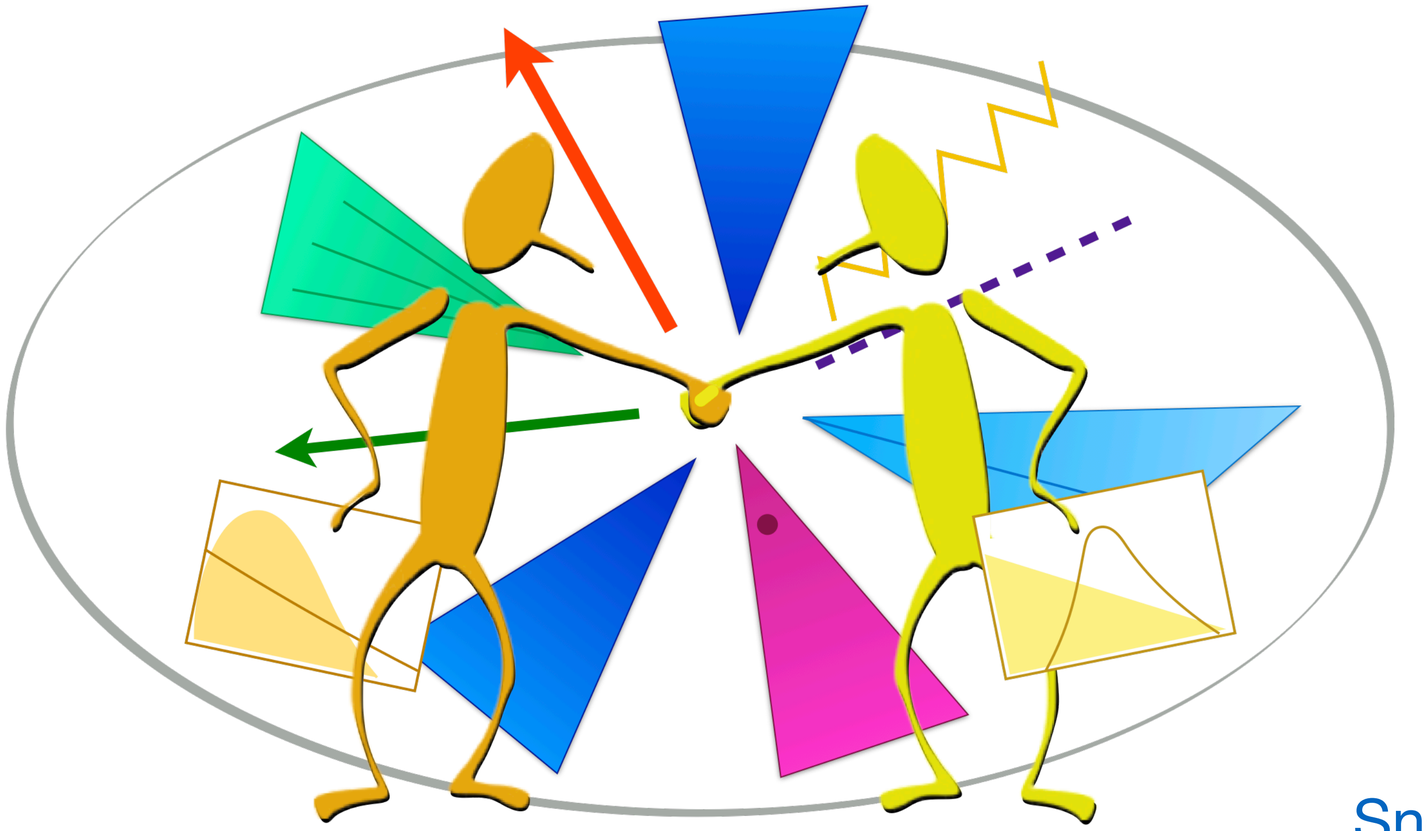


Analysis **D**escription **L**anguage: A DSL for HEP Analysis

Harrison Prosper (Florida State U.)
Sezen Sekmen (Kyungpook Nat. U.)
Gökhan Ünel (UC Irvine - ATLAS)
and the ADL/CutLang team

Snowmass Community Summer Study
16-27 July 2022, Seattle, USA

ADL white paper (CompF5, CompF7) : arXiv:2203.09886
ADL Documentation and references : cern.ch/adl

Analysis logic preservation

Thousands of analyses are designed by HEP experimentalists and phenomenologists.
—> Tremendous source of information to serve future physics studies.

We appreciate all analysis preservation efforts by experiments and organizations (e.g. CERN Analysis Preservation Group).

Preserving the physics content of analyses (the analysis logic) is particularly important. Current practices include:

- Papers: Hard to describe in full detail. Description detail non-uniform across analyses.
- Full analysis code: Hard to decipher <— Physics algorithm and technical operations are intertwined and handled together.
- Lightweight code / snippets: Better readability wrt full analysis code, but still based on general purpose languages (GPL), such as C++ and Python.

Our proposal: Improve the clarity and accessibility of analysis logic, and thereby its preservation, by using a domain-specific language tailored for HEP analyses.

Analysis description language for HEP

Analysis Description Language (ADL) is a declarative domain specific language (DSL) that describes the physics content of a HEP analysis in a standard and unambiguous way.

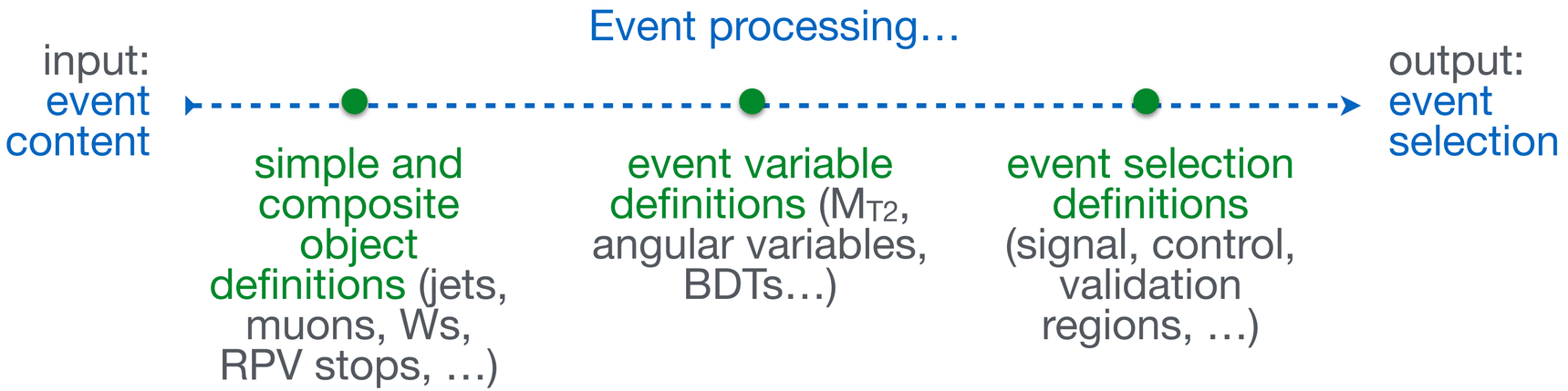
- External DSL: Custom-designed syntax to express analysis-specific concepts. Reflects conceptual reasoning of particle physicists. Focus on physics, not on programming.
- Declarative: States what to do, but not how to do it.
- Easy to read: Clear, self-describing syntax.
- Designed for everyone: experimentalists, phenomenologists, students, interested public…

ADL is framework-independent —> Any framework recognizing ADL can perform tasks with it.

- Decouples physics information from software / framework details.
- Multi-purpose use: Can be automatically translated or incorporated into the GPL / framework most suitable for a given purpose, e.g. exp. analysis, (re)interpretation, analysis queries, …
- Easy communication between groups: exp., pheno, referees, students, public, …
- Easy preservation of analysis logic.

ADL scope

- Event processing: Priority focus.



- Analysis results, i.e. counts and uncertainties: Available
- Histogramming: Available.
- Systematic uncertainties: Within the scope. Syntax design in progress.
- Operations with selected events, e.g. background estimation, scale factor derivation: Very versatile. Not yet within the scope.

The ADL construct

cern.ch/adl

ADL consists of

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

- ADL file consists of blocks separating object, variable and event selection definitions. Blocks have a keyword-instruction structure.
  - keywords specify analysis concepts and operations.

```
blocktype blockname
  keyword1 instruction1
  keyword1 instruction2
  keyword3 instruction3 # comment
```

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

ADL syntax with usage examples: link

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), Front. Big Data 4:659986, 2021
Several proceedings for ACAT and vCHEP

A very simple analysis example with ADL

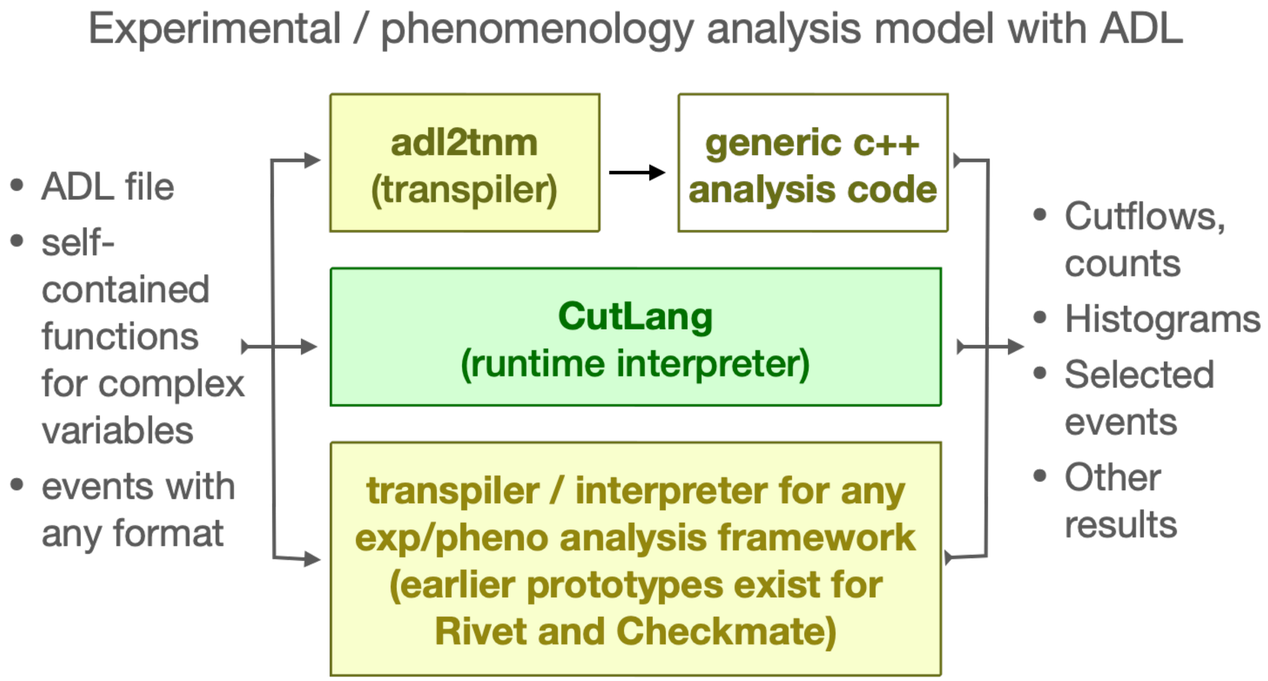
```
# OBJECTS
object goodMuons
  take muon
  select pT(muon) > 20
  select abs(eta(muon)) < 2.4

object goodEles
  take ele
  select pT(ele) > 20
  select abs(eta(ele)) < 2.5

object goodLeps
  take union(goodEles, goodMuons)

object goodJets
  take jet
  select pT(jet) > 30
  select abs(eta(jet)) < 2.4
  reject dR(jet, goodLeps) < 0.4
```

```
# EVENT VARIABLES
define HT = sum(pT(goodJets))
define MTl = Sqrt( 2*pT(goodLeps[0]) * MET*(1-cos(phi(METLV[0]) - phi(goodLeps[0]) )))

# EVENT SELECTION
region baseline
  select size(goodJets) >= 2
  select HT > 200
  select MET / HT <= 1

region signalregion
  baseline
  select Size(goodLeps) == 0
  select dphi(METLV[0], jets[0]) > 0.5

region controlregion
  baseline
  select size(goodLeps) == 1
  select MTl < 120
```

ADL implementations of some LHC analyses: https://github.com/ADL4HEP/ADLLHCanalyses

Running analyses with ADL

Multipurpose & framework-independent: Can be translated / integrated into any GPL / framework.



CutLang: C++ runtime interpreter for ADL.

- Formal grammar parsing by Lex & Yacc.
- Based on ROOT. Reads various TTree-like formats. NanoAOD, Delphes, open data, etc.
- Jupyter kernel exists.
- Used in several analyses for ATLAS, FCC, reinterpretation (via SModelS), open data, …
- Outputs cutflows, histograms, events, analysis description, i.e. provenance.

Physics information is fully contained in ADL. Current compiler infrastructures can be easily replaced by future tools / GPLs / frameworks.

CutLang Github repository: https://github.com/unelg/CutLang
Comput.Phys.Commun. 233 (2018) 215-236 (arXiv:1801.05727),
Front. Big Data 4:659986, 2021 (arXiv:2101.09031),
Several proceedings for ACAT and vCHEP

ADL for analysis preservation

ADL is designed in the spirit of long-term analysis preservation:

- Decoupled from analysis frameworks, portable, self-documenting, modular, domain-specific syntax, uniform structure.

ADL can be easily incorporated in the CERN Analysis Preservation (CAP) system:

- The CAP team has been following ADL since the beginning, and is very supportive.

Planning to build 3 web-based, searchable and citable databases for ADL content:
Discussions ongoing with the CAP team.

- ADL analyses database: Host ADL files of implemented analyses
- ADL objects database: Host object definitions written in ADL (e.g. 2016 tight isolated electrons for CMS leptonic SUSY analyses, boosted medium Higgs from ATLAS, etc.)
- ADL functions database: Host external functions of non-trivial or non-analytical variables (ML discriminants, complex kinematic variables, efficiencies, etc.)

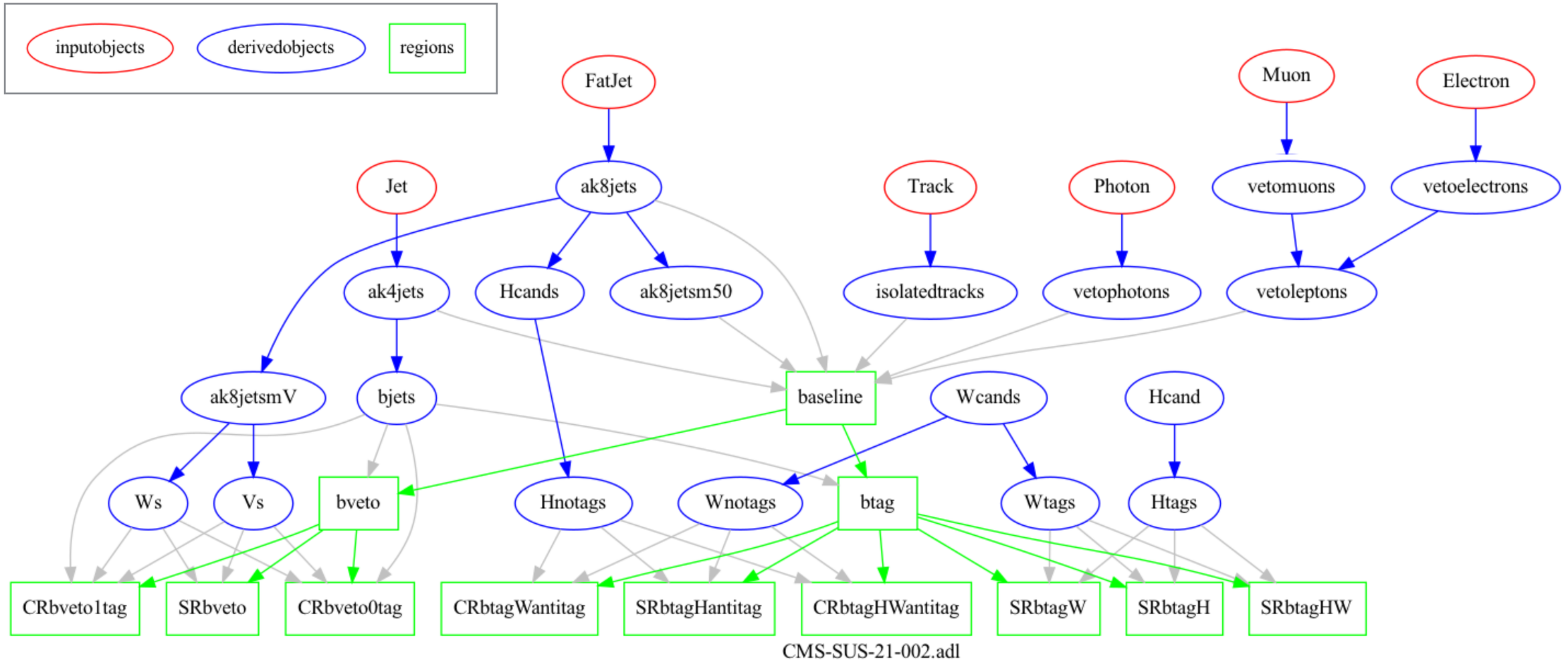
ADL helps to design and document a single analysis in a clear and organized way.

But its distinguishing strength is in navigating and exploring the multi-analysis landscape.

Uses for analyses preserved via ADL

- Use existing analyses to design new ones: Answer questions such as *“Which final states did the existing analyses look at?”* ; *“Which final states are unexplored?”* ; *“How much overlap exists between my analysis and the existing ones?”*
- Use existing objects: Directly implement in a new analysis, compare analyses choices, work with definition of the same object in different data tiers.
- Visualize & review analyses: Build graphs and tables from analyses using automated tools.
- Query analysis or object databases: Answer questions such as *“Which analyses require missing ET > at least 300?”* ; *“Which analyses use b-jets tagged with a certain criterion?”* ; *“Which muons use isolation?”* via automated query tools.
- Compare / combine analyses: Determine analysis overlaps, identify disjoint analyses or search regions; find the feasible combinations with maximal sensitivity; automate large scale combinations of analyses.
- Education: Provide a learning database for younger colleagues.
- Reinterpretation: p12.

Auto-generated graph of an ADL analysis (using graphviz)



arXiv:2205.09597: CMS Search for Electroweak SUSY in WW, WZ and WH hadronic final states

ADL for reinterpretation

ADL allows practical exchange of experimental analysis information with the pheno community.

- Clear description of the complete analysis logic.
- Enables straightforward adaptation from experiments to public input event formats.
  - Biggest difficulty is in reproducing an analysis is adapting object definitions:
    In ADL, e.g. just swap experimental object IDs with numeric efficiency maps.
  - Event selections stay ~the same (can swap trigger selections with efficiencies)
- Generic structure available for expressing analysis output in the ADL file:
  Data counts, BG estimates, signal predictions —> counts, uncertainties, cutflows.
  - Running CutLang puts preexisting results in histograms with the same format as the run output. —> Direct comparison of cutflows, limit calculations.
- ADL could facilitate providing information on analysis results to HEPDATA or similar platforms.

What is next for ADL?

Expand practical use and physics applications:

- Launch an analysis implementation and validation campaign for preservation and reinterpretation purposes; organize ADL hackathons; perform physics studies for future colliders with ADL/CutLang.

Extend the language scope:

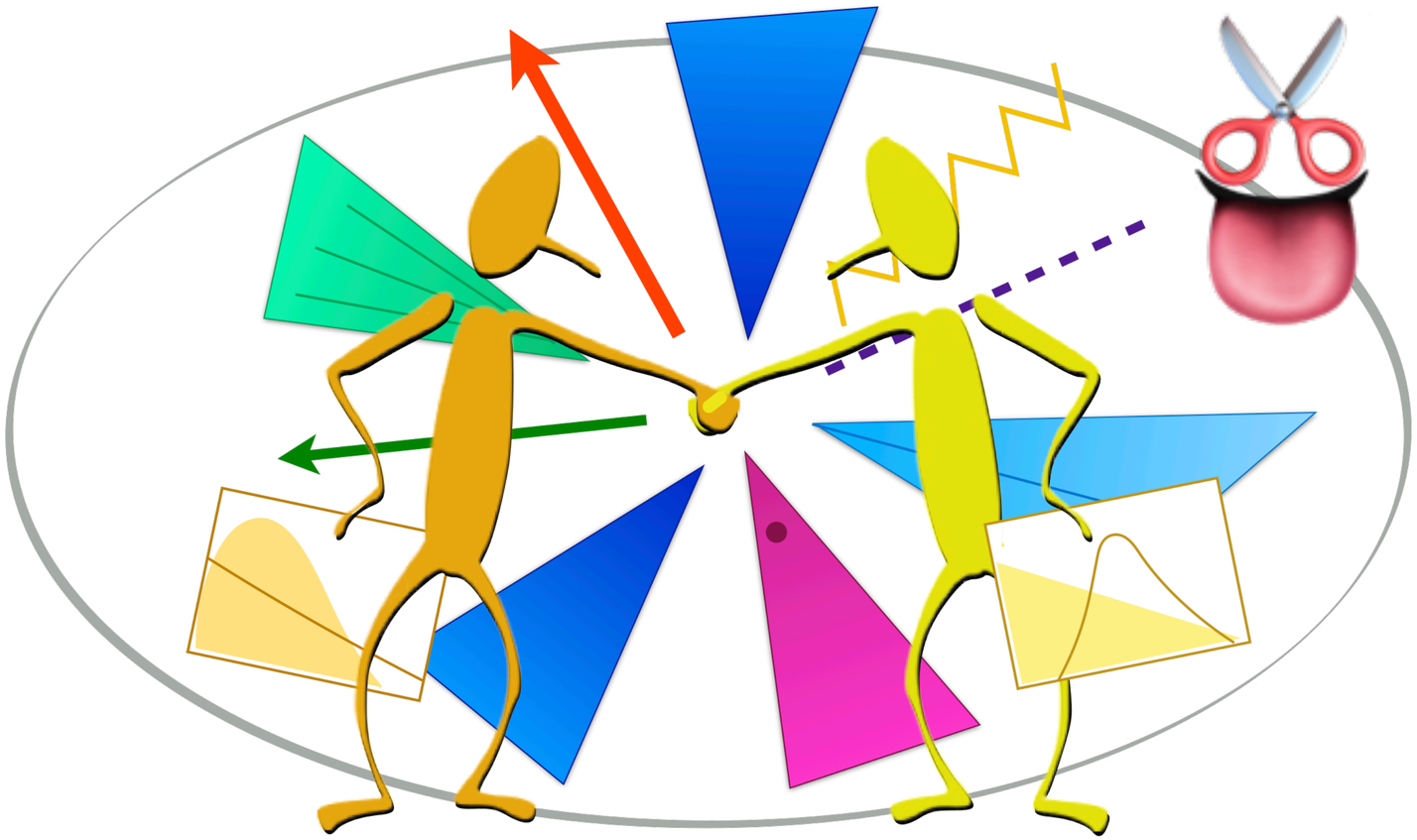
- ADL can already describe a large number of published analyses, but needs various extensions towards a domain-complete language, e.g. enhanced object handling, systematic uncertainties.

Advance the infrastructures and auxiliary tools:

- Build more formal compiler infrastructures with a layered design; have automated syntax verification; provide a toolkit for static analysis, queries, visualization; provide a setup for differentiable programming, …

Analysis preservation:

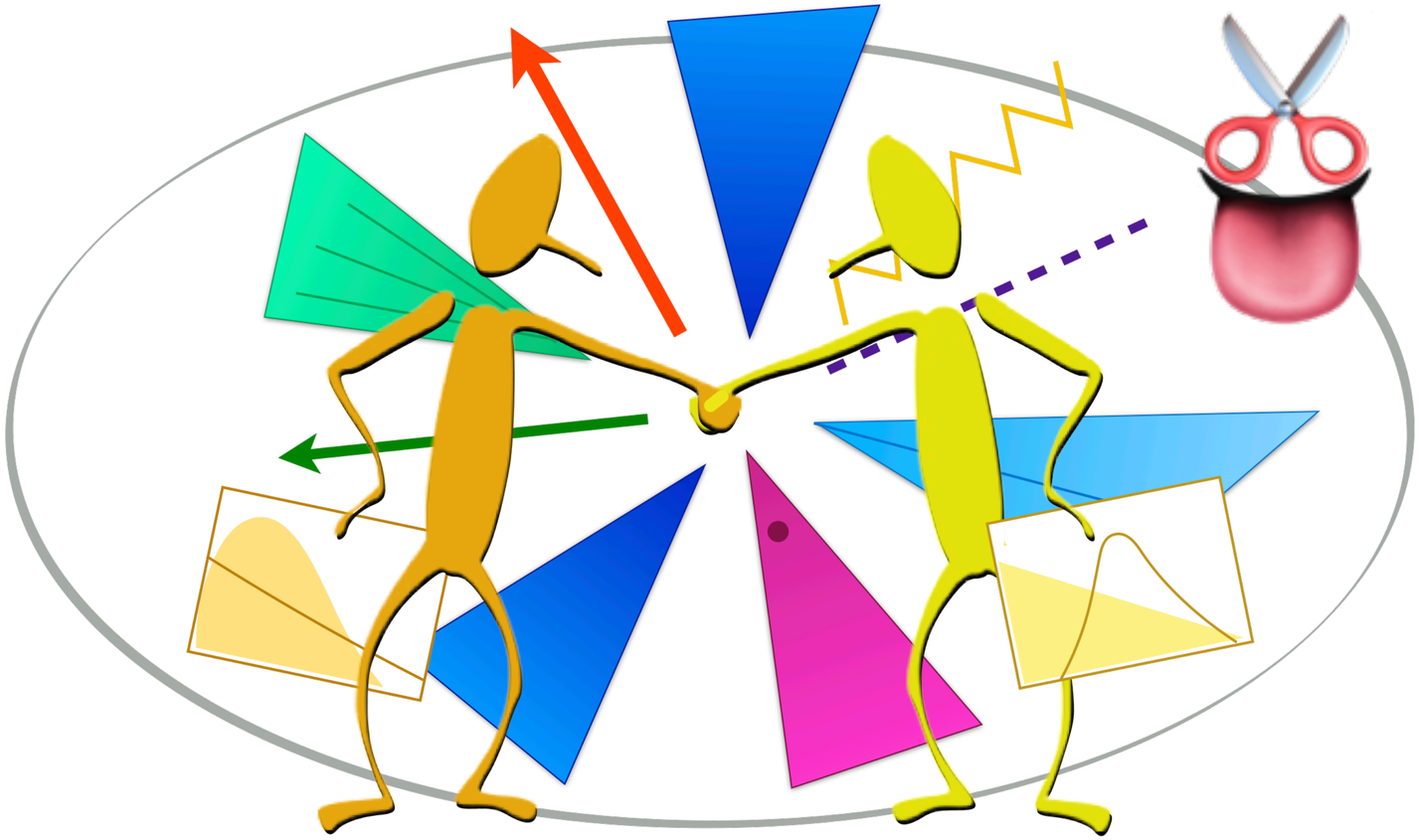
- Work towards building formal databases for ADL analyses, objects and external functions.



To conclude:

- ADL is an emerging approach with great potential for analysis preservation and reinterpretation.
- ADL and runtime interpreter CutLang so far show the feasibility of this approach.
  - ADL syntax and tools are under constant development.
- We invite the HEP community to explore ADL/CutLang and provide feedback (mattermost channel).

ADL is a community effort !
Everyone is welcome to join the development of the language and tools.



Extra slides

ADL syntax: main blocks, keywords, operators

Block purpose	Block keyword
object definition blocks	object
event selection blocks	region
analysis or ADL information	info
tabular information	table

Keyword purpose	Keyword
define variables, constants	define
select object or event	select
reject object or event	reject
define the mother object	take
apply weights	weight
bin events in regions	bin, bins
sort objects	sort
define histograms	histo
save variables for events	save

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

Syntax also available to write existing analysis results (e.g. counts, errors, cutflows…).

Syntax develops further as we implement more and more analyses.

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

ADL syntax: functions

Standard/internal functions: Sufficiently generic math and HEP operations could 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: union(), comb()…

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 and accessible by compilers via a database.

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

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

Physics with ADL / CutLang

Designing new analyses:

- Experimental analyses:
  - 2 ATLAS EXO analyses on VLQ and VLL ongoing with a customized version of CL.
- Phenomenology studies:
  - E6 isosinglet quarks at HL-LHC & FCC (Eur Phys J C 81, 214 (2021))

Implementing existing analyses:

ADL analysis database with ~15 CMS & ATLAS analyses:
https://github.com/ADL4HEP/ADLLHCanalyses
(another ~15 being implemented including CMS Snowmass analyses)

Using analysis implementations:

- Building a validation infrastructure for implemented analyses in collaboration with the SModelS team (W. Waltenberger et. al.)
  - Mass produce signals, run analyses, compute limits, compare with existing limits
- Reinterpretation studies:
  - Integrated ADL/CutLang into the SModelS framework for calculating efficiency maps.
- Static analysis for nalysis queries, comparisons, combinations: (Which analyses require HT > at least 500? Which have leptons? Which analyses/regions are disjoint?)
  - Automated tools under development arXiv:2002.12220, sec 17

CutLang interpreter and framework

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

CutLang runtime interpreter:

- No compilation of the analysis content. Directly runs the ADL file on events.
- CutLang itself is written in C++, works in any modern Unix environment.
- Based on ROOT classes for Lorentz vector operations and histograms.
- ADL syntax parsing by Lex & Yacc.
- Includes parsing of various standard and external functions. <— Still requires automation.

CutLang framework: interpreter + tools
Event and external function inputs:

- Input events: via ROOT files.
  - multiple input formats: Delphes, CMS NanoAOD, ATLAS/CMS Open Data, LVL0, FCC. More can be ~easily added.
  - All event types converted into predefined particle object types recognized by the runtime interpreter —> can run the same ADL file on different input types.

  <— working on automated matching of input object attributes in ADL & ROOT files.
- External functions input: Functions currently stored within the framework.

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

CutLang publications: arXiv:1801.05727, arXiv:1909.10621
arXiv:2101.09031

CutLang interpreter and framework

CutLang framework: interpreter + tools
Run output:

- Results output via ROOT files. Includes:
  - ADL file (for provenance tracking)
  - A TDirectory per region including cutflows, bins and user-defined histograms.
  - If existing results (counts, errors, cutflows) are specified in the ADL file, CutLang can record those in histograms similar to its run results.
- Saving selected event content at any analysis stage (e.g. for ML training)
  - Selected quantities can be saved into a CSV files after an event selection.

Auxiliary tools:

- Datasets, cross sections, sample-specific weights input by text-based config files.
- Some scripts for combining output files and plotting are also available.

All these are native to CutLang framework and currently outside the scope of ADL.
Work in progress.