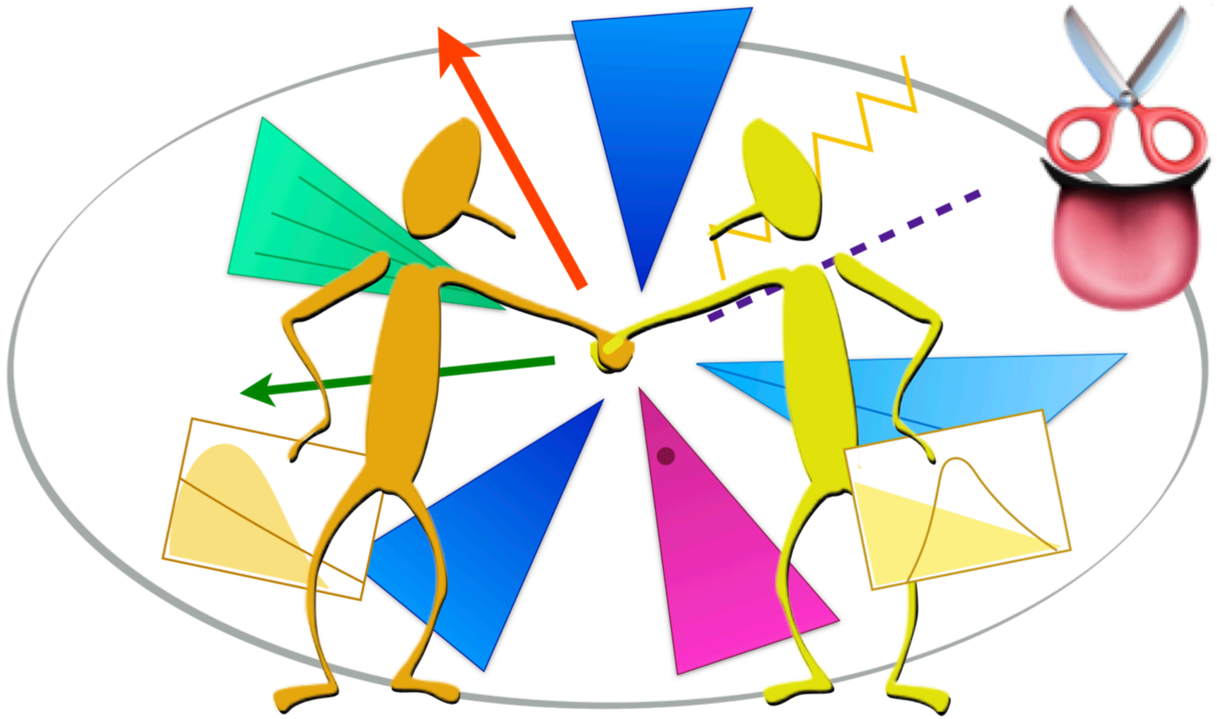


ADL/CutLang developments for open science

Ahmetcan Sansar (Istanbul U.)

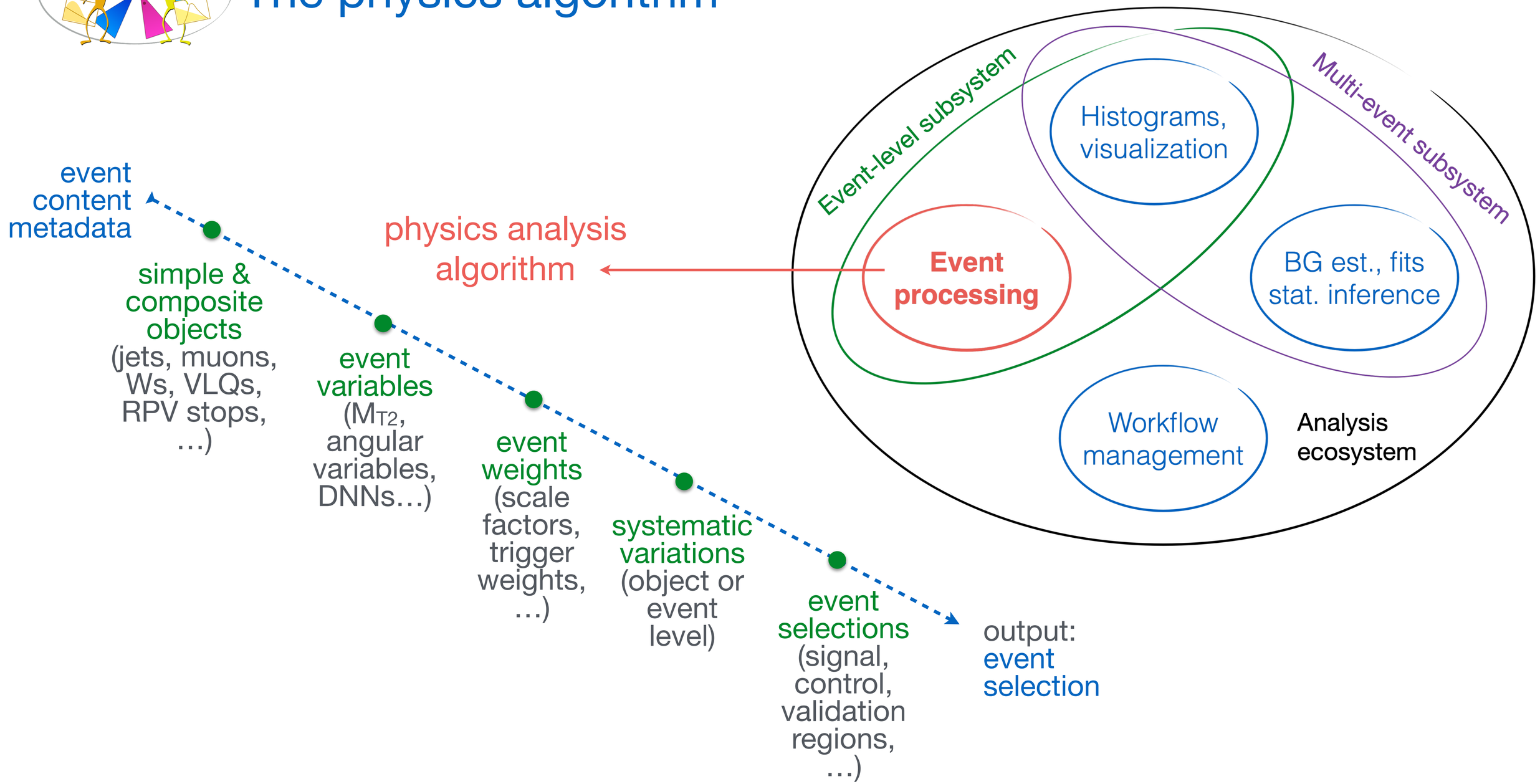
for the *ADL/CutLang* team

(Re)interpretation of the LHC results for new physics

25-28 February 2025, CERN

OpenMAPP

The physics algorithm



Physics analysis algorithms: why formalize them?

Many benefits:

- Clear descriptions of event processing independent of the underlying software.
- Communicating analysis details (within teams, with reviewers, with the community).
- Understanding multiple analyses easily.
- Comparing and contrasting multiple analyses easily.
- Performing (re)interpretation studies systematically.
- Performing creative studies using multiple analyses simultaneously.
- Preserving analyses so that decades later they can be replicated if desired.

—> **algorithms must be available in a clear, precise, and accessible manner.**

Analysis Description Language

cern.ch/adl

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

- External domain specific language: Custom syntax to express analysis-specific concepts. Reflects conceptual reasoning of particle physicists. Focus on physics, not on programming.
- Declarative: Tells what to do but not how to do it.
- Easy to read: Clear, self-describing syntax; organized structure.
- Generic construct designed for everyone: experimentalists, phenomenologists, students, public…

ADL is framework-independent. Decouples physics information from software / framework details.
—> Any framework recognizing ADL can perform tasks with it.

- Multi-purpose use: Auto-translatable into any language / framework most suitable for a purpose, e.g. experimental analysis, pheno / (re)interpretation, queries via static analysis, …
- Easy communication between groups: exp, pheno, referees, students, public.
- Automatic preservation of analysis physics content.

Analysis Description Language

cern.ch/adl

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

- External domain specific language: Custom syntax to express analysis-specific concepts. Reflects conceptual reasoning of particle physicists. Focus on physics, not on programming.
- Declarative: Tells what to do but not how to do it.
- Easy to read: Clear, self-describing syntax; organized structure.
- Generic construct designed for everyone: experimentalists, phenomenologists, students, public…

ADL is framework-independent. Decouples physics information from software / framework details.
—> Any framework recognizing ADL can perform tasks with it.

- Multi-purpose use: Auto-translatable into any language / framework most suitable for a purpose, e.g. experimental analysis, pheno / (re)interpretation, queries via static analysis, …
- Easy communication between groups: exp, pheno, referees, students, public.
- Automatic preservation of analysis physics content.

The ADL construct

ADL consists of

- a plain text ADL file describing the analysis algorithms 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 file consists of blocks separating object, variable and event selection definitions. Blocks have a keyword-instruction structure.

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

- keywords specify analysis concepts and operations.

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

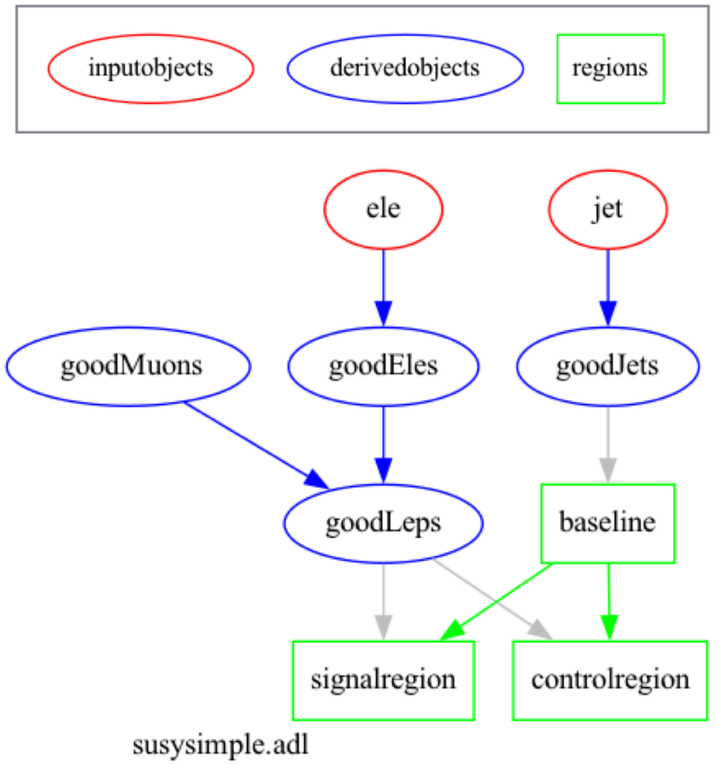
ADL syntax rules with usage examples: link

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

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
```



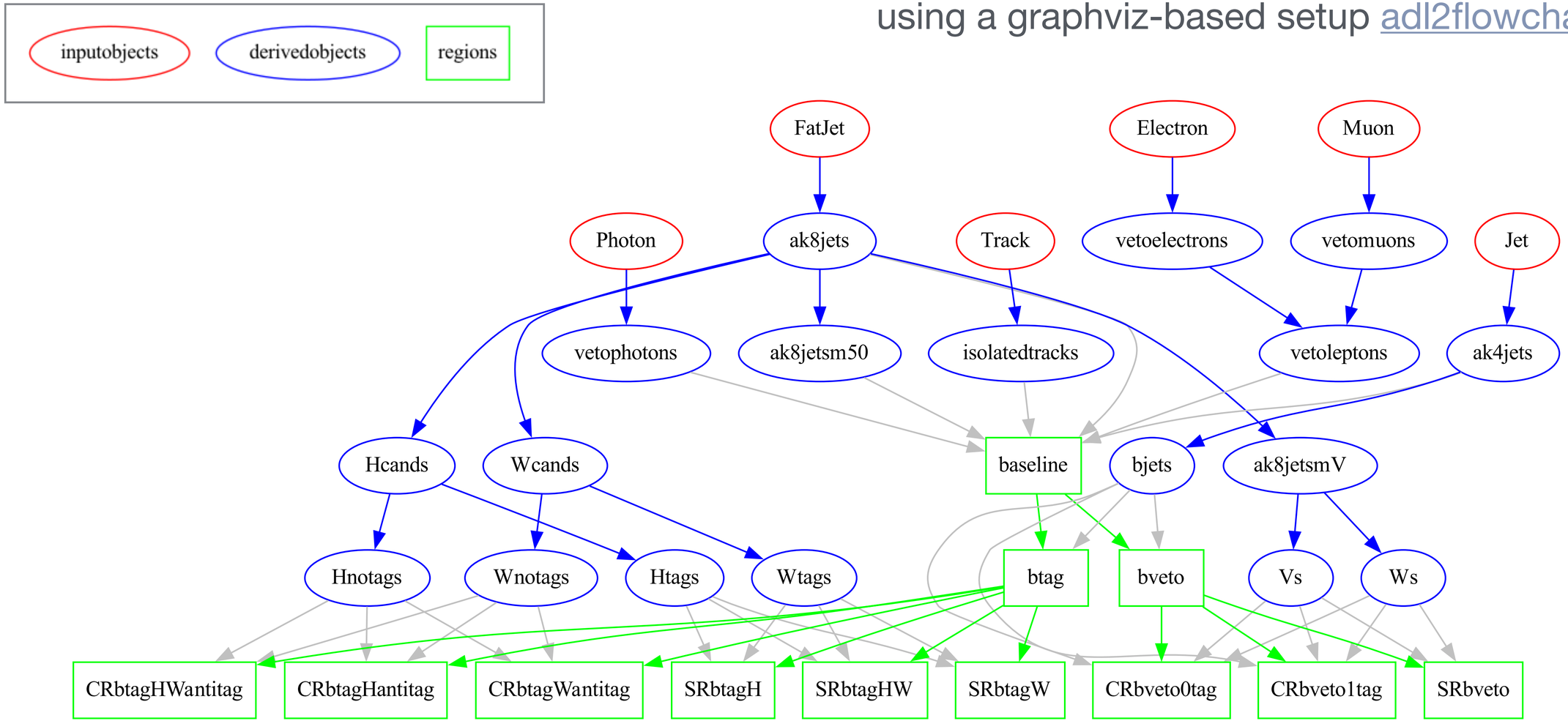
susysimple.adl

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

A not quite so simple example…

arXiv:2205.09597 (CMS-SUS-21-002): Search for EWK SUSY in WW, WZ and WH hadronic final states

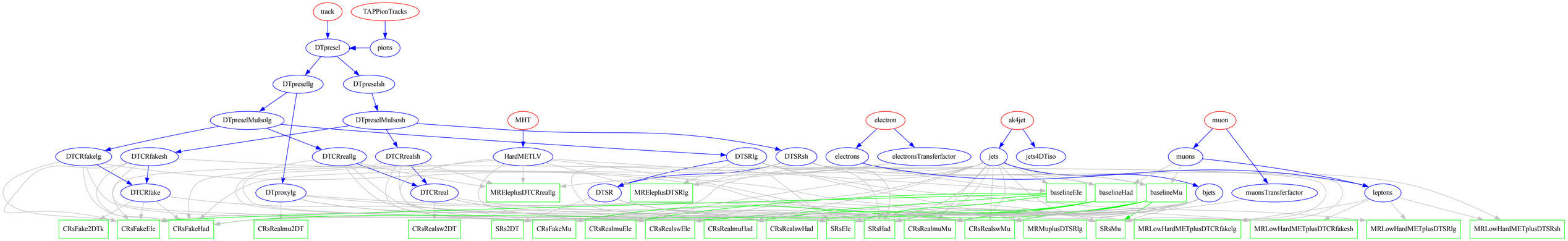
Flowchart automatically generated from the ADL file using a graphviz-based setup adl2flowchart.

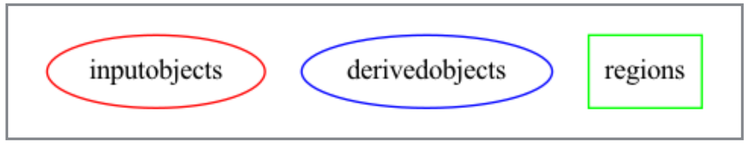


…and another much less simple example

arXiv:2309.16823 (CMS-SUS-21-006): SUSY disappearing track analysis

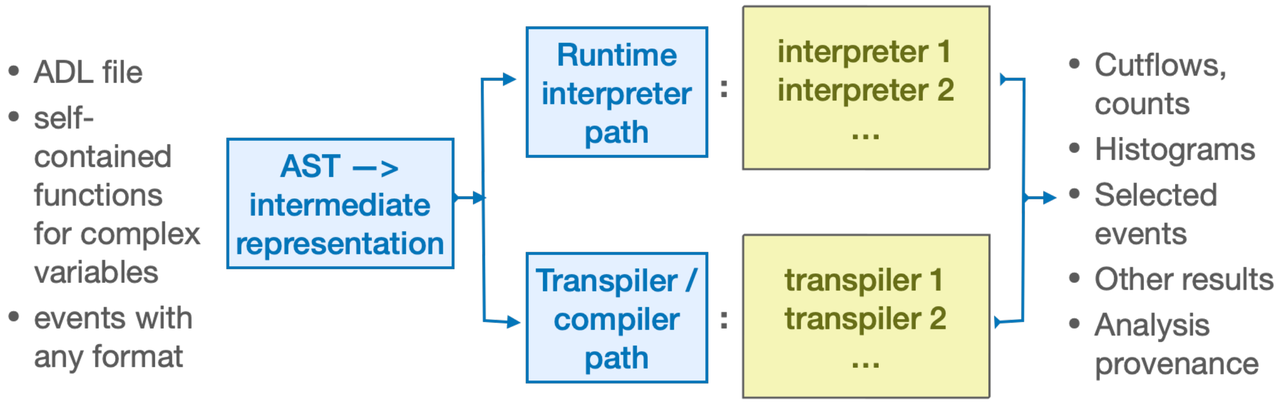
analysis ADL file





Running analyses with ADL: Runtime interpreter

Experimental / phenomenology analysis model with ADL



CutLang: C++ runtime interpreter for ADL.

Formal grammar parsing by Lex & Yacc.

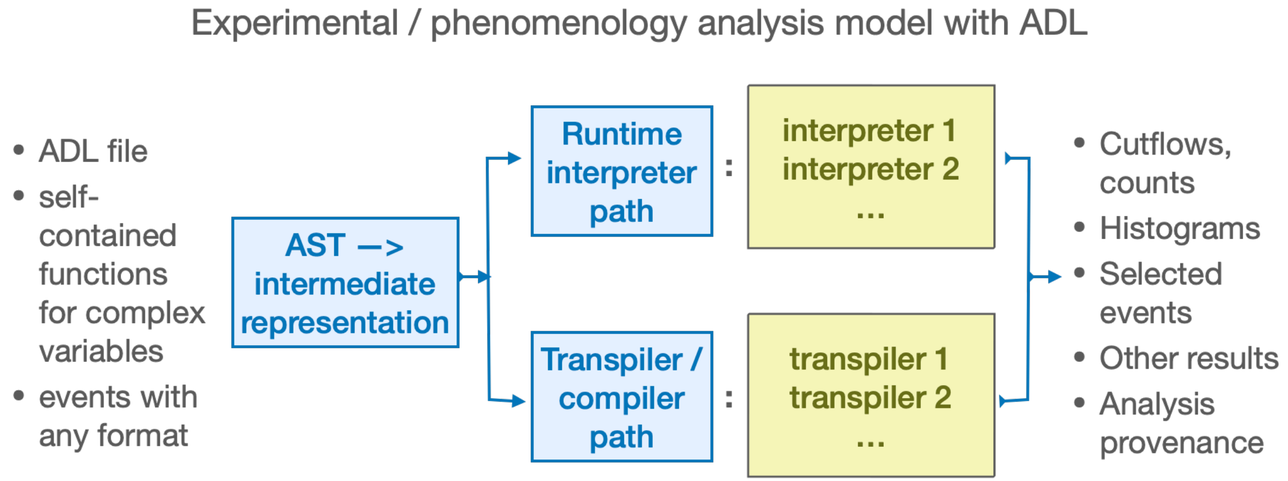
- Based on ROOT. Reads TTree-like formats. NanoAOD, Delphes, open data, etc. Semi-automated integration of new formats.
- Many external functions, including ML model interface via ONNX.
- Runs in linux, macOS. Available in Docker, Conda. Jupyter kernel exists (binder or conda).
- Outputs cutflows, histograms, events, analysis description, i.e. provenance.

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

Gökhan Ünel, Anna-Monica Toon, Burak Şen, et. al.

Running analyses with ADL: Transpilers

NEW!



Physics information fully contained in ADL.

Current compiler infrastructures can be easily replaced by future tools / languages / frameworks.

- ADL transpiler for CMS framework TIMBER (a layer on top of ROOT RDataFrame, used in CMS with NanoAOD data) ~80% done.
- ADL transpiler for commonly used columnar analysis tool Coffea : ~40% done.
- Both use new Analysis-Level Instruction Language (ALIL) as intermediate layer.
  - ADL -> ALIL -> TIMBER, Coffea, or... <insert your own framework here>
- Use for real data analyses and research level Open Data analyses.

Petar Maksimovic, Dennis Plotnikov, Lindsey Grey

ADL/CutLang for reinterpretation

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

- Clear description of the analysis logic.
- Straightforward adaptation from experiments to public formats.
  - Repurpose ADL files: swap experimental object definitions with simplified object blocks based on numerical object ID / tagging efficiencies.
  - Event selections stay almost the same.
  - Efficiencies can be implemented via hit-and-miss function.
- 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.

syntax examples in backup

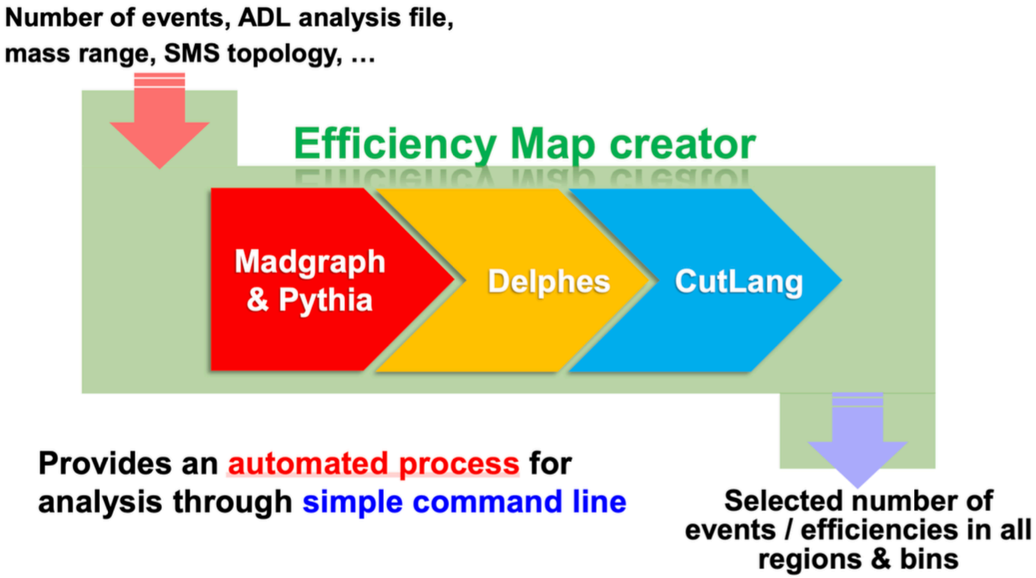
Efficiency Map Creator (EM-creator)

We launched a large scale analysis validation effort with ADL/CutLang.

- Main focus currently SUSY.

Using SModelS Efficiency Map Creator for validation:

- Validate analyses by comparing to experimental results
- Configurable user interface: define models and mass points to produce, steps to run, output to save.
- Infastructure set up in Vienna and KNU T3, also adapted for LXPLUS use.
- Easy to run in parallel with HTCondor.



- Limit calculation currently within SModelS
  - Work on decoupling is starting.

Wolfgang Waltenberger, Jan Mrozek, Gökhan Ünel, Junghyun Lee, Changgi Huh

ADL/CL validation effort team

Seniors: Aytul Adiguzel (Istanbul U.), Sezen Sekmen (KNU)

Post-Doc: Ekin Sila Yoruk (Bogazici U.)

PhD Students: Ahmetcan Sansar (Istanbul U.), Junghyun Lee (KNU), Feyza Baspehlivan (TOBB. ETU.)

Master Student: Kagan Sahan (Istanbul U.)

Undergraduate Student: Demircan Demirbag (Bogazici U.)
(more students will join soon)

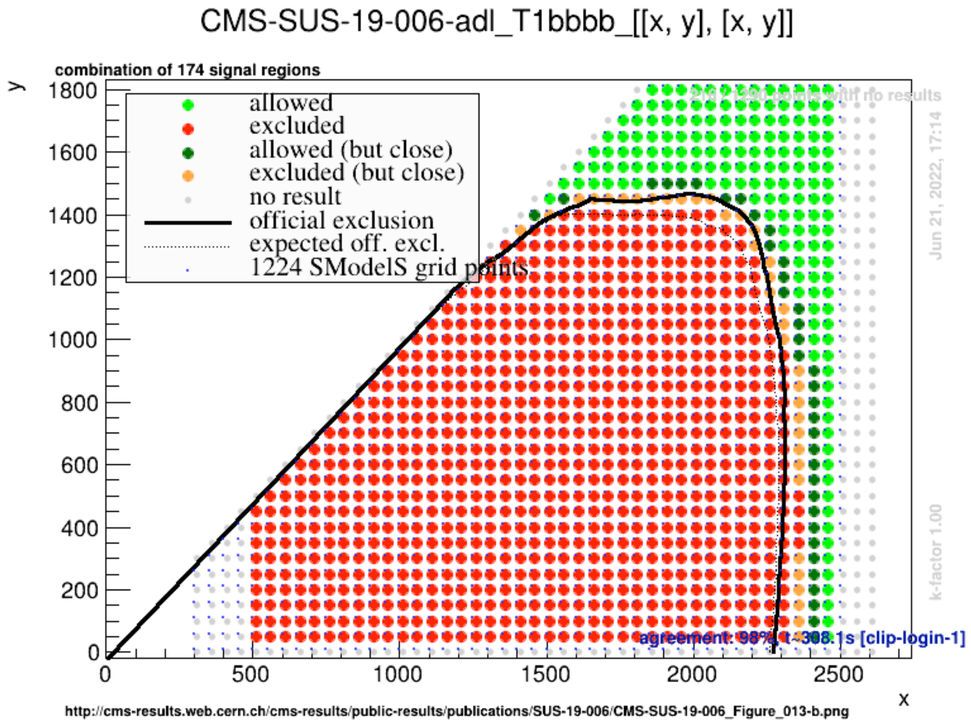
We have ~weekly meetings.

Generally doing well, but of course we are facing challenges:

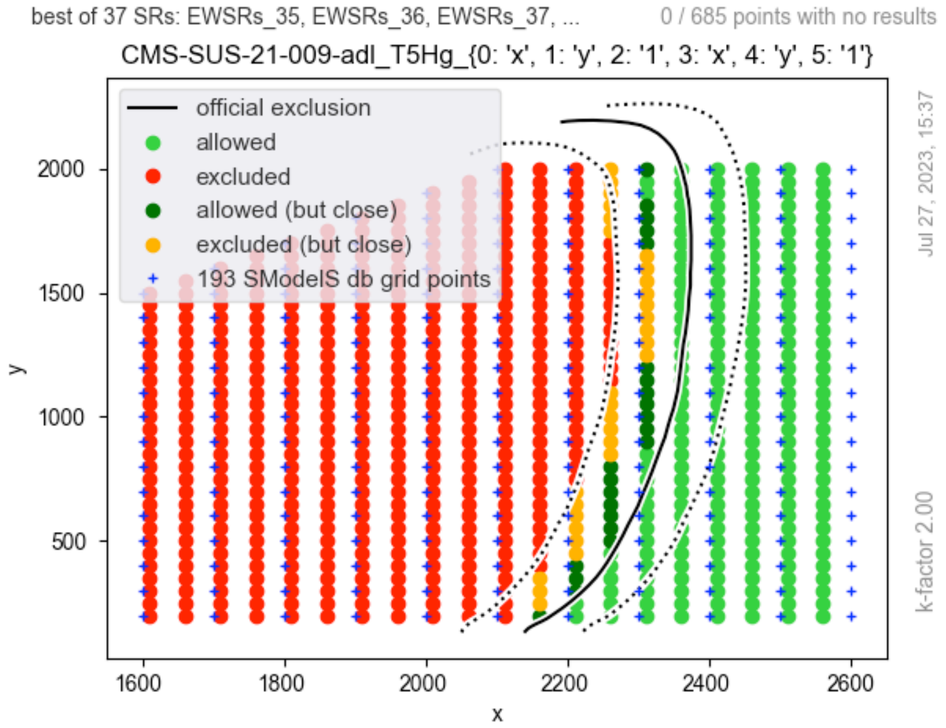
- ADL implementations are mostly straightforward.
- Lack of clear documentation on signal sample production details in papers / analysis notes.
- Incomplete records with missing information, e.g. object definitions / efficiencies, event selections and cutflows.

A couple validated examples

CMS-SUS-19-006: Inclusive SUSY with jets + b jets and MET — ADL files



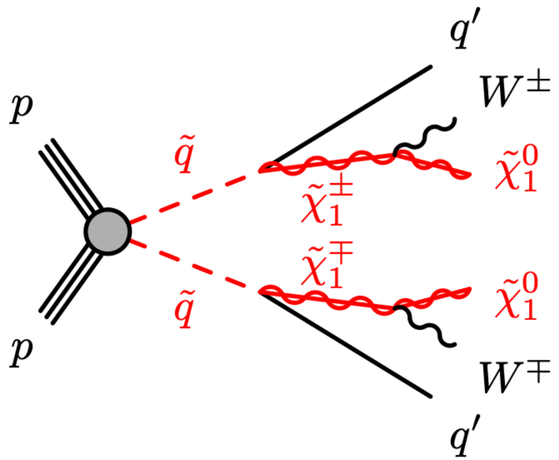
CMS-SUS-21-009: photons + multijets + MET
ADL files

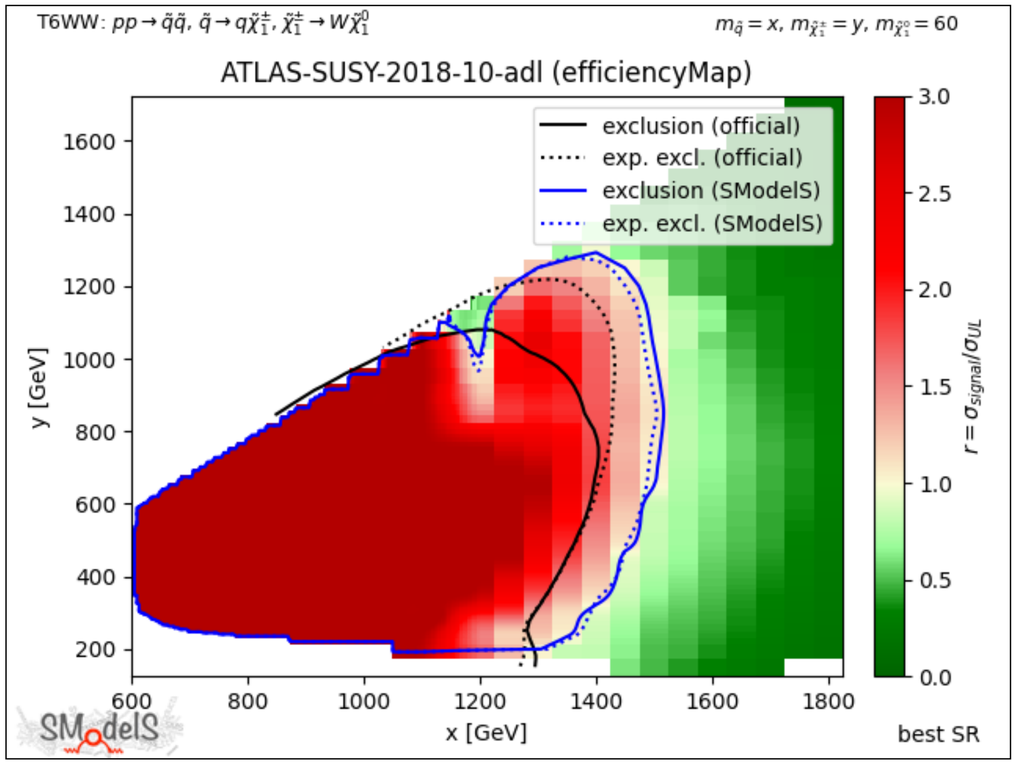


Examples from our recent validation studies - I

ATLAS-SUSY-2018-10: Search for squarks and gluinos in final states with one isolated lepton, jets + MET

- Two different models: SS_onestep and GG_onestep.
- Produced exclusion plot for Squark model with $\tilde{\chi}_1^0$ = 60 GeV mass spectra.
- Work on improvement is ongoing.





Examples from our recent validation studies - II

ATLAS-SUSY-2018-30: Search for supersymmetry in final states with missing transverse momentum and three or more b-jets

- Utilizing CutLang's ONNX Model Executor.
- Cutflow comparison shows consistent results for both models.
- Next step is mass grid production and calculating limits.

```
# define the ML output
define NN1Cut = OME("ANA-SUSY-2018-30_model.onnx" ,
NNSRGc21001Var, NNmeans, NNsigmas, 0 )
# Use in the selection
region SRGc21001
(…. other selection criteria)
select NN1Cut >= 0.9997
```

Gtt Model (SR-Gtt-2300-1200)

Selection	ATLAS	ADL/CL
	N_{events}	N_{events}
Derivation Skim	11.39	11.39
$(N_{lepton,signal} \geq 1) \| \| (N_{lepton,base} = 0) \&\& \Delta\phi_{min}^{4j} \geq 0.4$	7.66	9.57
P(Gtt) > 0.9993	2.95	3.61

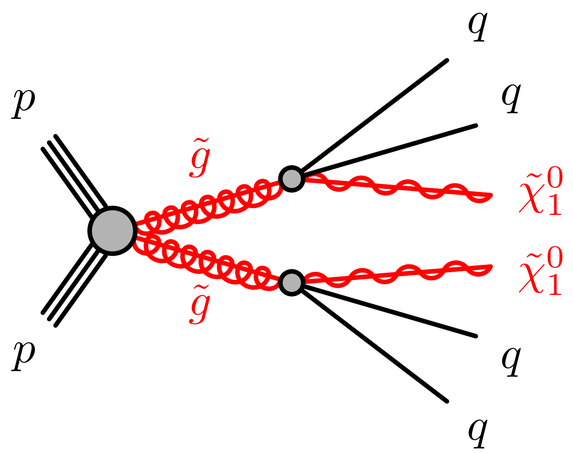
Gbb Model (SR-Gbb-2100-1600)

Selection	ATLAS	ADL/CL
	N_{events}	N_{events}
Derivation Skim	80.90	80.90
$N_{leptons,base} = 0$	80	80.84
$\Delta\phi_{min}^{4j} \geq 0.4$	61.61	62.99
P(Gbb) > 0.9993	6.20	7.77

Examples from our recent validation studies - III

ATLAS-SUSY-2018-22: Search for squarks and gluinos in final states with jets and missing transverse momentum

- Cutflow comparison done for "Model-independent search" regions (GG_direct model).
  - Aplanarity and Sphericity functions have been implemented in CutLang.
- Mass grid production is ongoing.



$m_{\tilde{g}}$= 1400 GeV, $m(\tilde{\chi}_1^0)$ = 1000 GeV	Selection	ATLAS		ADL/CL	
		N_{events}	Eff.	N_{events}	Eff.
Common Requirements	Pre-selection, $E_T^{miss} > 300$ GeV, $p_T(jet_1) > 100$ GeV, $m_{eff} > 800$ GeV, jet multiplicity≥ 2	1787.00	1.00	1787.00	1.00
SR4j-3400	Jet multiplicity >= 4	1355.00	0.76	1383.10	0.77
	$\Delta\Phi(jet_{1,2,(3)}, E_T^{miss})_{min} > 0.4$	1199.00	0.88	1295.09	0.94
	$\Delta\Phi(jet_{i>3}, E_T^{miss})_{min} > 0.2$	1099.00	0.92	1110.52	0.86
	$p_T(jet_4) > 100$ GeV	498.00	0.45	492.76	0.44
	$\lvert\eta(jet_{1,2,3,4})\rvert < 2.0$	435.00	0.87	436.43	0.89
	Aplanarity > 0.04	318.00	0.73	303.06	0.69
	$E_T^{miss}/\sqrt{H_T} > 10$ GeV$^{1/2}$	294.00	0.92	274.80	0.91
	$m_{eff} > 3400$ GeV	1.15	0.00	0.68	0.00

Analyses we are recently validating

CMS-SUS-18-004: Search for supersymmetry in final states with two or three soft leptons and missing transverse momentum.

CMS-SUS-20-003: Search for chargino-neutralino production in events with Higgs and W bosons.

ATLAS-SUSY-2019-22: Search for direct production of winos and higgsinos in events with two same-charge leptons or three leptons.

ATLAS-SUSY-2019-09: Search for chargino-neutralino pair production in final states with three leptons and missing transverse momentum.

Other analyses we are validating: have their ADL descriptions almost ready, but validation is nontrivial due to incomplete information.

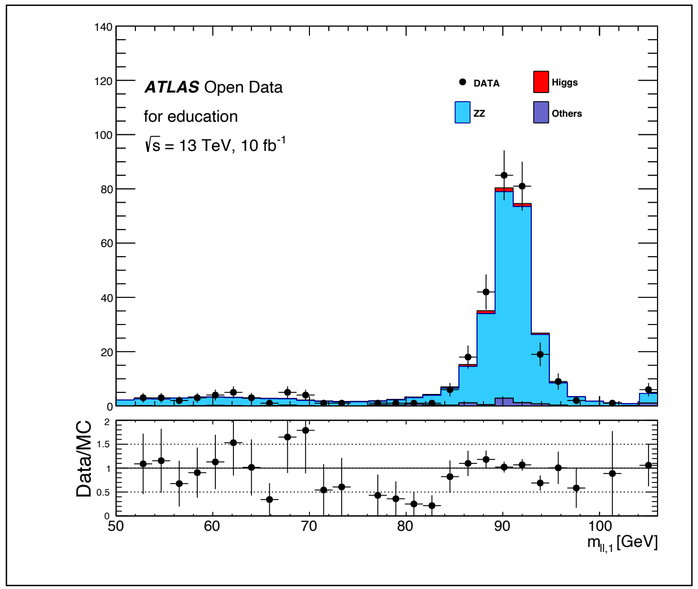
ADL/CL and ATLAS Open Data

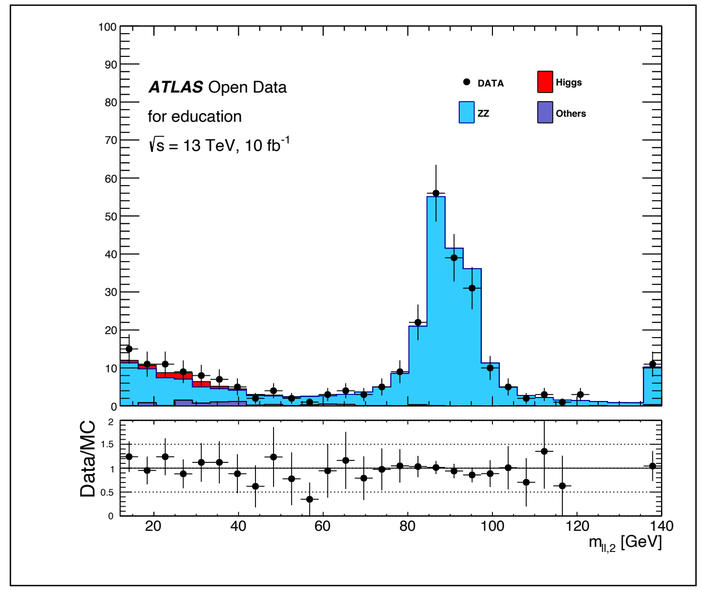
ADL/CL can be used to run analyses with ATLAS (educational) open data.

- Goal of the study:
  - Re-optimize and re-run the analyses provided by ATLAS-outreach-13TeV-framework.
  - Also a stress test for the new CutLang feature: Reading variables directly from NTuples.
  - ADL interpretations of ATLAS open data analyses can be accessed on GitHub.

```
object goodEle
  take ELE
  select Pt(ELE) > 7
  select absEta(ELE) < 2.47
  select lep_ptcone30(ELE)/lep_pt(ELE) < 0.3
  select lep_etcone20(ELE)/lep_pt(ELE) < 0.3

define SFactor: scaleFactor_ELE*scaleFactor_MUON*....

define totweight: XSection*mcWeight*SFactor*Lumi/SumWeights
```

CutLang is now able to read and process branches directly, without explicitly defining them.

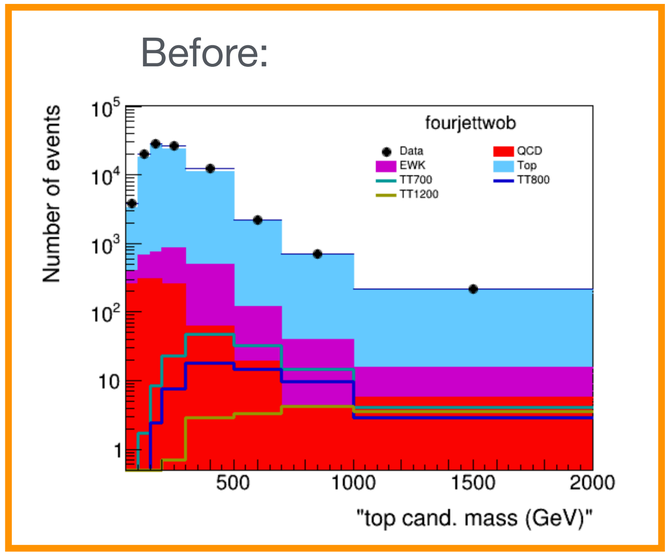




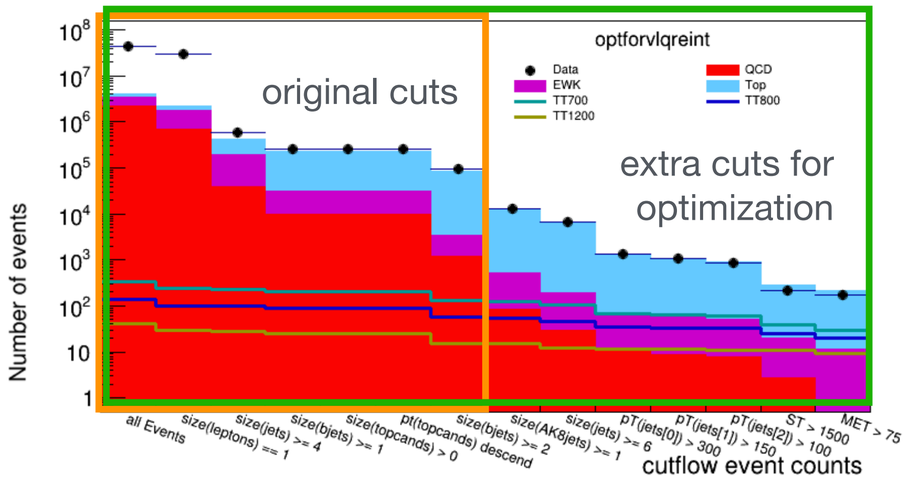
Detailed report in final state of preperation

ADL/CL + CMS Open Data + Reinterpretation

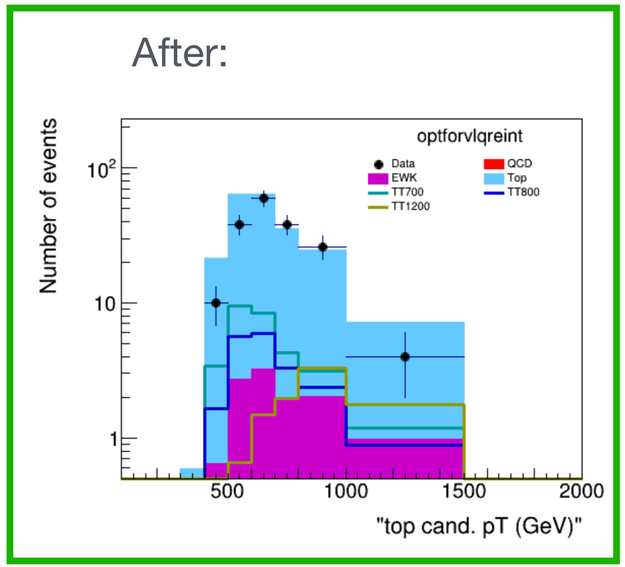
- Can be used to re-optimize and re-run recasted analyses to maximize sensitivity to new models.
- Studied exact and “optimized” reinterpretation of a ttbar analysis for a vector-like T quark signal.
  - Focus on reoptimizing the analysis to enhance sensitivity to VLT (Complete tutorial link.)
  - Runs on a full set of open data & MC events in a docker container hosting CutLang, ROOT, xrootd access to open data, and VNC.



Data, BG and 2 VLT signals vs. top candidate mass for the ttbar analysis selection.



Cutflow histograms auto-generated by CutLang



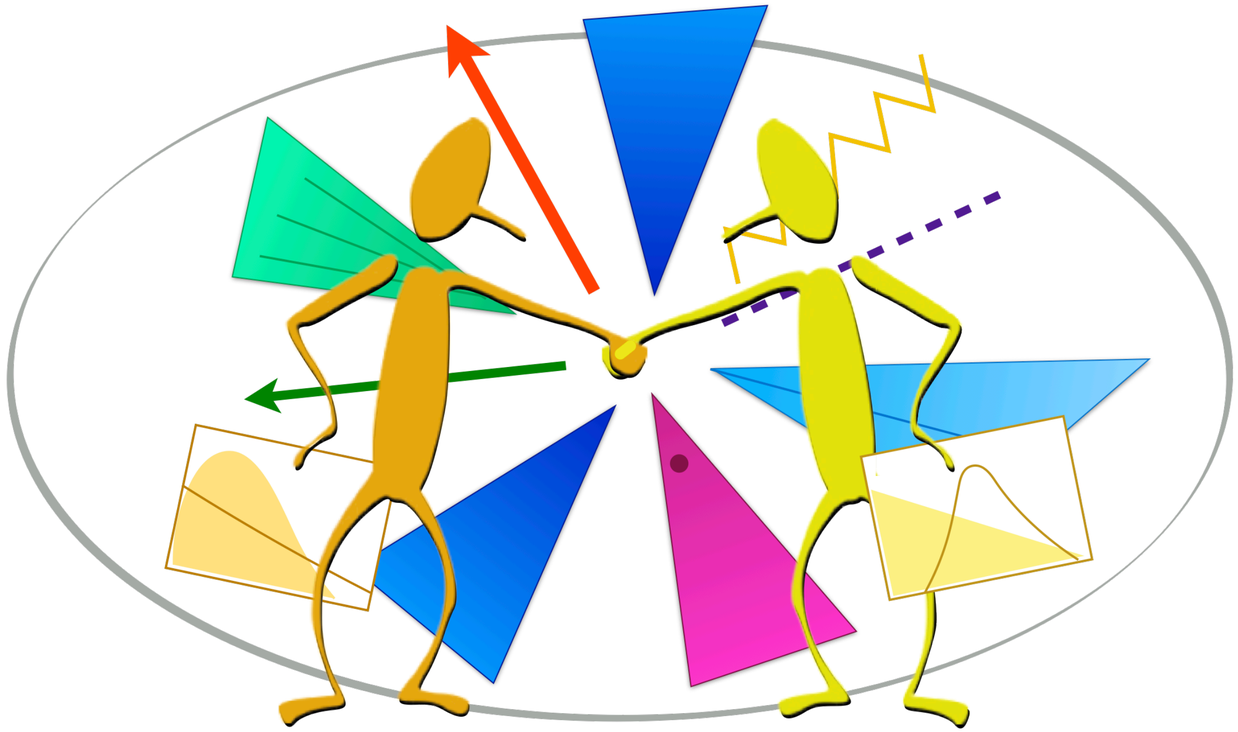
Top candidate pT after reoptimizing the ttbar selection by adding several cuts.

Conclusion

- ADL ecosystem presents a multipurpose and practical analysis approach.
  - Versatile implications for reinterpretation and open science.
- Analysis reimplementation / validation effort ongoing with ADL/CutLang via SModelS EM-creator.
  - The work on decoupling the limit calculation from SModelS is starting.
- ADL syntax and tools are under constant development.
- ADL/CutLang related developments will continue within the OpenMAPP consortium:
  - HEPData interface, generalized, robust interface to MC tools, statistical tools, etc.
- We invite all to explore ADL and provide feedback (mattermost channel).

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

Documentation and references : cern.ch/adl



BACKUP

ADL for *analysis preservation*

ADL's initial motive was to ensure long-term analysis logic preservation:

- Decoupled from analysis frameworks, portable, self-documenting, modular

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

1. Intend to build 3 web-based, searchable and citable CAP databases for ADL content:

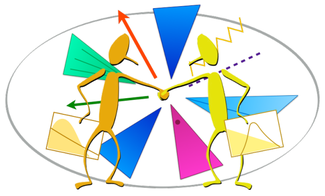
- ADL analyses database: Host ADL files of implemented analyses
- ADL objects database: Host object definitions written in ADL (e.g. 2016 tight isolated electrons for CMS leptonic SUSY analyses, boosted medium Higgs from ATLAS, etc.)
- ADL functions database: Host external functions of non-trivial or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

2. Intend to use ADL as a backend to fill CAP web forms with analysis logic information.

- Investigation started based on the ADL parser and AST infrastructure.

Versatile uses of ADL - I (beyond running on events)

- Analysis design (experimental or pheno):
  - Quick prototyping.
  - Easy comparison with existing analyses: “Was my phase space already covered?”
  - Simultaneous test of numerous selection options in a self-documenting way.
- Objects handling:
  - Easy reuse in new analyses.
  - Compare object definitions within or between analyses.
  - Bookkeeping for SF calculations.
  - Compare definitions in different input data types.
- Communication:
  - Between analysis team members (easy synchronization); with reviewers; between teams; between experiments or exp. and pheno.
- Analysis visualization:
  - Build analysis flow graphs and tables from analyses using static program analysis tools.



Versatile uses of ADL - II (beyond running on events)

- 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).
- …

… and how would YOU use it?

A little history - I

1. LHADA: Les Houches Analysis Description Accord (2015-2019)

- Frustrating difficulty in reproducing existing analyses due to insufficient information in publications. Also hard to trace physics details from analysis codes.
  - We experienced this first hand in the CMS Run1 pMSSM interpretation study.
  - Request for clear analysis descriptions from the pheno community for use in reinterpretation studies: arXiv:1203.2489.
- Les Houches PhysTeV workshop 2015: Discussion among a group of ~20 experimentalists and phenomenologists to create an accord for describing analysis physics content.
  - —> Initial design of LHADA (LH15 proceedings, arXiv:1605.02684.
    - inspired by earlier successful LH accords, e.g. SUSY LH Accord and LHE.
- 2 LHADA workshops: Grenoble, 25-26 Feb 2016; CERN, 16-18 Nov 2016 (link).
- Initial focus of LHADA: Provide a standard physics content description decoupled from frameworks; unobscured documentation of analyses; preservation.
- Providing infrastructures to make the description executable started later.

A little history - II

2. CutLang: A DSL (2015-2019) + runtime interpreter (2015-…)
arXiv:1801.05727 , arXiv:1909.10621 , arXiv:2101.09131 .

- Initial purpose: Provide an easy analysis infrastructure for beginner students that can bypass coding mistakes.
- Main focus: Explore building a runtime interpretable DSL.
- Developed adapting a bottom-up, practical approach.

2019: LHADA DSL + CutLang DSL —> ADL

- Analysis Description Languages for the LHC workshop at Fermilab LPC, 6-8 May 2019 (link): Brought experimentalists, phenomenologists and computer experts together for a dedicated discussion on DSLs.
- Started to build a generic and multipurpose language and compiler infrastructures towards realistic use for particle physics analysis tasks.

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 continuously evolves 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

ADL/CL for reinterpretation: Portable objects

Enables straightforward adaptation from experiments to public input event formats.

b-tagging for UL NanoAODv9

```
# b-tagged jets - medium
object MediumBTag
  take Jet
  select btagDeepB(Jet) >= 0.2783
```

b-tagging for public use, e.g. with Delphes

```
# b-tagged jets - medium
object MediumBTag
  take Jet
  select applyHM( btagdeepBmediumeff( pt(Jet), abs(eta(Jet)) )  == 1)
```

A generic function reading efficiencies, object attributes and applying the hit & miss method.

Efficiencies provided by CMS. ADL table blocks can host numerical efficiencies.

```
table btagdeepCSVmedium
  tabletype efficiency
  nvars 1
  errors true
  # eff      err-      err+      pTmin     pTmax
  0.5790     0.0016    0.0016    -10.4      30.0
  0.6314     0.0013    0.0013    30.0       35.0
  0.6442     0.0011    0.0011    35.0       40.0
  ...
```

- 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

ADL/CL for reinterpretation: 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)
```

```
table btagdeepCSVmedium
  tabletype efficiency
  nvars 1
  errors true
  # val    err-     err+       pTmin    pTmax
  0.5790   0.0016   0.0016    -10.4     30.0
  0.6314   0.0013   0.0013     30.0     35.0
  0.6442   0.0011   0.0011     35.0     40.0
  0.6596   0.0007   0.0007     40.0     50.0
  0.6727   0.0007   0.0007     50.0     60.0
  0.6812   0.0008   0.0008     60.0     70.0
  0.6855   0.0008   0.0008     70.0     80.0
  0.6873   0.0009   0.0009     80.0     90.0
  0.6881   0.0010   0.0010     90.0    100.0
  0.6880   0.0008   0.0008    100.0    125.0
  0.6867   0.0011   0.0011    125.0    150.0
  0.6826   0.0015   0.0015    150.0    175.0
  0.6734   0.0020   0.0020    175.0    200.0
  0.6624   0.0026   0.0026    200.0    225.0
  0.6494   0.0034   0.0034    225.0    250.0
  0.6419   0.0044   0.0044    250.0    275.0
  0.6301   0.0054   0.0054    275.0    300.0
  0.6202   0.0051   0.0051    300.0    350.0
```

ADL/CL for reinterpretation: Counts and cutflows

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

```
countsformat sigone
  process T1tttt1900200, "T1tttt 1900 200", stat
  process T1bbbb1800200, "T1bbbb 1800 200", stat
  process T1qqqq1300100, "T1qqqq 1300 100", stat
  process T5qqqqVV1800100, "T5qqqqVV 1800 100", stat

countsformat sigtwo
  process T1tttt13001000, "T1tttt 1300 1000", stat
  process T1bbbb13001100, "T1bbbb 1300 1100", stat
  process T1qqqq12001000, "T1qqqq 1200 1000", stat
  process T5qqqqVV14001100, "T5qqqqVV 1400 1100", 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 , 100.0 +- 0.1
  counts sigthree 100.0 +- 0.2 , 100.0 +- 0.5 , 100.0 +- 0.5
  counts sigfour 100.0 +- 0.0 , 100.0 +- 0.1 , 100.0 +- 0.2
  select size(jets) >= 2
  counts sigone 100.0 +- 0.8 , 100.0 +- 0.5 , 100.0 +- 0.0 , 100.0 +- 0.5
  counts sigtwo 100.0 +- 0.0 , 99.3 +- 0.1 , 99.6 +- 0.1 , 100.0 +- 0.1
  counts sigthree 99.9 +- 0.2 , 98.8 +- 0.5 , 99.1 +- 0.5
  counts sigfour 99.6 +- 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 90.1 +- 0.4 , 74.8 +- 0.5 , 82.0 +- 0.3 , 94.6 +- 0.4
  counts sigthree 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 sigthree 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
```

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

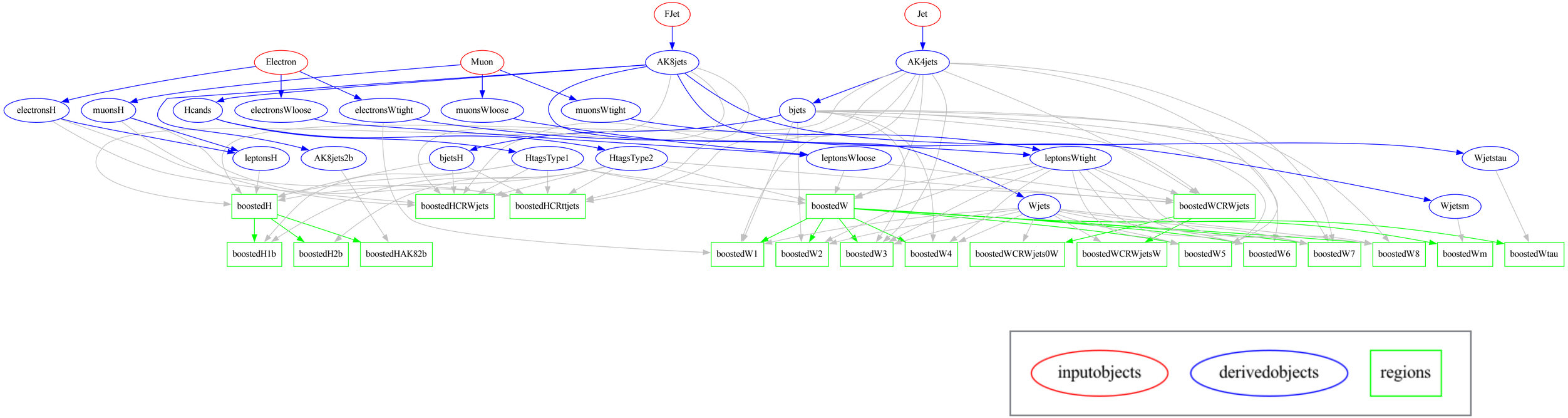
```
countsformat bgests
  process lostlep, "Lost lepton background", stat, syst
  process zinv, "Z --> vv background", stat, syst
  process qcd, "QCD background", stat, syst

countsformat results
  process est, "Total estimated BG", stat, syst
  process obs, "Observed data"
```

```
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 38870 +- 320 +- 580 , 89100 +- 200 +- 2600 , 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
```

…and quite complicated…

arXiv:1706.03408 (CMS-B2G-16-024): Search for vector-like B and T quarks in single lepton final states using jet substructure.

analysis ADL file



Physics analysis algorithms: how accessible are they?

We write and/or document physics analysis algorithms in multiple ways:

- Publications / papers: Only conceptual descriptions. Insufficient accurate technical detail.
- Analysis notes (AN): More detail than in papers, but technical description often incomplete, out of data, and inaccessible outside collaborations.
- Twikis: Often incomplete. Few incentives to keep up-to-date.
- Analysis software frameworks: Analyses written in frameworks are typically the definitive description of these analyses. But:
  - Frameworks are written in general purpose languages like C++ / Python, etc.
  - Physics algorithms and language technicalities are intertwined making it difficult to abstract the algorithm from the code.
  - Code is often confined to an analysis group and everyone's code is different.

An alternative: Describe physics algorithms in a uniform, more accessible, shorthand.

But how exactly?