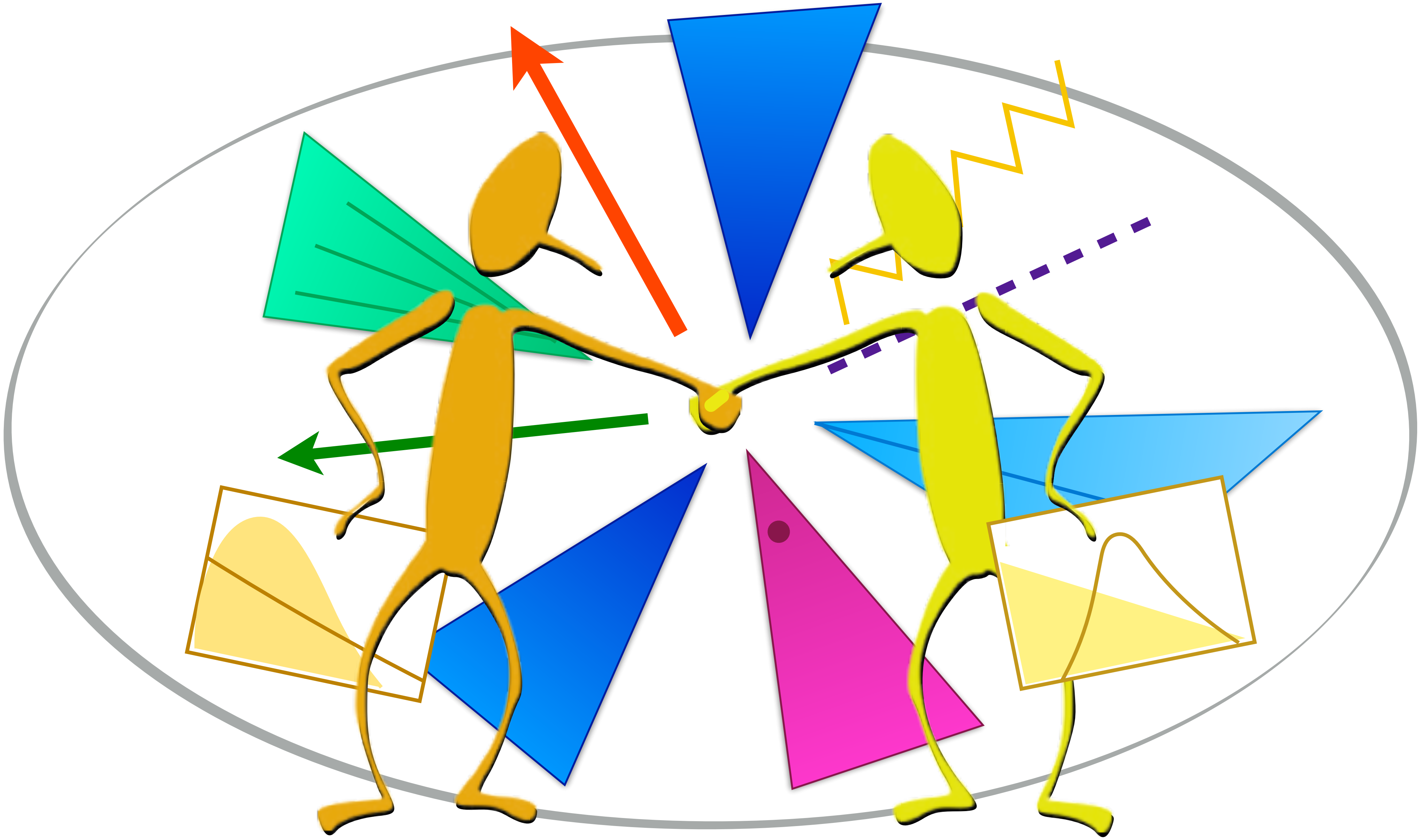


Analysis Description Languages for LHC BSM searches

Sezen Sekmen (Kyungpook National University)

SUSY 2019, Corpus Christi, 20-24 May 2019

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!

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 any 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

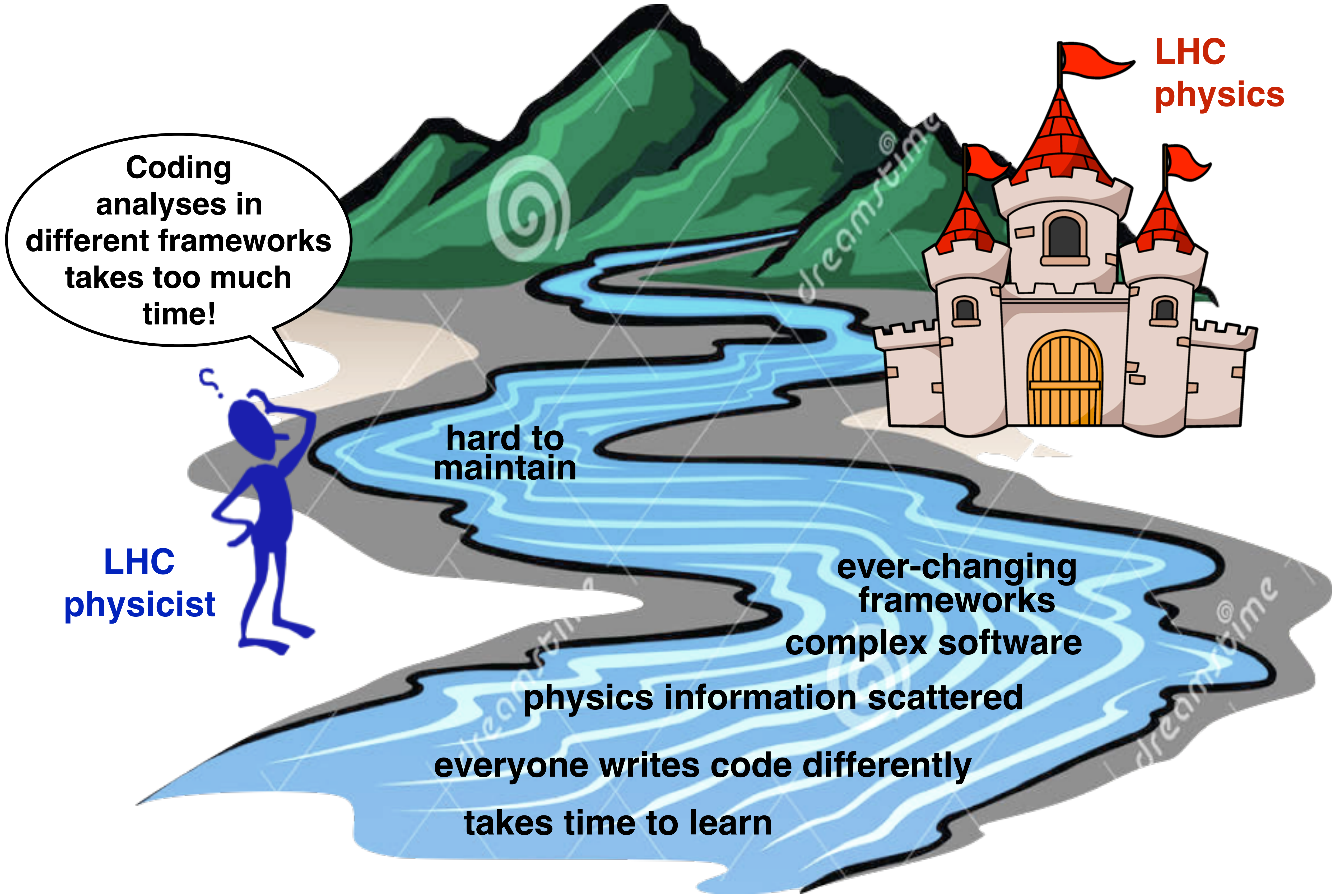
(optional: standard reports, visualizations)

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.

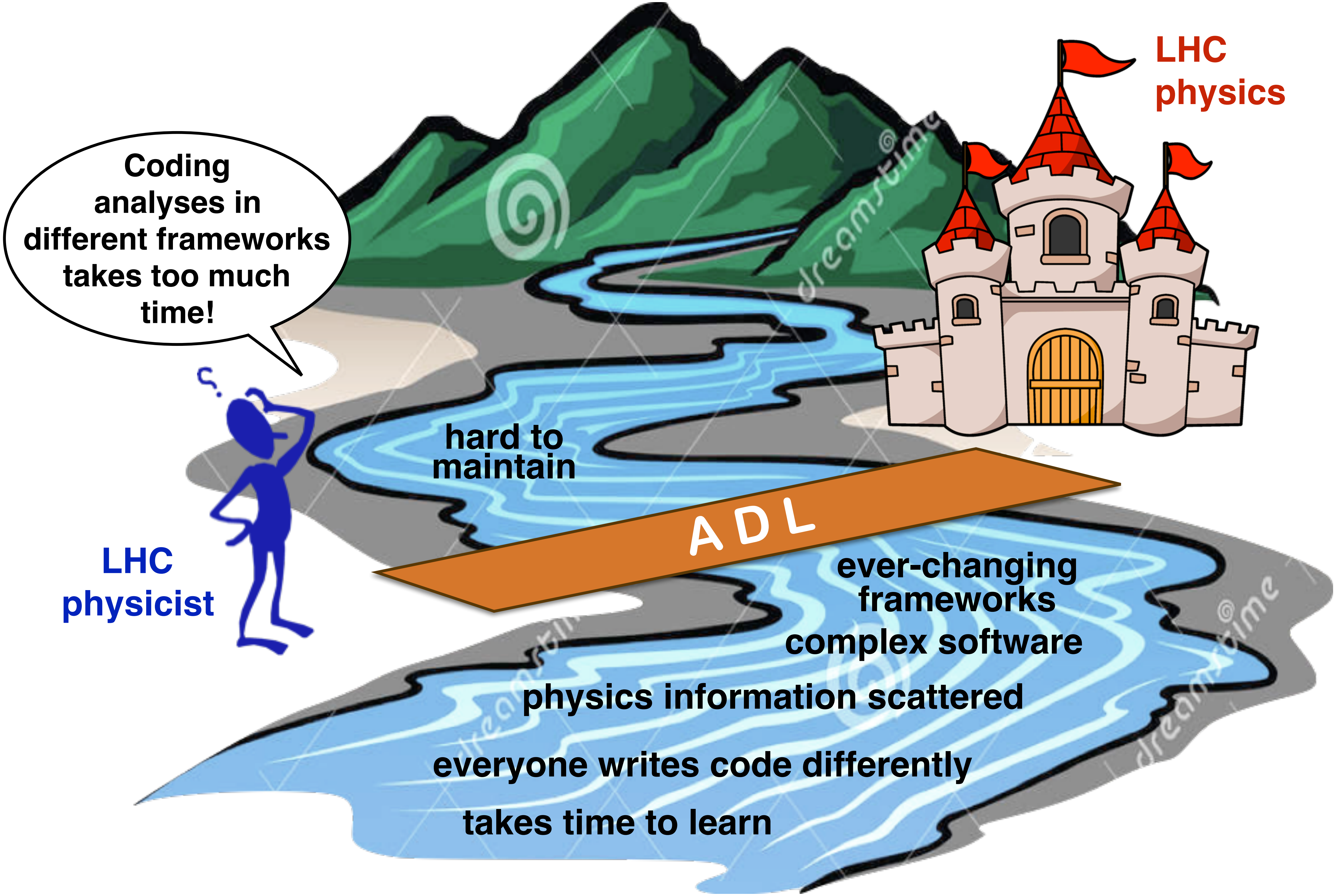
ADLs would help everyone

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



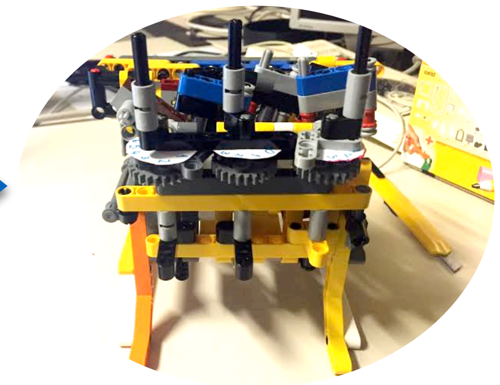
A specific ADL proposal

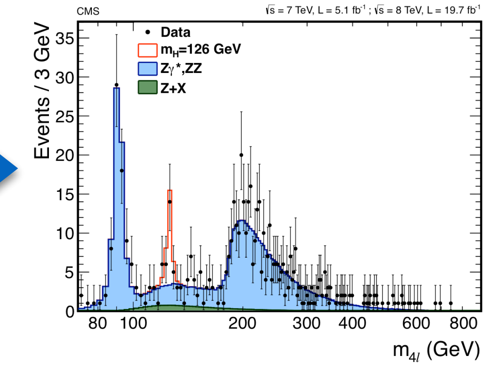
What could be a good way to systematically organize the components of an analysis?

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

S. Sekmen, G. Ünel

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

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.

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.

The ADL syntax

Both LHADA and CutLang ADLs comprise of blocks with a keyword value structure.

```
blocktype blockname
  # general comment
  keyword1 value1
  keyword2 value2
  keyword3 value3 # comment about value3
```

- Blocks allow a clear separation of analysis components.
- Operators and functions are used for evaluating values.

Disclaimers:

- ADL development is work in progress! Content constantly changes.
- What is shown here goes beyond the original LHADA proposal.
- CutLang and LHADA follow same principles but slightly differ in syntax.

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 : JET
# or object AK4jets
#       take 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]
```

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

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

Green: Implemented in (some) parser/interpreter tools

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 interpreters)
- 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.

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.
- 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.

Recent workshop to seriously start community-wide discussions.

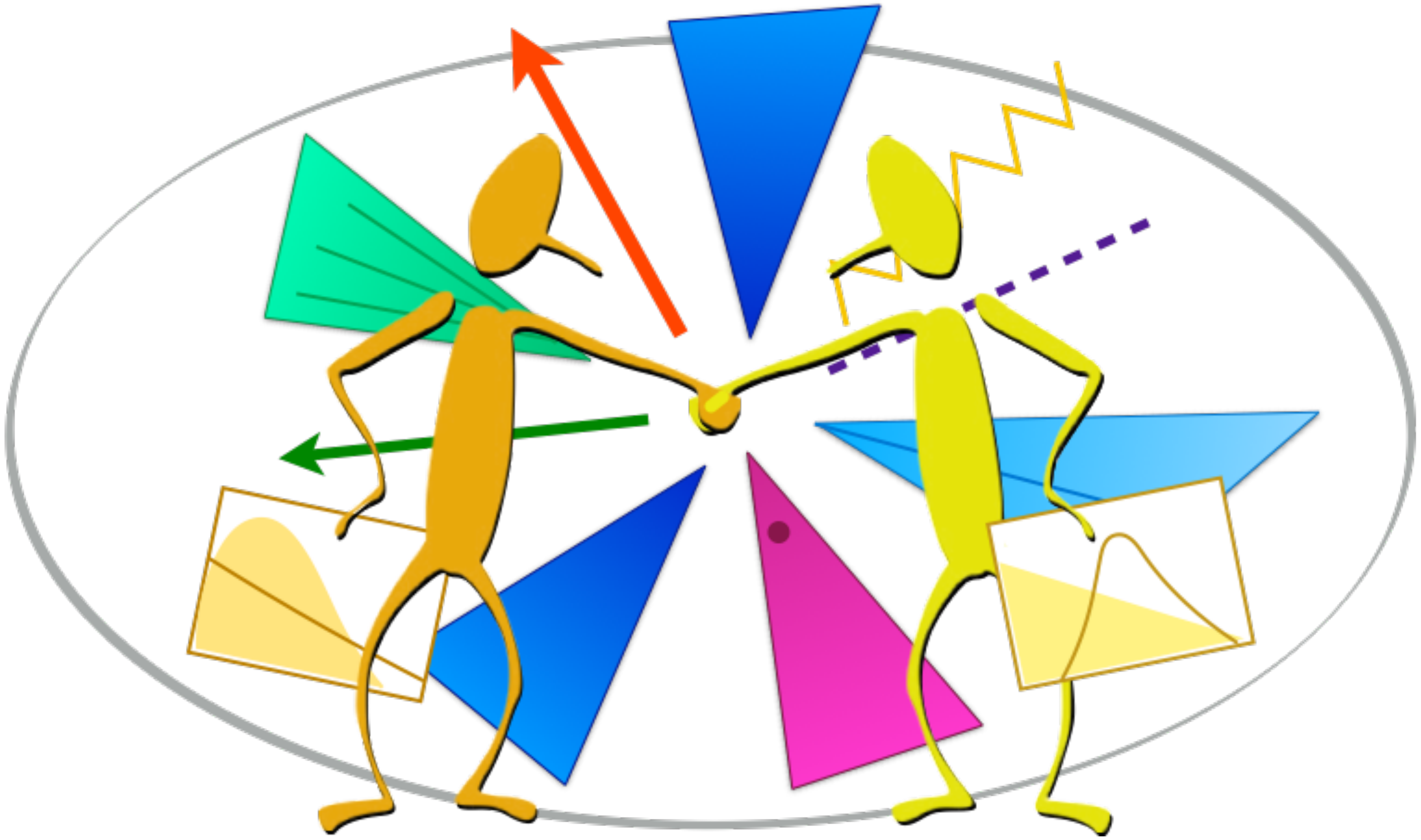
Participation by experimentalists, phenomenologists, computer scientists.

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/



To conclude

- An ADL would greatly facilitate BSM studies for the whole LHC community.
- Several prototypes have proven the feasibility of ADLs.
- 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!