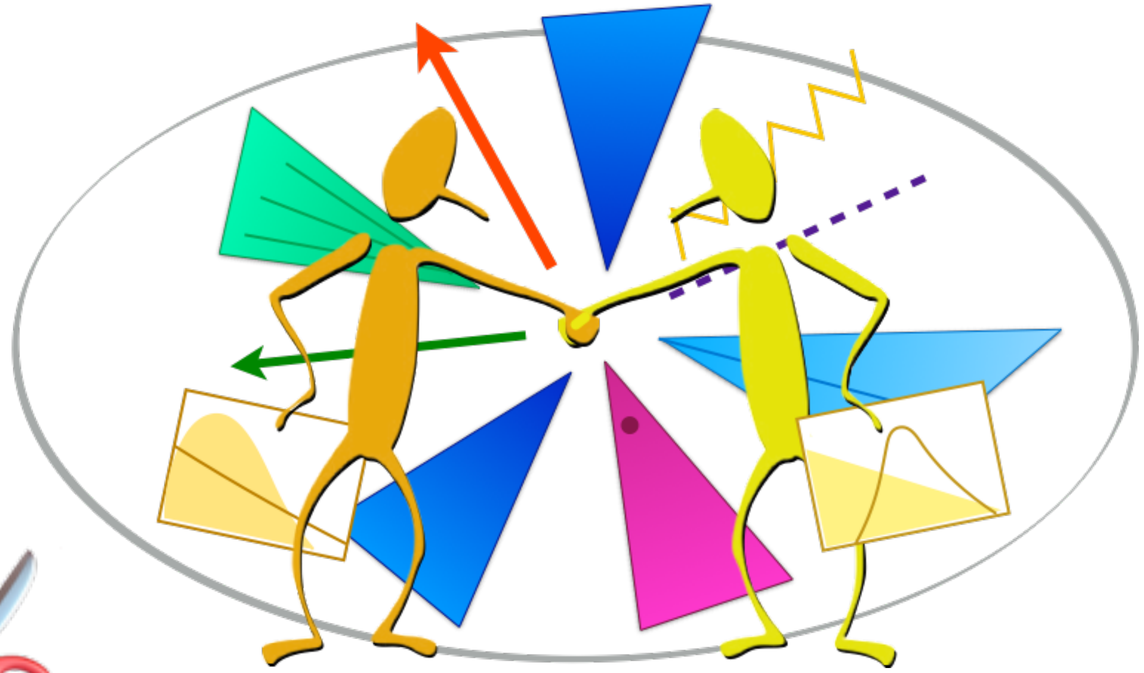
opendata
CERN



Feature demo: Open Data analysis with ADL & CutLang

Harrison Prosper (Florida State U.)
Sezen Sekmen (Kyungpook National U.)
Gökhan Ünel (U. of California, Irvine)

REMOTE CMS Open Data Workshop for Theorists at the LPC
30 Sep - 2 Oct 2020

Why this demo?

Yesterday, we have seen how to perform the H→ττ analysis within an analysis framework.

- The framework is written in C++ / Python code.
- Physics content and technical operations coexist and are handled together.

Could there be an alternative way which

- allows for more direct interaction with data and
- decouples physics information from purely technical tasks?

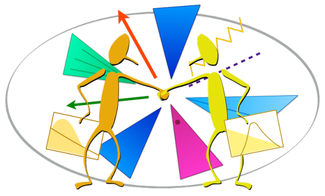
Performing physics analyses at LHC

What we mainly use at the LHC now:

- Analysis frameworks based on general purpose languages (GPL): A computer language that is broadly applicable across many application domains (FORTRAN, C++, Python, …).
  - Used for solving a very wide range of problems.

What we can consider for the future:

- Domain-specific language (DSL): a computer language specialized to a particular application domain (regular expressions, Makefile, SQL, …).
  - Designed to model how domain experts think about specific problems they wish to solve.
- Embedded DSL (eDSL): A DSL based on the syntax of a GPL (embedded SQL, LINQ, …)
- Declarative language: A language that expresses the logic of a computation without describing its control flow. Describes what needs to be done, but not how to do it.



Analysis description languages for LHC

Analysis Description Language (ADL) for the LHC is:

- A domain specific and declarative language capable of describing the physics content 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…

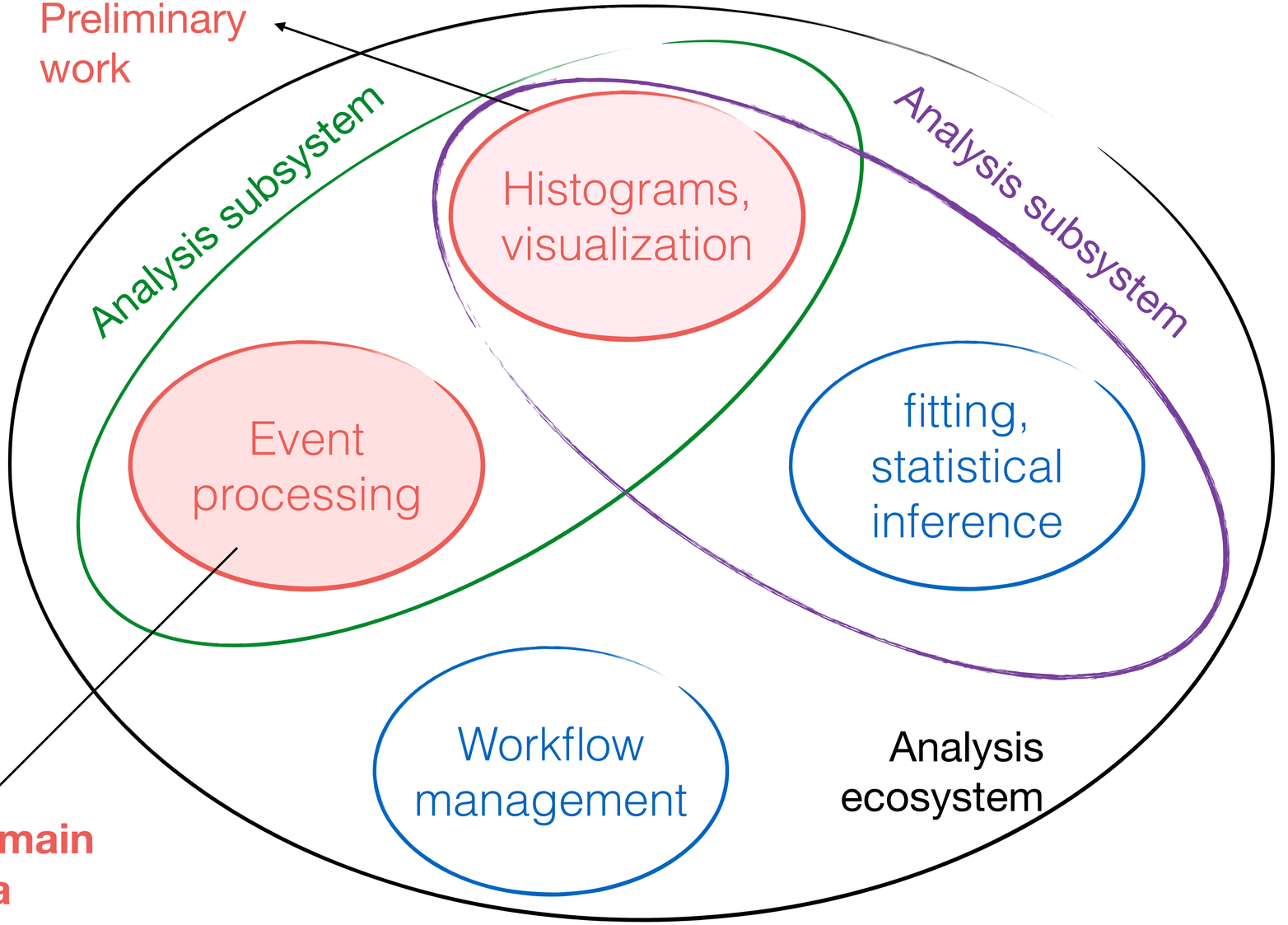
Earlier physics / analysis-related HEP formats/languages proved successful and useful:

- SUSY Les Houches Accord
- Les Houches Event Accord

The features of any analysis description language:
(Les Houches 2015 new physics WG report, arXiv:1605.02684, sec 16)

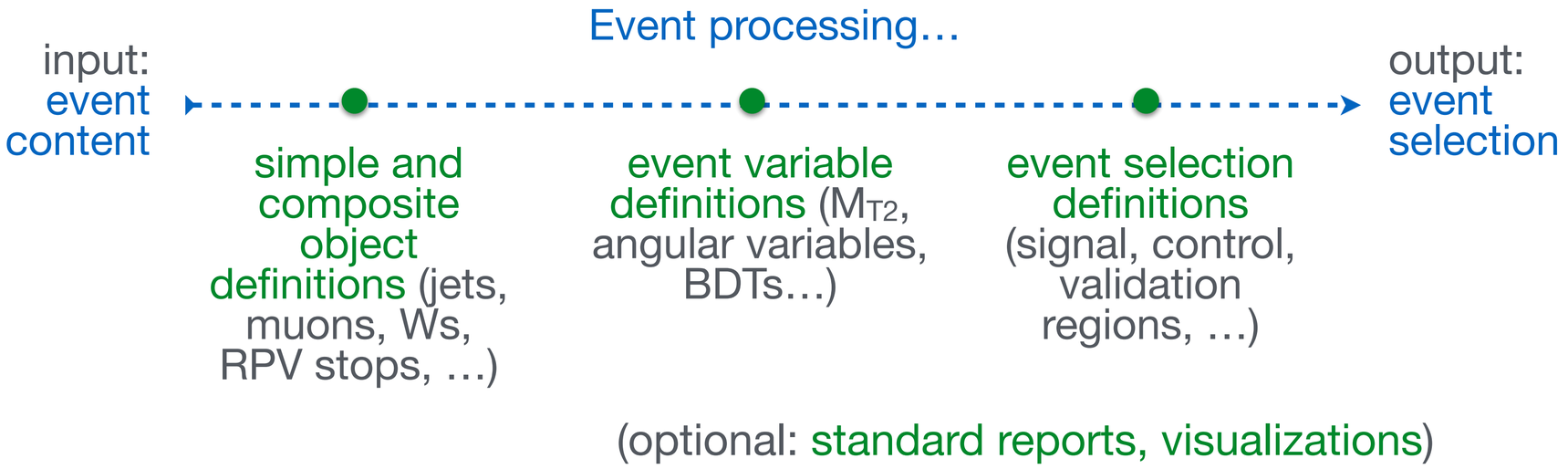
- Should be complete in content, demonstrably correct, easily learnable and sustainable.
- Can be easy to read, self-contained and analysis framework-independent.

What can an analysis description language describe?



Language scope

By construction, ADL is not designed to be general purpose; therefore, getting the right scope is key. The core of ADL would include:



Further operations with selected events (background estimation methods, scale factor derivations, etc.) can vary greatly, and thus may not be easily considered within a standard language scope.

LHADA/CutLang —> ADL

cern.ch/adl

ADL consists of

- a plain text file describing the analysis using an easy-to-read DSL with clear syntax rules.
- a library of self-contained functions encapsulating variables that are non-trivial to express with the ADL syntax (e.g. MT2, ML algorithms). Internal or external (user) functions.

- ADL database with 15 LHC analyses: https://github.com/ADL4HEP/ADLLHCanalyses

- Separate object, variable, event selection definitions into blocks with a keyword value structure, where keywords specify analysis concepts and operations.

```
blocktype blockname
  keyword1 value1
  keyword1 value2
  keyword3 value3 # comment
```

- Syntax includes mathematical and logical operations, comparison and optimization operators, reducers, 4-vector algebra and HEP-specific functions (dϕ, dR, …).

LHADA (Les Houches Analysis Description Accord): Les Houches 2015 new physics WG report (arXiv:1605.02684, sec 17)

CutLang: Comput.Phys.Commun. 233 (2018) 215-236 (arXiv:1801.05727), ACAT 2019 proceedings (arXiv:1909.10621)

ADL syntax: blocks, keywords, operators

Block purpose	Block keyword
object definition blocks	object
event selection blocks	region
analysis information	info
tables of results, etc.	table

Keyword purpose	Keyword
define variables, constants	define
select object or event	select
reject object or event	reject
define the mother object	take
define histograms	histo
applies object/event weights	weight
bins events in regions	bin

Operation	Operator
Comparison operators	> < => =< == != [] (include)][(exclude)
Mathematical operators	+ - * / ^
Logical operators	and or not
Ternary operator	condition ? truecase : falsecase
Optimization operators	~= (closest to) ~! (furthest from)
Lorentz vector addition	LV1 + LV2 LV1 LV2

ADL syntax rules: https://twiki.cern.ch/twiki/bin/view/LHCPhysics/ADL

ADL syntax: 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(), sqrt(), sin(), cos(), tan(), log(), …
- Collection reducers: size(), sum(), min(), max(), any(), all(),…
- HEP-specific functions: dR(), dphi(), deta(), m(), ….
- Object and collection handling: sort, comb(), union()…

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: M_{T2}, aplanarity, razor variables, …
- Non-analytic variables: Object/trigger efficiencies, variables/efficiencies computed with ML, …

ADL syntax rules: https://twiki.cern.ch/twiki/bin/view/LHCPhysics/ADL

Running analyses with ADL

Experimental / phenomenology analysis model with ADLs

- ADL file
- self-contained functions for complex variables
- events with any format

adl2tnm (transpiler) → **generic c++ analysis code**

CutLang (runtime interpreter)

transpiler / interpreter for any exp/pheno analysis framework (earlier prototypes exist for Rivet and Checkmate)

- Cutflows, counts
- Histograms
- Selected events
- Other results

Running analyses with ADL: adl2tnm

H. B. Prosper,
S. Sekmen

- Python transpiler converting ADL to C++ code.
- C++ code executed within the 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.
- Currently moving from proof of principle to the use of formal grammar building and parsing.

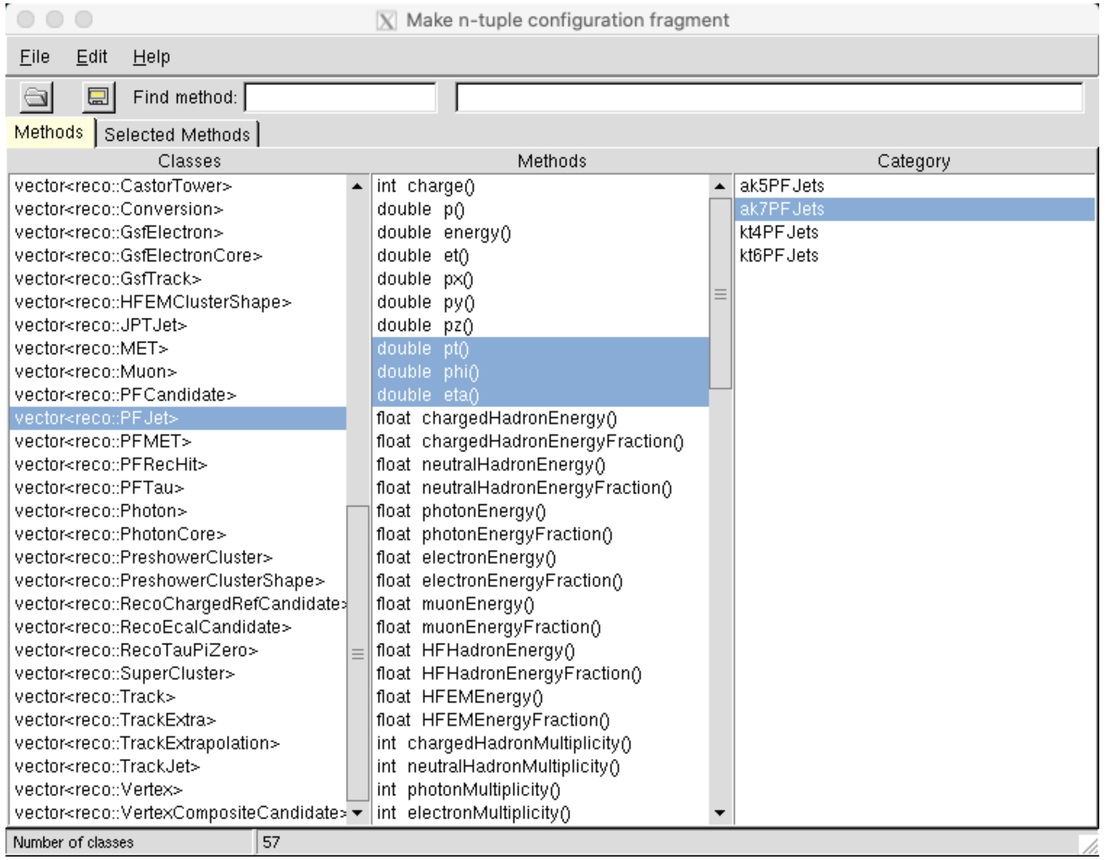
adl2tnm ref: Les Houches 2017 new physics WG report (arXiv:1803.10379, sec 23)

adl2tnm Github: https://github.com/hbprosper/adl2tnm

Interlude: Making ntuples with TheNtupleMaker (TNM)

H. B. Prosper,
S. Sekmen

- An automated system for creating ntuples with customized content from data in CMS EDM format.
- Works with the open data setup in ROOT5 (ROOT6 version under development).
- EDM data to be extracted is specified in simple python-based config file.
- Provides a GUI to select the EDM content and automatically write the to config file.
- Provides mechanisms to add user-defined variables to the ntuple and skim events.
- Self-documentating system: Automatically stores provenance information in the ntuple.
- Automatically generates an analyzer to read and analyze the resulting ntuples.



TNM Github and instructions for Open Data:
https://github.com/hbprosper/TheNtupleMaker

Running analyses with ADL: CutLang

G. Unel, B. Örgen,
A. Paul, N. Ravel,
S. Sekmen, J. Setpal,
A. M. Toon

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 open data, 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, variable and object definitions, histograms for each region in a separate TDirectory.

CutLang Github: https://github.com/unelg/CutLang

CutLang publications: arXiv:1801.05727, (arXiv:1909.10621)

Open data H→ττ analysis with ADL & CutLang

Install CutLang from github (link) and compile:
Requirements: ROOT6, lex&yacc or flex/bison which come by default with GNU.

```
git clone https://github.com/unelg/CutLang
cd CutLang/CLA
make clean; make -j 4
cd ../runs
```

We will run the open data outreach H→ττ analysis (record link). Get the ADL file for the analysis from github (link).

```
wget https://raw.githubusercontent.com/ADL4HEP/ADLLHCanalyses/master/CMS-OD-12350-Htautau/CMS-OD-12350-Htautau_CutLang.adl
```

Get the reduced NanoAOD files listed in the HTauTau analysis record, e.g. GluGluToHTauTau.root (link).

Run CutLang (Command: ./CLA.sh or ./CLA.py [inputrootfile] [inputeventformat] -i [adlfilename] -e [nevents])

```
./CLA.sh GluGluToHTauTau.root CMSNANO -i CMS-OD-12350-Htautau_CutLang.adl -e 100000
```

Cutflow results will be written on screen.
Cutflows, results and histograms will also be in histoOut-CMS-OD-12350-Htautau_CutLang.root

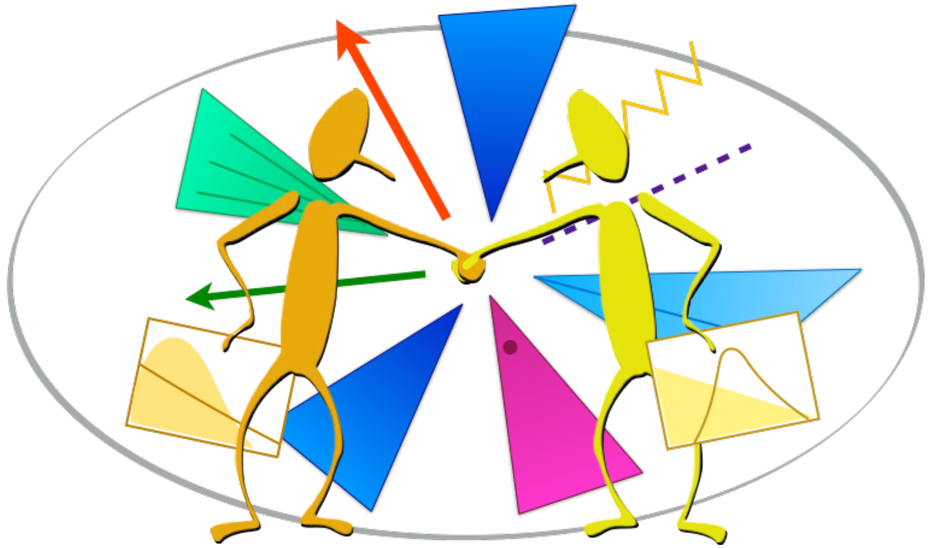
```
root -l histoOut-CMS-OD-12350-Htautau_CutLang.root
_file0->cd("baseline")   (Enter the directory for the "baseline" region)
_file0->ls()             (lists all histograms and the ADL file content for the region. cutflow histogram is default)
```

Why ADL for Open Data?

Many frameworks and tools exist for LHC analyses; so why use ADL for Open Data analysis?

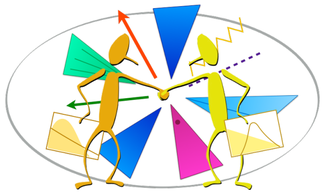
- ADL decouples physics analysis logic and algorithms from frameworks. Therefore, ADL:
  - allows the focus to be on analysis design rather than on the technicalities of programming;
  - does not require system or software level expertise to run analyses on data;
  - makes it easier to explore ideas and ask “what if” questions of data, and
  - with the low “barrier to entry”, ADL has the potential to broaden the pool of LHC data explorers.
- ADL is in the spirit of long-term analysis preservation, aligned with the spirit of Open Data.
  - Decoupling analysis description from analysis frameworks not only makes communication of the content of an analysis easier but also makes its preservation beyond the lifetime of frameworks and even beyond the lifetime of the general purpose languages in which they are written.

opendata
CERN



In summary

- ADL/CutLang is a practical way to design and perform LHC analyses and can be readily used with Open Data NanoAOD. All information on cern.ch/adl .
- ADL already covers the majority of standard physics content in an LHC analyses.
- Continuing progress! ADL keeps being refined. CutLang & adl2tnm are being developed into more practical user tools.
- We are ready to help with writing and running analyses with ADL/CutLang on Open Data. Please contact us for any questions or assistance !



Making this a community effort

- Discussions and work at the Les Houches PhysTeV workshops among phenomenologists and experimentalists (contributions in Les Houches 2015, 2017, 2019 proceedings).
- LHADA workshop, Grenoble, 25-26 Feb 2016
- LHADA workshop, CERN, 16-18 Nov 2016 (link)
- Discussions and activities within HSF data analysis WG and IRIS-HEP.
- 1st dedicated Analysis Description Languages for the LHC workshop (with experimentalists, phenomenologists, computing experts), Fermilab LPC, 6-8 May 2019 (link)
- 1st Data Analysis with ADL+CutLang School (3-7 Feb 2020), Istanbul (link)
- 2nd Analysis Description Languages workshop being planned for 2020 or 21.
- Gitter forum for discussions (link)

Workshop on

Analysis Description Languages for the LHC

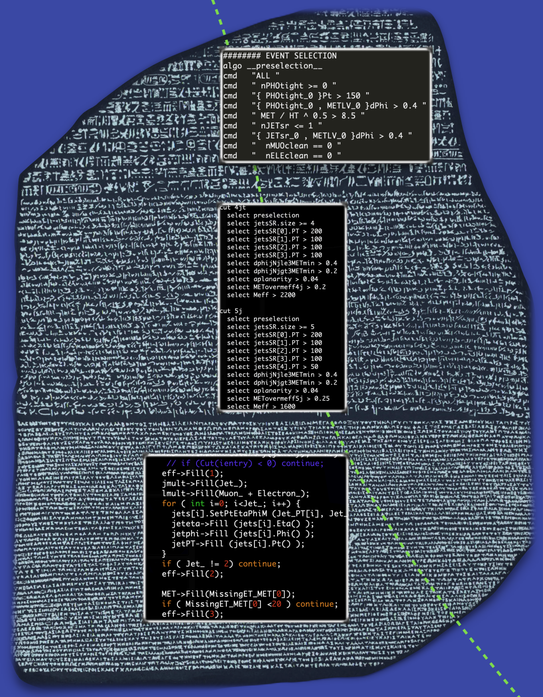
6-8 May 2019, Fermilab LPC

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

An analysis description language (ADL) is a human readable declarative language that unambiguously describes the contents of an analysis in a standard way, independent of any computing framework.

Adopting ADLs would bring numerous benefits for the LHC experimental and phenomenological communities, ranging from analysis preservation beyond the lifetimes of experiments or analysis software to facilitating the abstraction, design, visualization, validation, combination, reproduction, interpretation and overall communication of the contents of LHC analyses.

Several attempts were made recently to develop ADLs, and tools to use them, and an effort is underway to arrive at the core of a unified ADL.



In this workshop

(for experimentalists, phenomenologists and computing experts)

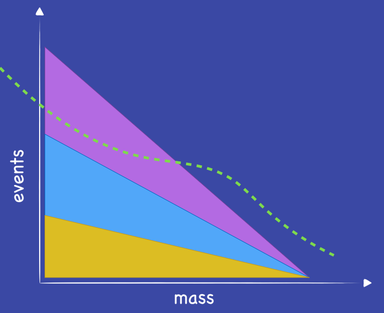
- The ADL concept
- Current examples: CutLang and LHADA
- Hands on exercises
- Language structure
- Parsing and interpreting methods
- Feasibility for experimental analyses
- Analysis preservation

Organizing committee:
Steve Mrenna (Fermilab)
Jim Pivarski (Princeton U.)
Harrison Prosper (Florida State U.)
Sezen Sekmen (Kyungpook Nat. U.)
Gökhan Ünel (U.C. Irvine)

LPC coordinators:
Cecilia Gerber (UIC)
Sergo Jindariani (Fermilab)

Local organization:
Gabriele Benelli (Brown U.)
Alexx Perloff (U. Colorado Boulder)
Marc Weinberg (Carnegie Mellon U.)

LPC events committee:
Gabriele Benelli (Brown U.)
Ben Kreis (Fermilab)
Kevin Pedro (Fermilab)



Recent dedicated workshop for a community-wide expert discussion.

Participation by experimentalists, phenomenologists, computer scientists.

- Overview of existing ADL efforts
- Language making tools

Extensive discussions on

- Why/where do we need an ADL?
- ADL physics scope and content
- ADL users' requirements
- What kind of ADL syntax we need?
- Parsing / interpreting methods
- ADLs for analysis preservation

Extensive information on indico:

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

ADLs for analysis preservation

- **C**ERN **A**nalysis **P**reservation group aims to preserve physics analyses to facilitate their future use (link)
- Working to build a stable, flexible, collaborative tool for physicists to capture and share their analyses.
- Preserved analyses can be used at large scale by tools like REANA (a system for reusable analysis execution on the cloud).
- ADL concept is highly in line with CAP goals and can be easily incorporated in the CAP system.
- CAP view: Ideally “One ADL to rule them all”. If not, an abstract layer that can bridge smaller use cases.
- CAP can provide database resources for ADL files, object structure types, object selections and external functions.