select {MUO_}AbsEta < 2.4
  select {MUO_}miniPFRelIsoAll < 0.15
  select abs({MUO_}dxy) < 0.05
  select abs({MUO_}dz) < 0.1

# electrons - veto
object electronsVeto : ELE
  select {ELE_}Pt > 5
  select {ELE_}AbsEta < 2.5
  select {ELE_}miniPFRelIsoAll < 0.1
  select abs({ELE_}dxy) < 0.05
  select abs({ELE_}dz) < 0.1

# electrons - select
object electronsSel : ELE
  select {ELE_}Pt > 10
  select {ELE_}AbsEta < 2.5
  select {ELE_}AbsEta  ][ 1.442 1.556
  select {ELE_}miniPFRelIsoAll < 0.1
  select abs({ELE_}dxy) < 0.05
  select abs({ELE_}dz) < 0.1

# taus - veto
object tausVeto : TAU
  select {Tau_}Pt > 18
  select {Tau_}AbsEta < 2.5
  select {Tau_}dMVAnewDM2017v2 >= 4

# photons - select
object photons : PHO
  select {PHO_}Pt > 5
  select {PHO_}AbsEta < 2.5

# jets - no photon
object AK4jetsNOpho : AK4jets
  select dR(AK4jets_, photons_ ) >=0.4 OR {photons_}Pt/{AK4jets_}Pt ][ 0.5 2.0
#  reject dR(AK4jets_, photons) < 0.4 AND photons.pt/j.pt [] 0.5 2.0

### EVENT VARIABLES
object megajets : AK4jets
  select fmegajets(AK4jets) == 2

object megajetsNOpho : AK4jetsNOpho
  select fmegajets(AK4jetsNOpho) == 2

def newdefinitions ### this comment with a dummy ID has to sit right after obje
define MR : fMR(megajets)
define Rsq : (fMTR(megajets, MET) / MR )^0.5
define dphimegajets : dPhi(megajets_0, megajets_1)
define dphimegajetsNOpho : dPhi(megajetsNOpho_0, megajetsNOpho_1)

define METLVe   : METLV_0 electronsVeto_0
define METLVm   : METLV_0 muonsVeto_0
define METLVee  : METLV_0 electronsVeto_0  electronsVeto_1
define METLVmm  : METLV_0 muonsVeto_0  muonsVeto_1
define METLVpho : METLV_0 photons_0

define R2e : (fMTR(megajets, METLVe) / MR)^0.5
define R2m : (fMTR(megajets, METLVm) / MR)^0.5
define R2ee : sqrt(fMTR(megajets, METLVee) / MR)
define R2mm : sqrt(fMTR(megajets, METLVmm) / MR)
define MR0pho : fMR(megajetsNOpho)
define R2pho : sqrt(fMTR(megajetsNOpho, METLVpho) / MR0pho)
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
```

Razor boost example 2/2

```
# EVENT SELECTION
# Boosted categories

# Boost pre-selection cuts
region preselection
  select ALL   # This is only to see the initial event count
  select Size(AK4jets) >= 3
  select Size(AK8jets) >= 1
  select Size(megajets) == 2
  select MR > 800
  select Rsq > 0.08

region WcategorySR
  preselection
  select Size(electronsVeto) + Size(muonsVeto) == 0
  select Size(tausVeto) == 0
  select Size(bjetsMedium) >= 1
  select Size(Wjets) >= 1
  histo hWjPT, "Wjets Pt GeV", 10, 0, 500, {Wjets_0}Pt
  select Size(Wjets) >= 1
  select dphimegajets < 2.8

region WcategoryCRQ
  preselection
  select Size(electronsVeto) + Size(muonsVeto) == 0
  select Size(tausVeto) == 0
  select Size(WjetsAntitag) >= 1
  select Size(bjetsLoose) == 0
  select dphimegajets >= 2.8

region WcategoryCRT
  preselection
  select Size(electronsVeto) + Size(muonsVeto) == 1
  select Size(bjetsLoose) >= 1
  select Size(Wjets) >= 1
  select dphimegajets < 2.8
  select Size(muonsVeto) == 1 ? MTm < 100 : MTe < 100

region WcategoryCRW
  preselection
  select Size(muonsVeto) + Size(electronsVeto) == 1
  select Size(bjetsLoose) == 0
  select Size(WjetsMasstag) >= 1
  select dphimegajets < 2.9
  select Size(muonsVeto) == 1 ? MTm [] 30 100 : MTe [] 30 100
```

```

region WcategoryCRL
  select Size(AK4jets) >= 3
  select Size(AK8jets) >= 1
  select Size(megajets) == 2
  select MR > 800
  select Size(muonsVeto) + Size(electronsVeto) == 1
  select Size(muonsVeto) == 1 ? R2m > 0.08 : R2e > 0.08
  select Size(bjetsLoose) == 0
  select Size(WjetsMasstag) >= 1
  select dphimegajets < 2.10
  select Size(muonsVeto) == 1 ? MTm [] 30 100 : MTe [] 30 100

region WcategoryCRZ
  select Size(AK4jets) >= 3
  select Size(AK8jets) >= 1
  select Size(megajets) == 2
  select MR > 800
  select (Size(muonsSel) == 2 AND Size(electronsVeto) == 0) OR (Size(electronsSel) == 2 AND Size(muonsVeto) == 0)
  select Size(muonsSel) == 2 ? {muonsSel_0}q + {muonsSel_1}q == 0 : {electronsSel_0}q + {electronsSel_1}q == 0
  select Size(muonsSel) == 2 ? Abs({muonsSel_0 muonsSel_1}m - mZ) < 10 : Abs({electronsSel_0 electronsSel_1}m - mZ) < 10
  select Size(muonsSel) == 2 ? R2mm > 0.08 : R2ee > 0.08
  select Size(WjetsMasstag) >= 1
  select dphimegajets < 2.8

region WcategoryCRG
  select Size(photons) > 0
  select Size(AK4jetsNOpho) >= 3
  select Size(AK8jets) >= 1
  select Size(electronsVeto) + Size(electronsVeto) == 0
  select Size(tausVeto) == 0
  select Size(megajetsNOpho) == 2
  select MR0pho > 800
  select R2pho > 0.08
  select Size(WjetsMasstag) >= 1
  select dphimegajetsNOpho < 2.8

# # Top category signal and control regions
region TopcategorySR
  preselection
  select Size(electronsVeto) + Size(muonsVeto) == 0
  select Size(tausVeto) == 0
  select Size(topjets) >= 1
```

```
  select dphimegajets < 2.8

region TopcategoryCRQ
  preselection
  select Size(electronsVeto) + Size(muonsVeto) == 0
  select Size(tausVeto) == 0
  select Size(topjetsAntitag) >= 1
  select Size(bjetsLoose) == 0
  select dphimegajets >= 2.8

region TopcategoryCRT
  preselection
  select Size(electronsVeto) + Size(muonsVeto) == 1
  select Size(bjetsLoose) >= 1
  select Size(topjets) >= 1
  select dphimegajets < 2.8
  select Size(muonsVeto) == 1 ? MTm < 100 : MTe < 100

region TopcategoryCRW
  preselection
  select Size(muonsVeto) + Size(electronsVeto) == 1
  select Size(bjetsLoose) == 0
  select Size(topjetsMasstag0b) >= 1
  select dphimegajets < 2.8
  select Size(muonsVeto) == 1 ? MTm [] 30 100 : MTe [] 30 100

region TopcategoryCRL
  select Size(AK4jets) >= 3
  select Size(AK8jets) >= 1
  select Size(megajets) == 2
  select MR > 800
  select Size(muonsVeto) + Size(electronsVeto) == 1
  select Size(muonsVeto) == 1 ? R2m > 0.08 : R2e > 0.08
  select Size(bjetsLoose) == 0
  select Size(topjetsMasstag0b) >= 1
  select dphimegajets < 2.10
  select Size(muonsVeto) == 1 ? MTm [] 30 100 : MTe [] 30 100

region TopcategoryCRZ
  select Size(AK4jets) >= 3
  select Size(AK8jets) >= 1
  select Size(megajets) == 2
  select MR > 800
  select (Size(muonsSel) == 2 AND Size(electronsVeto) == 0) OR (Size(electronsSel) == 2 AND Size(muonsVeto) == 0)
```


  select (Size(muonsSel) == 2 AND Size(electronsVeto) == 0) OR (Size(electronsSel) == 2 AND Size(muonsVeto) == 0)


```
  select Size(muonsSel) == 2 ? {muonsSel_0}q + {muonsSel_1}q == 0 : {electronsSel_0}q + {electronsSel_1}q == 0
  select Size(muonsSel) == 2 ? Abs({muonsSel_0 muonsSel_1}m - mZ) < 10 : Abs({electronsSel_0 electronsSel_1}m - mZ) < 10
  select Size(muonsSel) == 2 ? R2mm > 0.08 : R2ee > 0.08
  select Size(topjetsMasstag) >= 1
  select dphimegajets < 2.8

region TopcategoryCRG
  select Size(photons) > 0
  select Size(AK4jetsNOpho) >= 3
  select Size(AK8jets) >= 1
  select Size(electronsVeto) + Size(electronsVeto) == 0
  select Size(tausVeto) == 0
  select Size(megajetsNOpho) == 2
  select MR0pho > 800
  select R2pho > 0.08
  select Size(topjetsMasstag) >= 1
  select dphimegajetsNOpho < 2.8
```

```
#  experiment ATLAS
#  id SUSY-2013-15
#  publication Eur. Phys. J. C(2016) 76: 392
#  sqrtS 13.0
#  lumi 3.2
#  arXiv 1605.03814
```

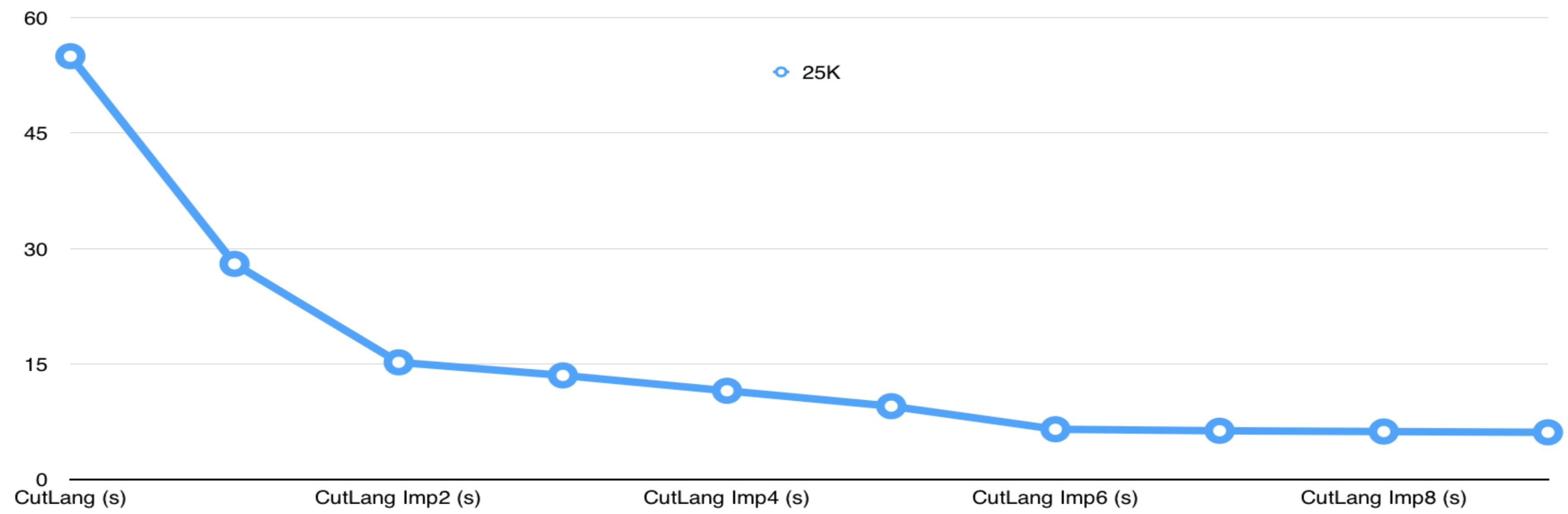
Compatibility

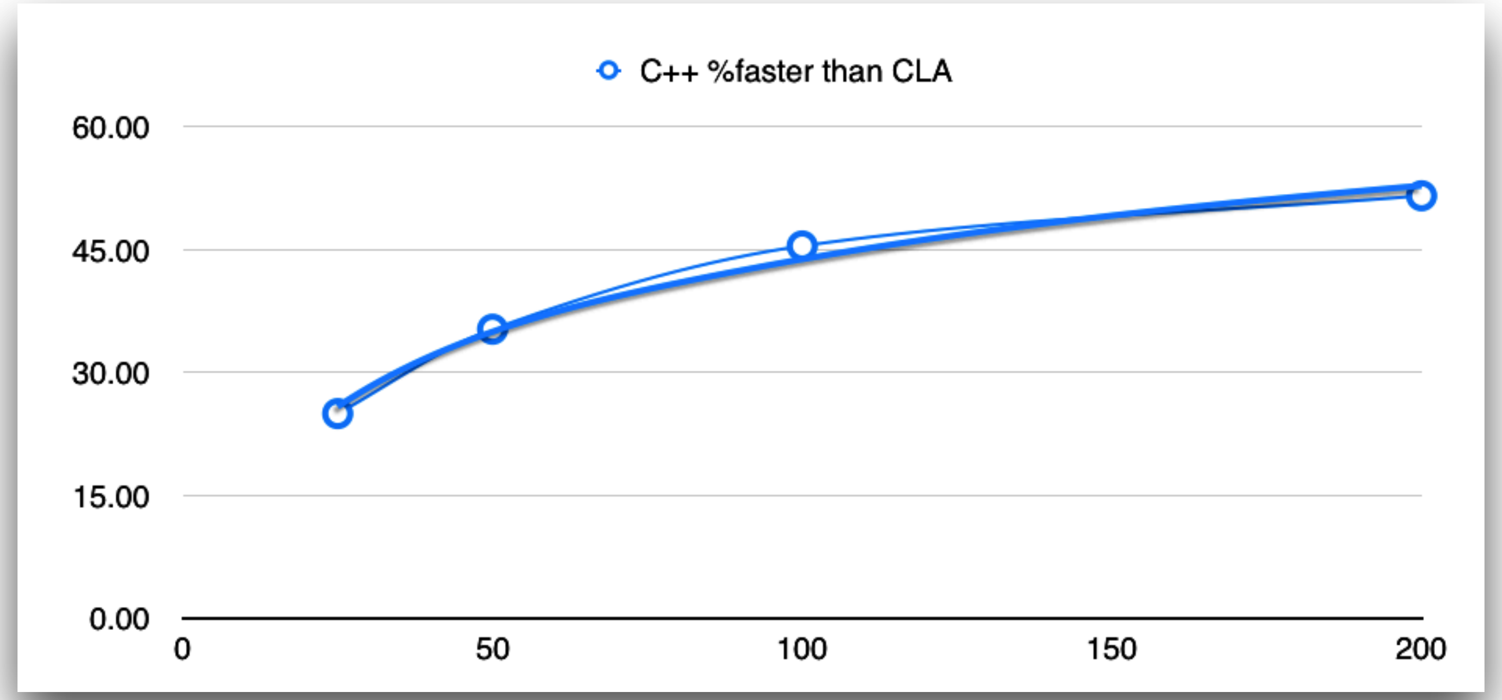
	Rivet			MadAnalysis 5			CheckMATE	CutLang		
Description	#evt	tot.eff	rel.eff	#evt	tot.eff	rel.eff	tot.eff	#evt	total eff.	rel. eff.
2jl cut-flow	31250	1	-	31250	1	-		31250	1.000	-
Pre-sel+MET+pT1	28592	0.91	0.91	28626	0.92	0.92		28431	0.91	0.91
Njet	28592	0.91	1	28625	0.92	1		28430	0.91	1.00
Dphi_min(j,MET)	17297	0.55	0.6	17301	0.55	0.6		16661	0.53	0.59
pT2	17067	0.55	0.99	17042	0.55	0.99		16381	0.52	0.98
MET/sqrtHT	8900	0.28	0.52	8898	0.28	0.52		8159	0.26	0.50
m_eff(incl)	8896	0.28	1	8897	0.28	1		8156	0.26	1.00
2jm cut-flow	31250	1	-	32150	1	-	1	31250	1.000	-
Pre-sel+MET+pT1	28472	0.91	0.91	28478	0.91	0.91	0.91	28301	0.91	0.91
Njet	28472	0.91	1	28477	0.91	1	0.91	28300	0.91	1.00
Dphi_min(j,MET)	22950	0.73	0.81	22889	0.73	0.8	0.73	22441	0.72	0.79
pT2	22950	0.73	1	22889	0.73	1	0.73	22441	0.72	1.00
MET/sqrtHT	10730	0.34	0.47	10710	0.34	0.47	0.33	10043	0.32	0.45
m_eff(incl)	10630	0.34	0.99	10609	0.34	0.99	0.32	9896	0.32	0.99
2jt cut-flow	31250	1	-	31250	1	-		31250	1.000	-
Pre-sel+MET+pT1	28592	0.91	0.91	28626	0.92	0.92		28431	0.91	0.91
Njet	28592	0.91	1	28625	0.92	1		28430	0.91	1.00
Dphi_min(j,MET)	17297	0.55	0.6	17301	0.55	0.6		16661	0.53	0.59
pT2	17067	0.55	0.99	17042	0.55	0.99		16381	0.52	0.98
MET/sqrtHT	5083	0.16	0.3	5098	0.16	0.3		4375	0.14	0.27
Pass m_eff(incl)	4861	0.16	0.96	4889	0.16	0.96		4132	0.13	0.94

Debugging & speeding

ATLAS hadronic ttbar tests

	25K	50K	100K
Sezen (s)	12	24	45.7
Sezen imp (s)	3.1	4.7	7.9
CutLang (s)	55	106	210
CutLang Imp (s)	28	55	108
CutLang Imp2 (s)	15.2	29	56.2
CutLang Imp3 (s)	13.5	25.8	49.7
CutLang Imp4 (s)	11.5	21.6	41.9
CutLang Imp5 (s)	9.5	17.5	33.5
CutLang Imp6 (s)	6.5	11.5	21.6
CutLang Imp7 (s)	6.3	11.1	20.8
CutLang Imp8 (s)	6.2	11.0	20.4
CutLang Imp9 (s)	6.1	10.9	20.1
ratio best	1.9677419	2.319148	2.544303
%slow	103.226	136.170	163.291
%faster	49.180	56.881	60.697
	25	50	100





ATLAS wz tests e- channel

	500K
Sezen (s)	6.3
CutLang Imp (s)	6.9
ratio	1.10
%faster	8.70

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++.