ADL/CutLang developments towards large scale (re)interpretation

(Re)interpretation of the LHC results for new physics

Gökhan Ünel (UC Irvine & Boğaziçi U.)
for
the ADL/CutLang team
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.
The ADL construct

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
  - keyword2 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

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
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

Burak Şen (Middle East Tech. U.)
Running analyses with ADL

Once an analysis is written, it needs to run on events.

ADL is **multipurpose & framework-independent**: It can be translated / integrated into any language or framework for analysis tasks:

Physics information is fully contained in ADL. Current compiler infrastructures can be easily replaced by future tools / languages / frameworks.
CutLang (CL) runtime interpreter and framework

CutLang runtime interpreter:

- No compilation. User writes an ADL file and runs CutLang directly 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 parsing by Lex & Yacc.

CutLang framework: interpreter + tools

- 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. —> can run the same ADL file on different input types.
- Includes many internal functions.
- Output in ROOT files: ADL file, cutflows, bins and histograms; event pass/fail ntuples for each region in a separate directory
- Available in Docker, Conda, Jupyter (via Conda or binder). (win/lin/mac + portables)

CutLang Github repository: https://github.com/unelg/CutLang
ADL/CL scope

• Event processing: *Priority focus!*
• Analysis results, i.e. counts and uncertainties: Available
• Histogramming: Available => HistoSets, 1D, 2D, variable width...
• Systematic uncertainties: ATLAS type syntax now available. Following HEP community discussions on how to express systematics.
• Data/MC comparison, limits: Within the scope, implementation being tested.
• Operations with selected events, e.g. background estimation, scale factor derivation: Very versatile. Not yet within the scope.

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

BONUS: Library functions guaranteed to be bug free

**WYGIWYS** analysis, no double counting, correct sorting, $\chi^2$ evaluation, combinatorics, unions...

Its distinguishing strength is in navigating and exploring the multi-analysis landscape.
Versatile uses of ADL - I

- Analysis design (experimental or pheno):
  - Quick prototyping.
  - Simultaneous test of numerous selection options in a self-documenting way.
  - Easy comparison with existing analyses: “Was my phase space already covered?”

- Objects handling:
  - Easy reuse in new analyses.
  - Compare object definitions within or between analyses.
  - Compare definitions in different input data types.

- Analysis visualization:
  - Build analysis flow graphs and tables from analyses using static program analysis tools.

- Communication:
  - Between analysis team members (easy synchronization); with reviewers; between teams; between experiments or exp. and pheno.
Versatile uses of ADL - II

• Analysis preservation: Queryable databases for analysis logic and objects.

• Queries in analysis or object databases: Use static analysis tools to answer questions such as
  • “Which analyses require MET > at least 300?”; “Which use b-jets tagged with criterion X?”;
    “Which muons use isolation?”

• Analysis comparisons / combinations:
  • Determine analysis overlaps, identify disjoint analyses or search regions;
  • Automate finding the combinations with maximal sensitivity; phase space fragmentation.

• Education:
  • Provide a learning database for students (and everyone).
  • Easy entry to running analyses (several schools & trainings organized).

• Reinterpretation: Next page.

• … … and how would YOU use it?
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.**
  - Repurpose ADL files: swap experimental object definition blocks with simplified object blocks based on numerical object ID / tagging efficiencies.
  - Event selections stay almost the same: can swap trigger selections with trigger efficiencies
  - Efficiencies can be implemented via hit-and-miss function (see next slide)
- **Generic syntax 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.
  - Could facilitate communicating information to/from HEPDATA or similar platforms.
Object efficiencies

- Object efficiencies versus (multiple) attributes and their uncertainties provided by the experiments can be recorded in the ADL file via tables.

- CutLang can apply these efficiencies to input objects via the hit-and-miss method, for selecting objects with the efficiency probability.
  - both at object selection and event selection level.

object bjets
  take jets
  select abs(flavor(jets)) == 5
  select applyHM( btagdeepCSVmedium( Pt(jets) ) == 1)
Counts and cutflows

• Record cutflow values from the experiment.
• Run CL on local sample and obtain cutflow.
(same histogram format)
• Compare with experiment.

- Record data and BG estimates from the exp.
- Run CL and obtain signal predictions.
(same histogram format)
- Compute limits.

`countsformat sigpone`

process Tttttt1900200, "Ttttt 1900 200", stat
process Tbbbb1800200, "Tbbbb 1800 200", stat
process Tqqqq1300100, "Tqqqq 1300 100", stat
process TqqqqV1800100, "TqqqqV 1800 100", stat

`countsformat sigtwo`

process Tttttt1300100, "Ttttt 1300 100", stat
process Tbbbb1300110, "Tbbbb 1300 110", stat
process Tqqqq1200100, "Tqqqq 1200 100", stat
process TqqqqV1400110, "TqqqqV 1400 110", stat

# preselection region

region presel

select all

counts sigone 100.0 +/- 0.8 , 100.0 +/- 0.5 , 100.0 +/- 0.0 , 100.0 +/- 0.5
counts sigtwo 100.0 +/- 0.0 , 100.0 +/- 0.1 , 100.0 +/- 0.1
select size(jets) >= 2

counts sigone 100.0 +/- 0.8 , 100.0 +/- 0.5 , 100.0 +/- 0.0 , 100.0 +/- 0.5
counts sigtwo 99.3 +/- 0.1 , 99.6 +/- 0.1 , 100.0 +/- 0.1
select sizethree 99.3 +/- 0.2 , 98.8 +/- 0.5 , 99.1 +/- 0.5
counts sigfour 99.5 +/- 0.0 , 95.4 +/- 0.1 , 97.8 +/- 0.2
select HT > 300

counts sigone 100.0 +/- 0.8 , 100.0 +/- 0.5 , 100.0 +/- 0.0 , 100.0 +/- 0.5
counts sigtwo 98.1 +/- 0.4 , 74.8 +/- 0.5 , 82.0 +/- 0.3 , 94.6 +/- 0.4
counts sizefour 98.7 +/- 0.4 , 98.3 +/- 0.5 , 98.9 +/- 0.6
counts sigfour 72.2 +/- 0.3 , 58.2 +/- 0.3 , 83.0 +/- 0.4
select MHT > 300

counts sigone 85.5 +/- 2.7 , 86.8 +/- 1.9 , 77.1 +/- 0.5 , 83.0 +/- 2.1
counts sigtwo 13.8 +/- 0.4 , 19.9 +/- 0.5 , 21.2 +/- 0.4 , 22.2 +/- 0.7
counts sizehree 74.5 +/- 1.2 , 79.6 +/- 1.4 , 88.1 +/- 1.4
counts sigfour 9.2 +/- 0.2 , 13.6 +/- 0.2 , 31.3 +/- 0.5

region searchbins

presel

# Table 3, 1-10
bin MHT [ ] 300 350 and HT [ ] 300 600 and size(jets) [ ] 2 3 and size(bjets) == 0

counts bgests 38878 +/− 320 +/− 500, 89100 +/− 200 +/− 2000, 1800 +/− 1000 +/− 1200 +/− 800

counts results 129800 +/− 1100 +/− 2800 , 130718

bin MHT [ ] 300 350 and HT [ ] 600 1200 and size(jets) [ ] 2 3 and size(bjets) == 0

counts bgests 2760 +/− 61 +/− 39 , 4970 +/− 50 +/− 150 , 330 +/− 180 +/− 160

counts results 8060 +/− 200 +/− 220 , 7820

bin MHT [ ] 300 350 and HT >= 1200 and size(jets) [ ] 2 3 and size(bjets) == 0

counts bgests 181 +/− 17 +/− 3 , 308 +/− 12 +/− 18 , 62 +/− 34 +/− 27
We are implementing a variety analyses in ADL and testing them with CutLang. Main target of analysis implementations so far was refining ADL syntax and CL capabilities. We now launch a large scale analysis validation effort to make analyses available for reinterpretation studies. Training 8 students for the job. All validation material / results to be hosted at a github repository of ADL analyses.

Use **SModelS Efficiency Map Creator** for validation:

- Developed by Wolfgang W. to produce selection efficiency maps on SMSs for input to SModelS. can be used to validate analysis implementations by comparing to experimental results. EM-creator was adapted to work with ADL/CutLang by Wolfgang W. and Jan Mrozek. Final step: Efficiency maps. Limit calculation currently within SModelS.
• Currently works with SUSY-like SMSs. Plan to generalize to full models.
• Configurable user interface: can specify which models and mass points to produce, which steps to run, which output to save.
• Works in Vienna computing cluster & Korea KNU T3 cluster. Parallelized with HTCondor.
• Tests and refinements ongoing towards release for public use.

Example validation for CMS-SUS-19-006: "Inclusive search with Jets + MET".

Changgi Huh, Junghyun Lee, Sezen Sekmen (Kyungpook Nat. U.)
ADL/CL and LHC Open Data

- ADL/CL can be used to run analyses with ATLAS (educational) and CMS (research) open data.
- Use related to reinterpretation: Provide capability to re-optimize and re-run recasted analyses from ADL database to maximize sensitivity to new models.

A complete analysis selection was implemented for the 2022 CMS Open Data Workshop for a CMS Run 1 vector-like quark analysis with boosted W and Higgs bosons, CMS-B2G-16-024.

- Runs the full set of data & MC events used by the analysis.
- Runs on a docker container hosting CutLang, ROOT, xrootd access to open data, and VNC.
- Complete tutorial link.

Data, BG and 2 signal models vs. $S_T$ for the H1b signal region. BG prediction from MC.

Junghyun Lee, Sezen Sekmen (Kyungpook Nat. U.)
Infrastructure developments

Formal AST production infrastructure developments ongoing in two areas:

1. Decoupling the grammar implementation from input data attributes, external functions, ... :
   • Particle and function names would no longer be hardcoded in the ADL parser.
   • After initial parsing, match function and particle names to those within an external library.

=> Portability of different data types, attributes, functions.

2. Generalization to multiple grammars for multiple domains (e.g. astroparticle experiments):
   • Building a new protocol called Dynamic Domain Specific eXtensible Language (DDSXL).
To conclude:

• ADL and CutLang present a multipurpose and practical analysis approach.

• Numerous analysis implementation studies confirm the feasibility of this approach.

• SModelS EM-creator is adapted for use with ADL/CutLang for analysis validation.

• A large-scale analysis implementation and validation effort is launched for reinterpretation studies.

• ADL/CutLang can process various ATLAS & CMS Open Data.

• ADL syntax refinements and formal compiler/interpreter infrastructure developments are ongoing.

ADL / CL intended as a community effort!
Everyone is welcome to join the development of the language and tools.
Extra slides (syntax)
### 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**
---|---
declare 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](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: \texttt{abs()}, \texttt{sqrt()}, \texttt{sin()}, \texttt{cos()}, \texttt{tan()}, \texttt{log()}, ...

• Collection reducers: \texttt{size()}, \texttt{sum()}, \texttt{min()}, \texttt{max()}, \texttt{any()}, \texttt{all()}, ...

• HEP-specific functions: \texttt{dR()}, \texttt{dphi()}, \texttt{deta()}, \texttt{m()}, ....

• Object and collection handling: \texttt{union()}, \texttt{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: \texttt{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