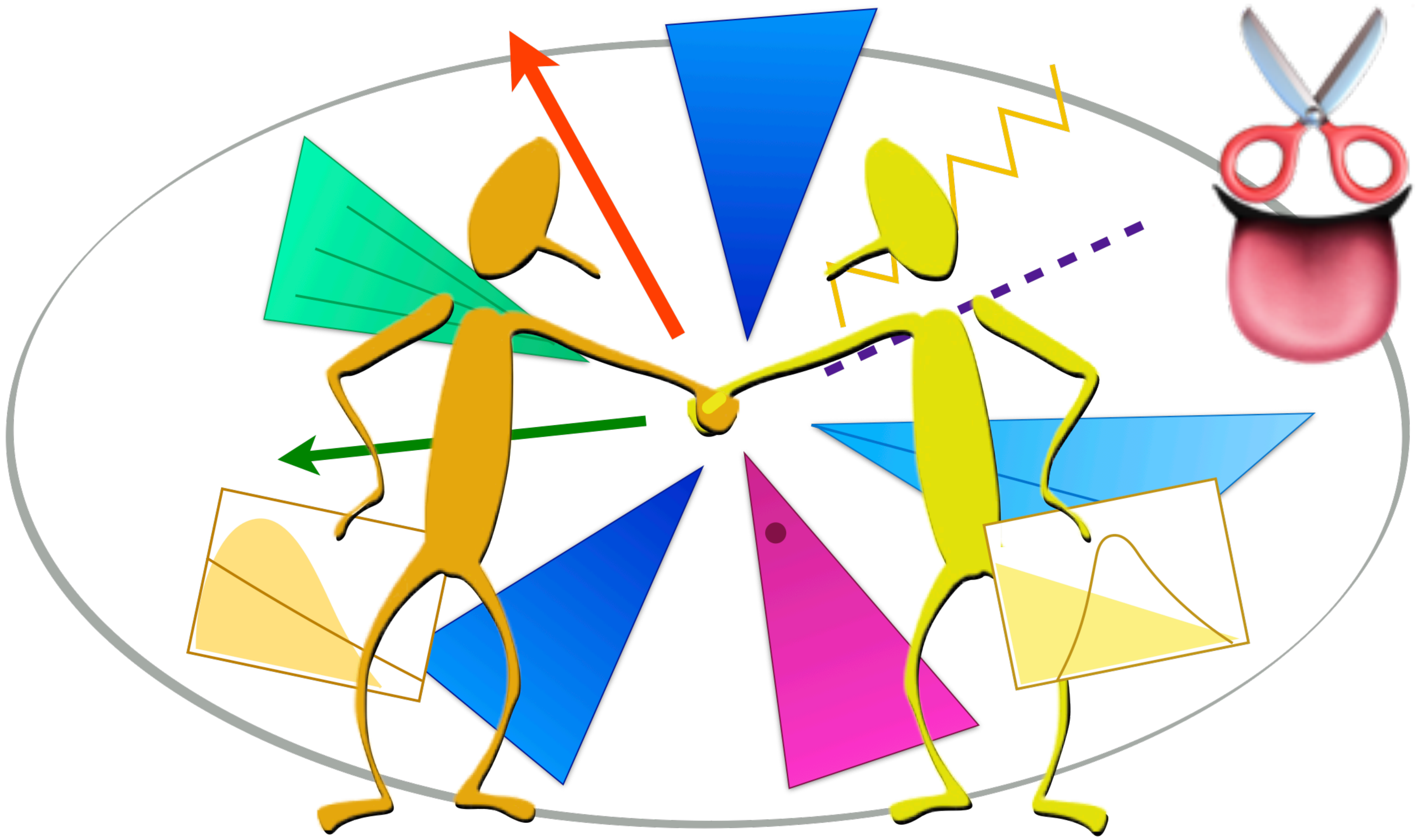
Documentation and references : cern.ch/adl



ADL/CutLang developments towards transparent (re)interpretation

Sezen Sekmen (Kyungpook Nat. U.)

for *the ADL/CutLang team*

Reinterpretation Forum Workshop

29 Aug - 1 Sep 2023, IPPP Durham

Dec 2022

Analysis Description Language for HEP

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

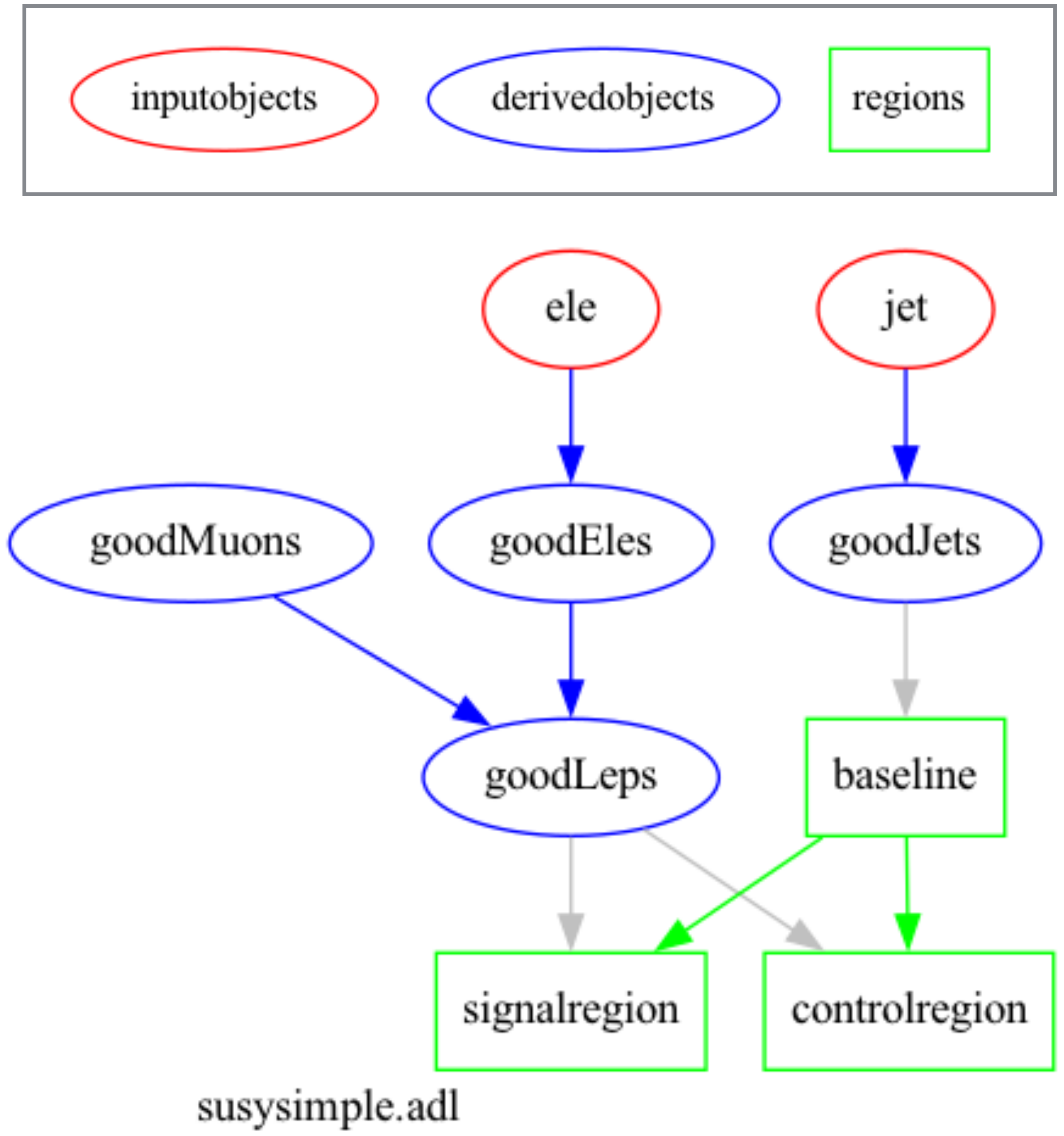
LHADA (Les Houches Analysis Description Accord): Les Houches 2015 new physics WG report (arXiv:1605.02684, sec 17)

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
```



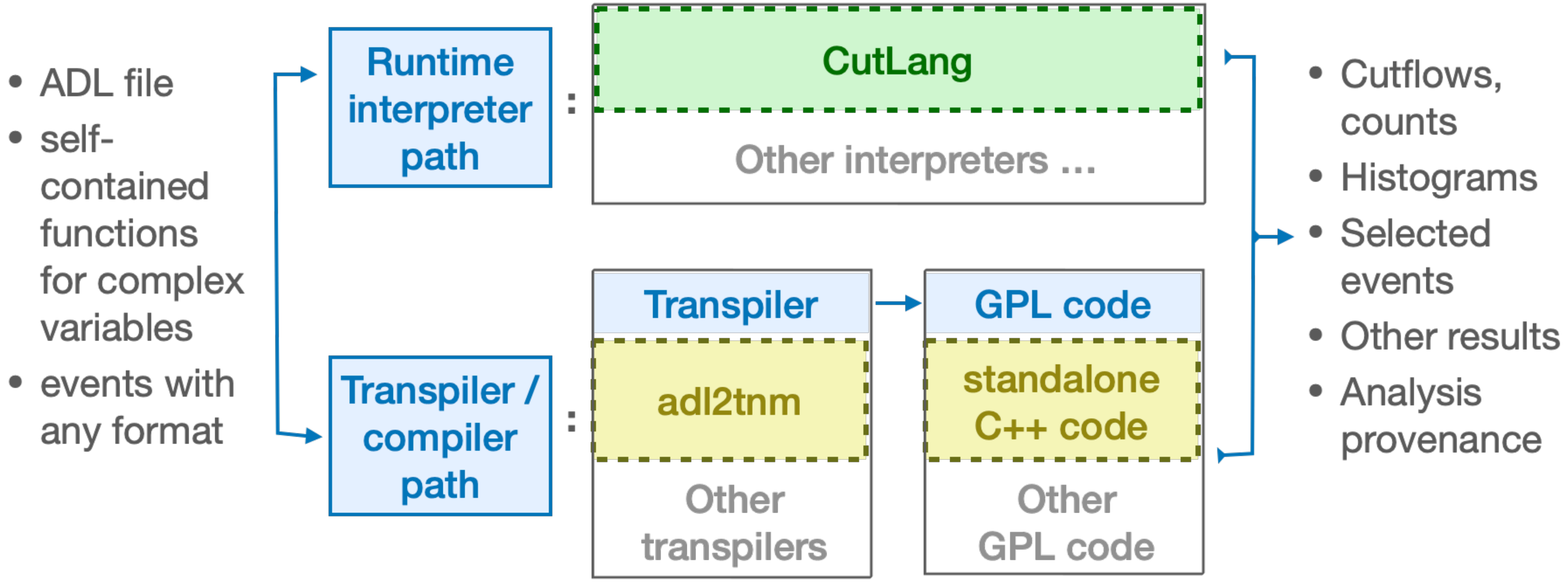
ADL implementations of some LHC analyses: https://github.com/ADL4HEP/ADLLHCanalyses

Running analyses with ADL

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:

Experimental / phenomenology analysis model with ADL



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 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

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)

ADL/CL 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.
  - 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 backup slides).
- Generic syntax available for expressing analysis output in the ADL file: (see backup slides) 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.

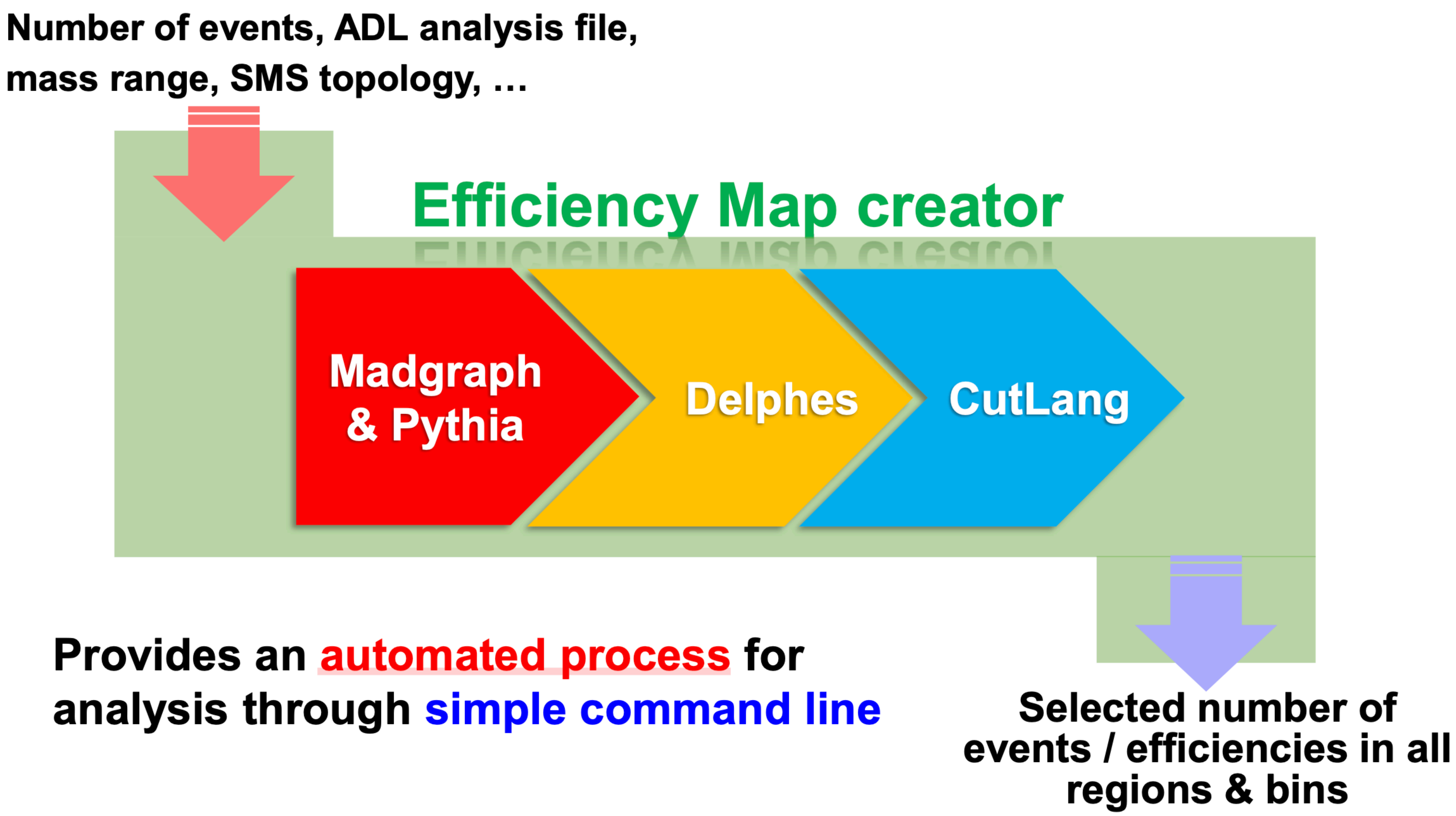
Validation for reinterpretation: Efficiency Map Creator - I

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

- Main focus still SUSY, but also extending to EXO.

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 analyses by comparing to experimental results.
  - Configurable user interface: can specify which models and mass points to produce, which steps to run, which output to save.
- EM-creator was adapted to work with ADL/ CutLang by Wolfgang W. and Jan Mrozek.
  - Final step: Efficiency maps.
  - Limit calculation currently within SModelS.

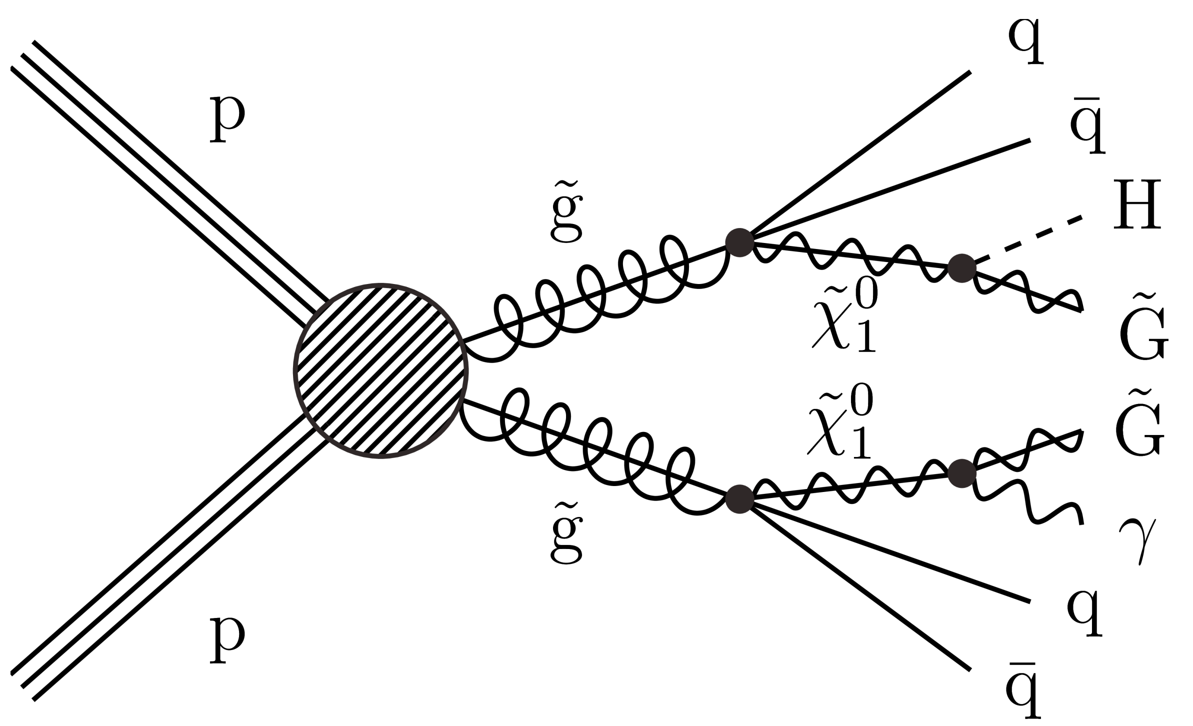


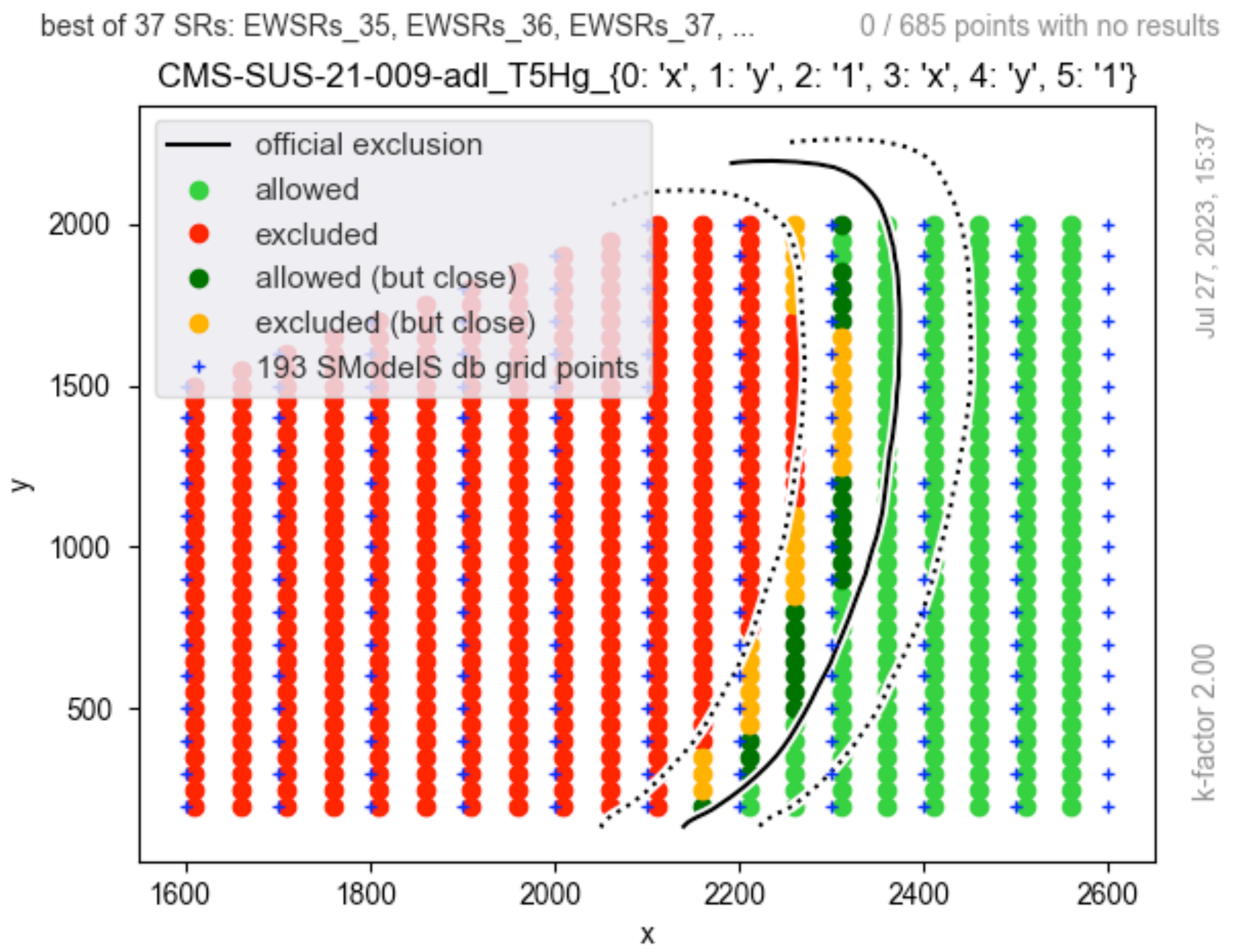
Validation for reinterpretation: Efficiency Map Creator - II

Working to validate several recently published ATLAS and CMS analyses.

- Working with a group of ATLAS and CMS students and partially with CMS analysis teams.

Recent validation example: CMS-SUS-21-009: “photons + multijets + MET”.





Integrating ML models

ADL/CutLang is now adapted to execute ML models.

- Currently work with .onnx models. (https://onnx.ai/) (see https://netron.app/ for an online viewer)
- Integrated in ADL/CL via the downloadable Onnx Model Executer function, with the following syntax:

```
# define the list of inputs
define listofinputs = {var1 var2 var3 …… varN}
# define the ML output
define myMLvar = OME(my/directory/myfunc.onnx, listofinputs)
```

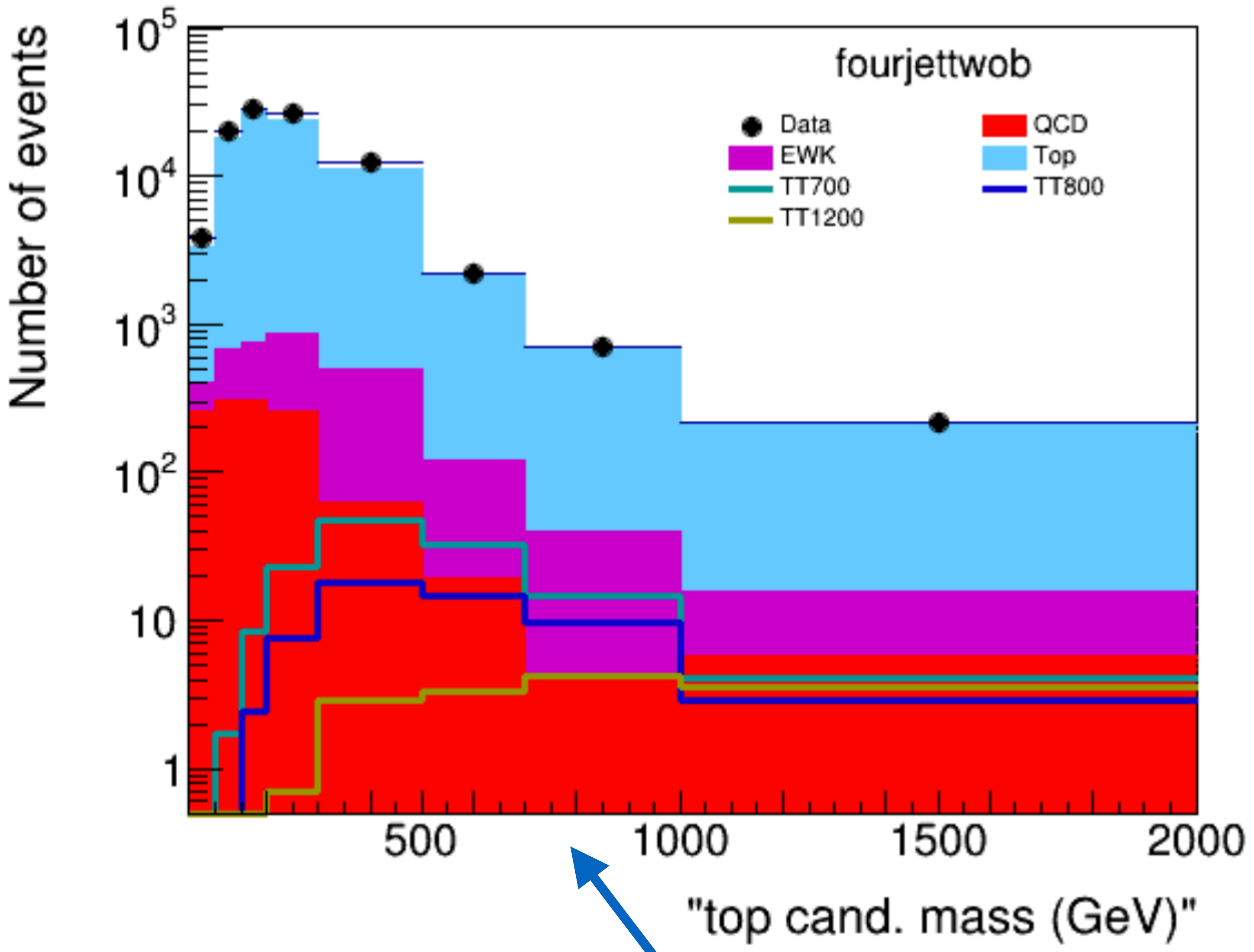
- Implemented the two ATLAS analyses for which onnx functions are available: ATLAS-SUSY-2019-04 (RPV leptons + jets) and ATLAS-SUSY-2018-30 (multi-b + MET). (analyses discussed during RIF'22.)
  - Thanks Krzysztof Rolbiecki for sharing information from the CheckMate experience!.
- Validation in progress for ATLAS-SUSY-2018-30 .

ADL/CL and LHC Open Data - I

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

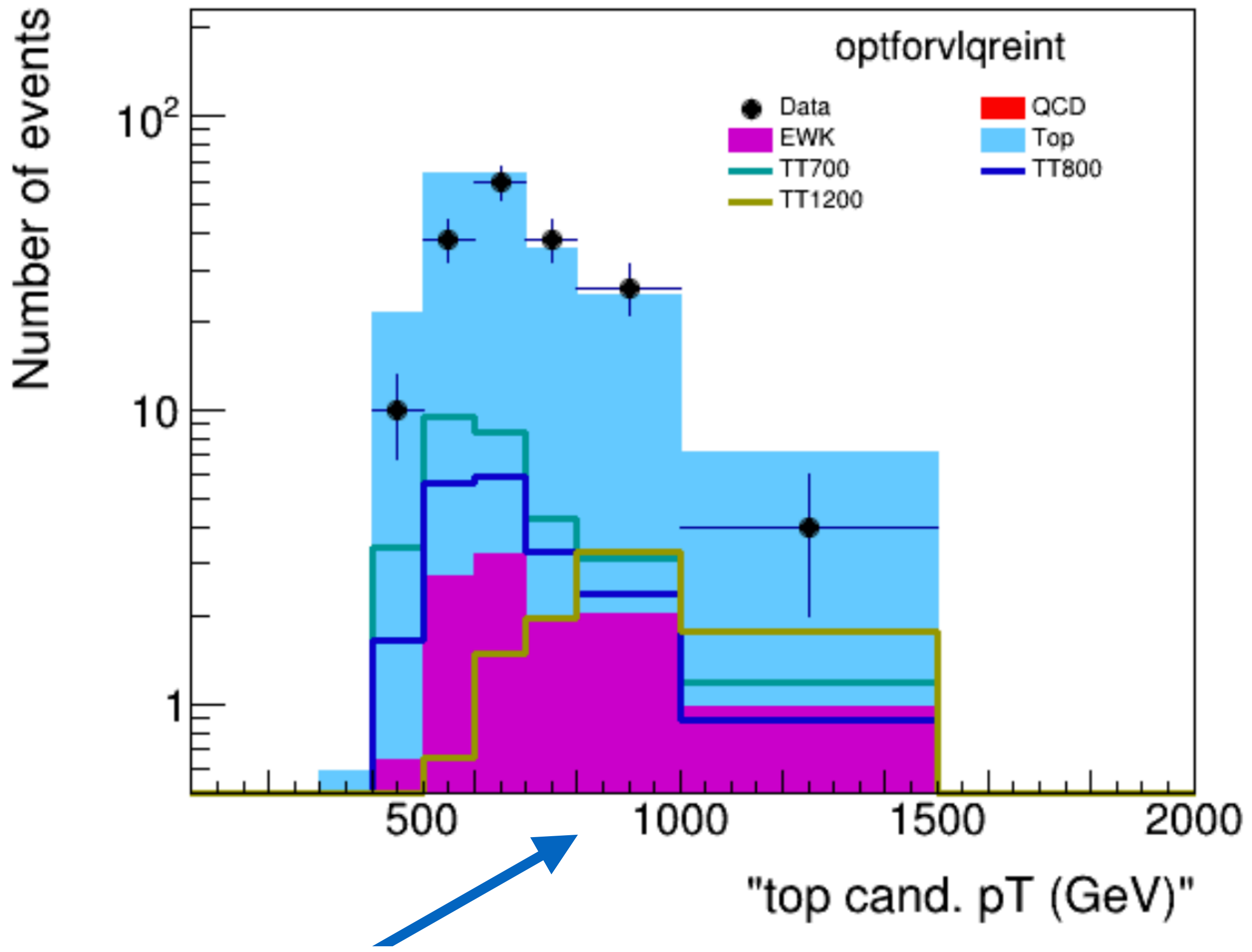
- Use related to reinterpretation: Provide capability to re-optimize and re-run recasted analyses from ADL database to maximize sensitivity to new models.
- A first example / tutorial for "reinterpretation with open data" was prepared using ADL/CL for the 2023 CMS Open Data Workshop in June 2023.
  - Study exact and "optimized" reinterpretation of a ttbar analysis for vector-like T quark signal.
  - Focus on reoptimizing the analysis to enhance sensitivity to VLT. Complete tutorial link.
  - Runs on a full set of relevant open data & MC events.
  - Runs on a docker container hosting CutLang, ROOT, xrootd access to open data, and VNC.
- Earlier complete tutorial for 2022 CMS Open Data Workshop — reimplement CMS Run 1 vector-like quark analysis with boosted W and Higgs bosons, CMS-B2G-16-024:
  - Complete tutorial link

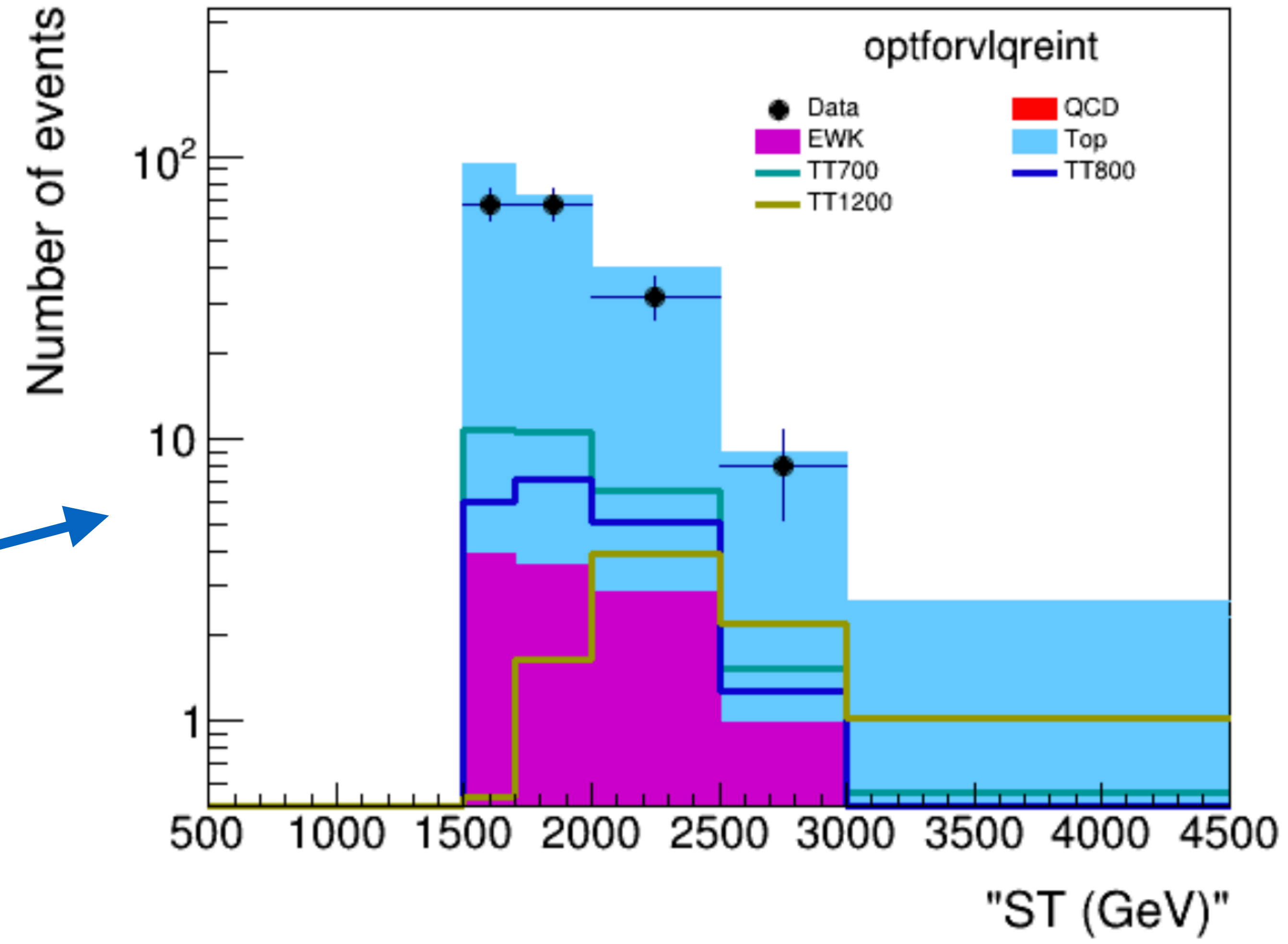
ADL/CL and LHC Open Data - II

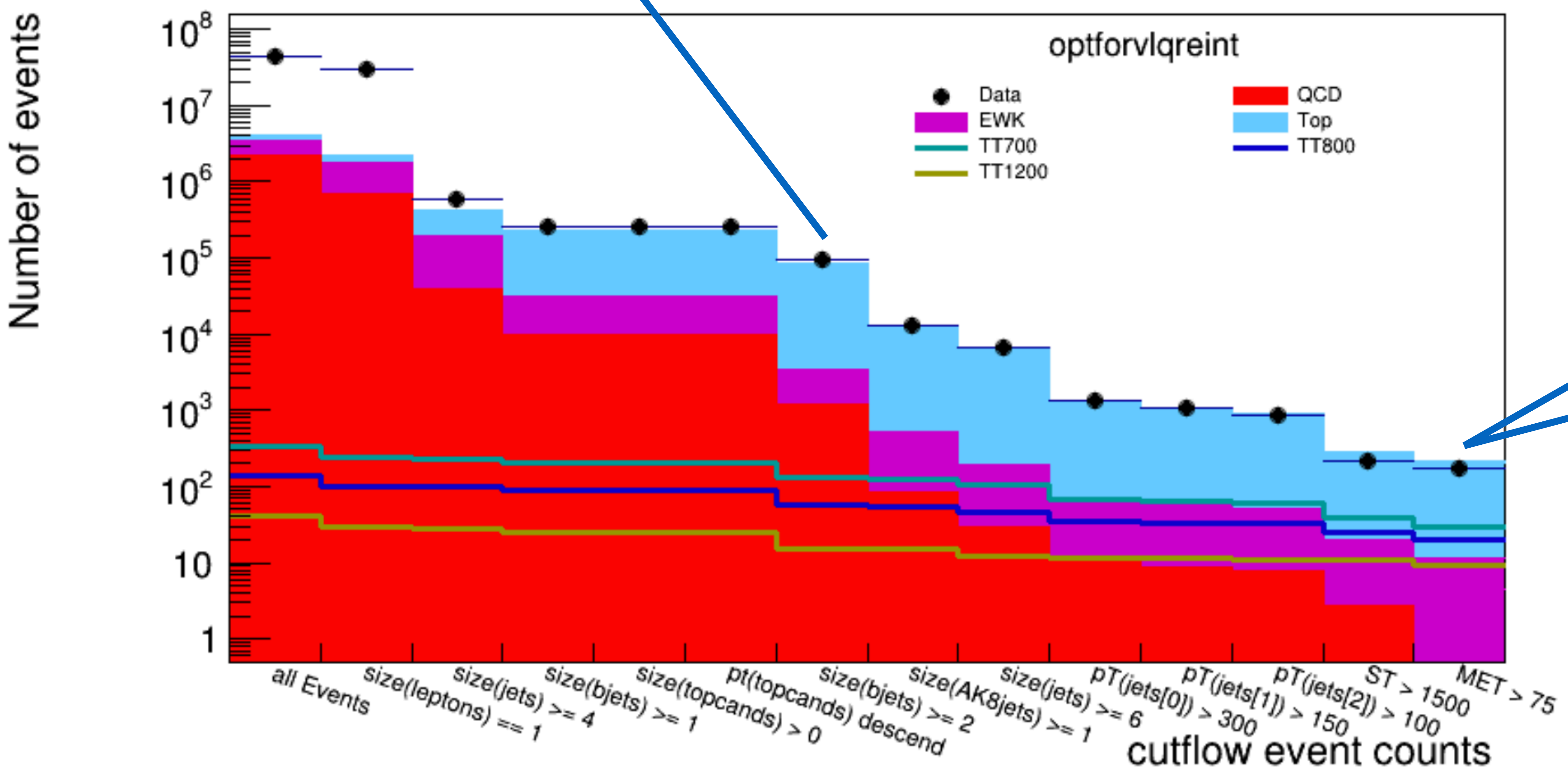


Data, BG and 2 VLT signals vs. top candidate mass for the ttbar analysis selection.

Top candidate mass pT and ST after reoptimizing the ttbar selection by adding several new cuts. (B2G-16-024 uses ST)







Cutflow histograms automatically generated by CutLang.

Histogramming and plotting tools

ADL/CL have extensive histogramming capabilities:

- 1D and 2D fixed bin and variable bin histograms defined with single line syntax.
- Histogram lists can be defined and reused in different regions and cut levels.
- Cutflow histograms and analysis bin histograms automatically drawn.

CutLang is enhanced with various easily configurable plotting tools based on standalone PyROOT or PyROOT in Jupyter notebooks (link).

- Shape comparisons between different processes.
- Weighted comparison between two processes (e.g. signal and background)
- Weighted full plotting including data, different backgrounds (stacked) and multiple signals (e.g. as in the previous page).

Recent infrastructure developments

Various developments in the CutLang core infrastructure are ongoing:

- Decoupling the grammar implementation from input data attributes, external functions, … :
  - Particle and function names are no longer needed to be hardcoded in the ADL parser.
  - After initial parsing, function and particle names are matched to those within an external library .

  => Portability of different data types, attributes, functions.
- Abstract syntax tree (AST) can be produced.
- Visualization tools: Converting analyses to graphs / flowcharts from the AST.
- Many stability and functionality improvements.

- Infrastructure in place and partially deployed. Tests ongoing.

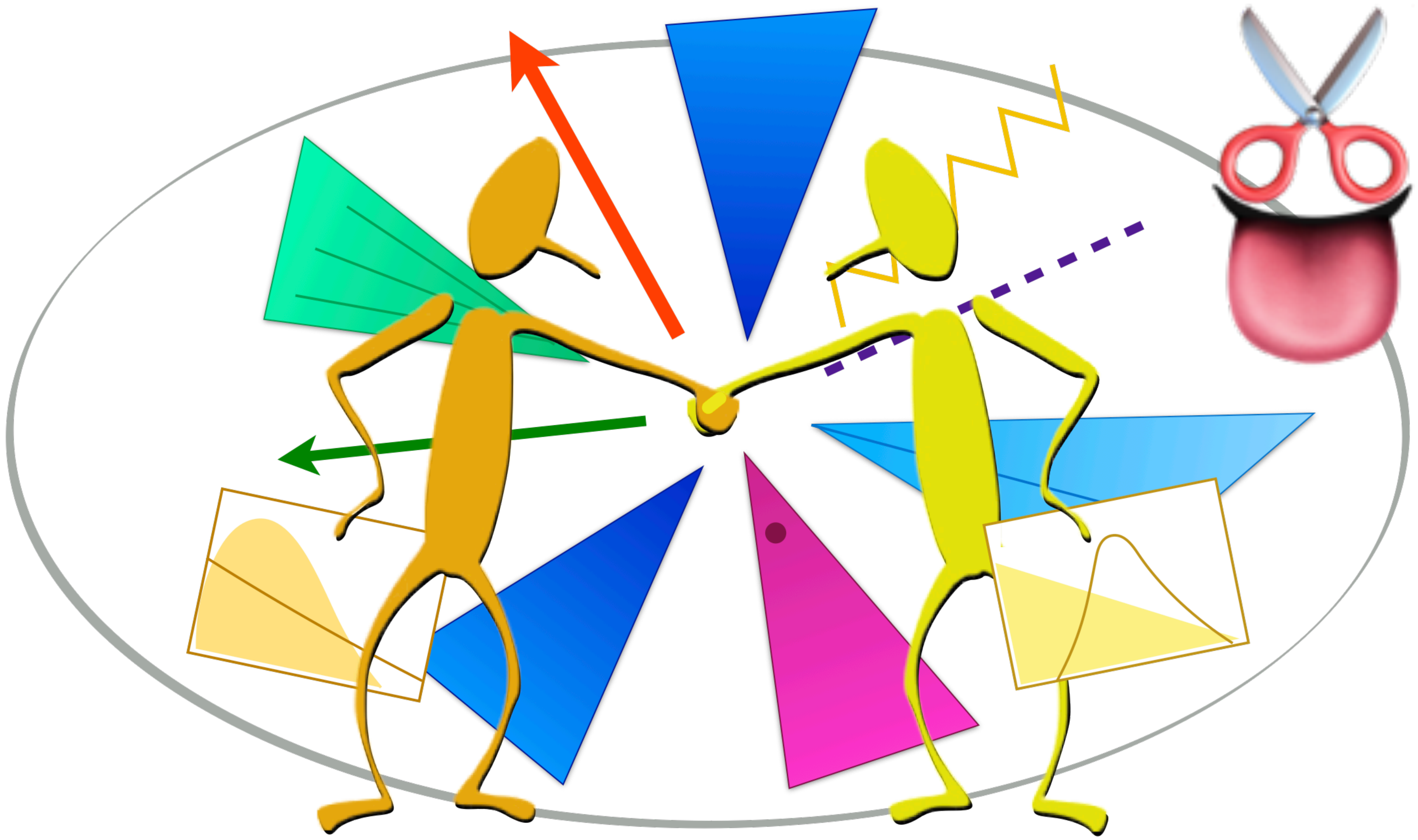
Daniel Riley, Grigory Fedyukovich (Florida State U.), Gokhan Unel (UC Irvine)

To conclude

- ADL and CutLang present a multipurpose and practical analysis approach.
- ADL/CutLang highly suitable for reinterpretation studies.
- Large scale analysis reimplementation / validation effort ongoing.
- SModelS EM-creator is used for SUSY analysis validation.
- PyROOT (and Jupyter) based plotting tools available.
- 2 complete CMS Open Data tutorials available, one featuring reinterpretation via optimization.
- ADL syntax refinements and formal compiler/interpreter infrastructure developments are ongoing.
    - ML models added,
    - ADL/CL keywords and ntuple variables now decoupled,
    - new parser with AST and automatic graphic generation is being deployed,
    - many stability and functionality improvements…

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

Documentation and references : cern.ch/adl



Extra slides

Dec 2022

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.

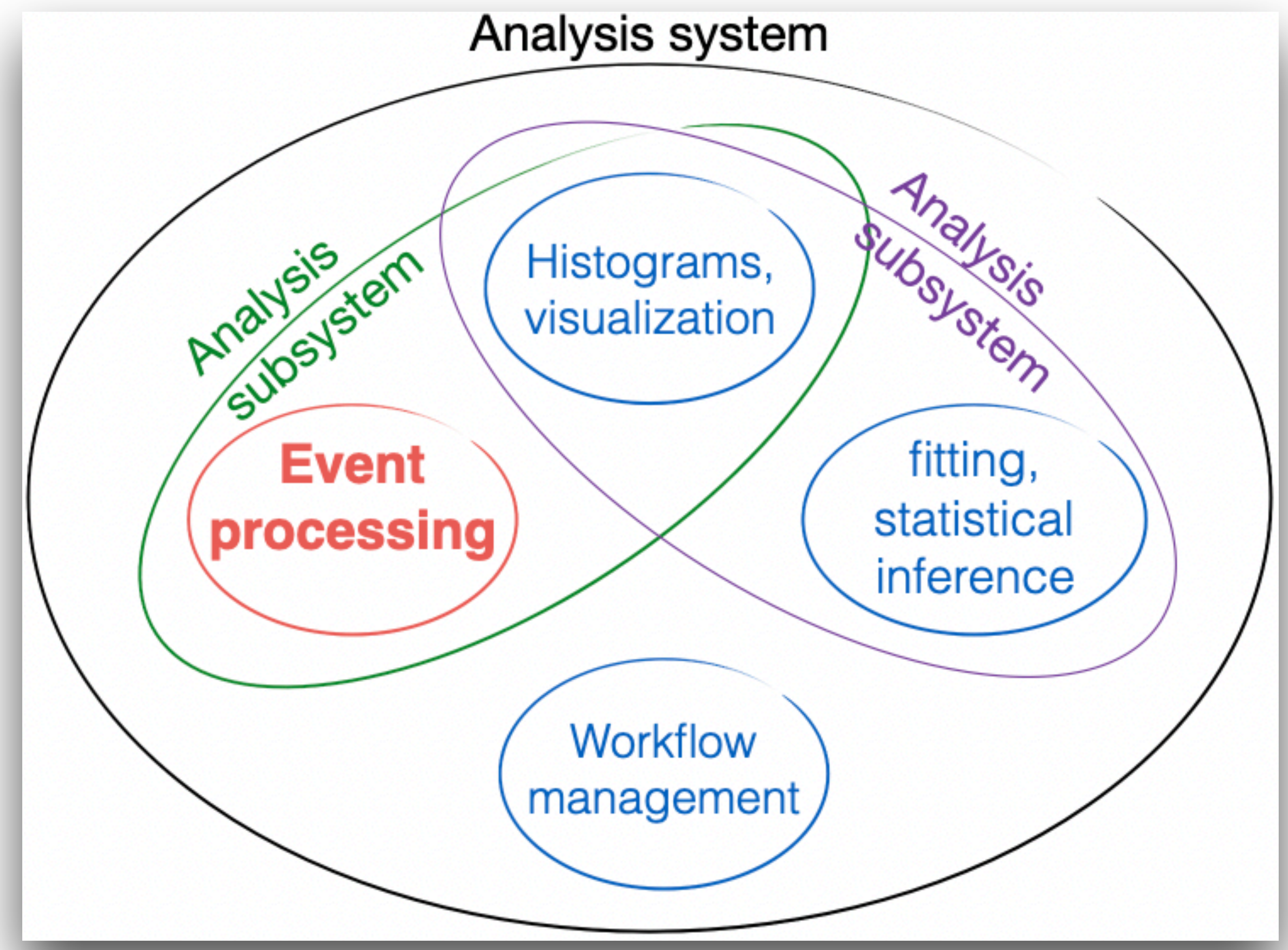
- Math functions: abs(), sqrt(), sin(), cos(), tan(), log(), …
- Collection reducers: size(), sum(), min(), max(), any(), all(),…
- HEP-specific functions: dR(), dphi(), deta(), m(), ….
- Object and collection handling: union(), comb()…

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

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, χ2 evaluation, combinatorics, unions...

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

Versatile uses of ADL - I

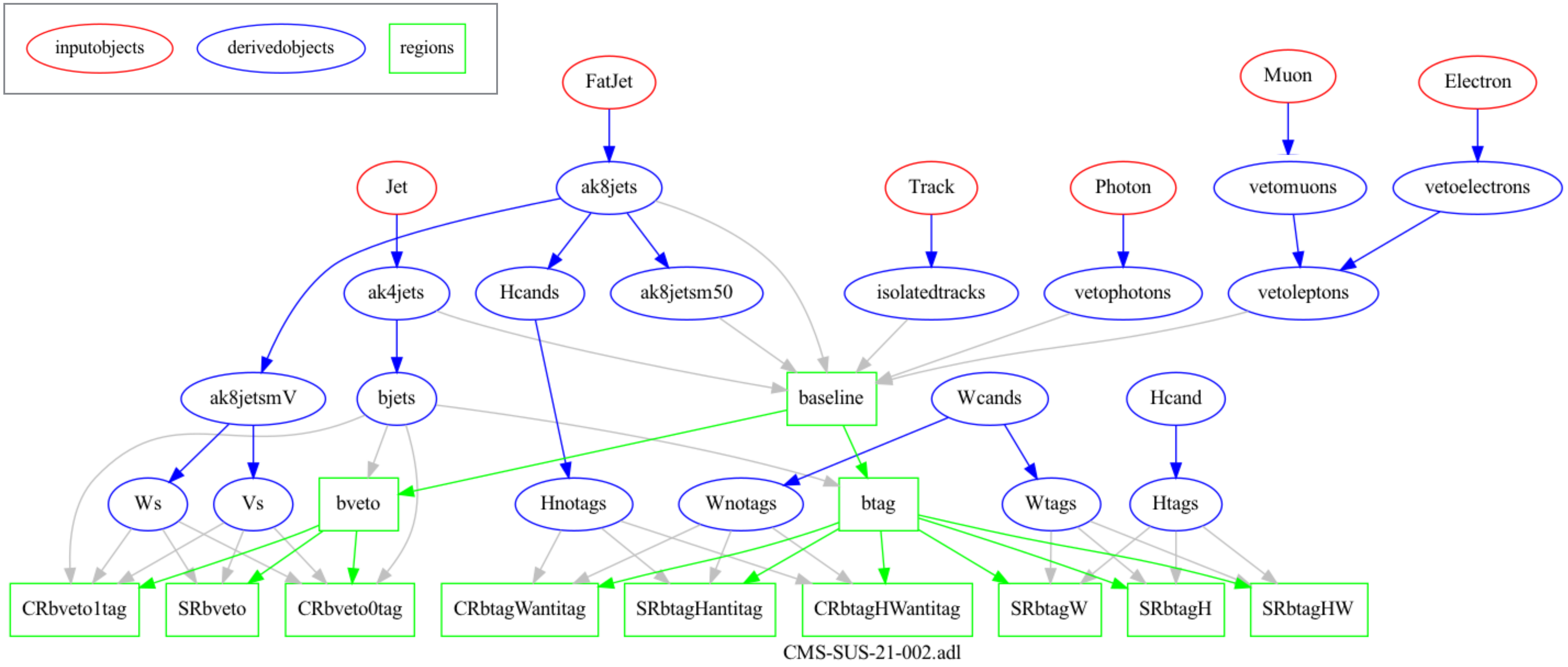
- Analysis design (experimental or pheno):
  - Quick prototyping.
  - Simultaneous test of numerous selection options in a self-documenting way.
  - Easy comparison with existing analyses: *“Was my phase space already covered?”*
- Objects handling:
  - Easy reuse in new analyses.
  - Compare object definitions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

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.)

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
```

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
```

Use ONNX Runtime with the platform of your choice

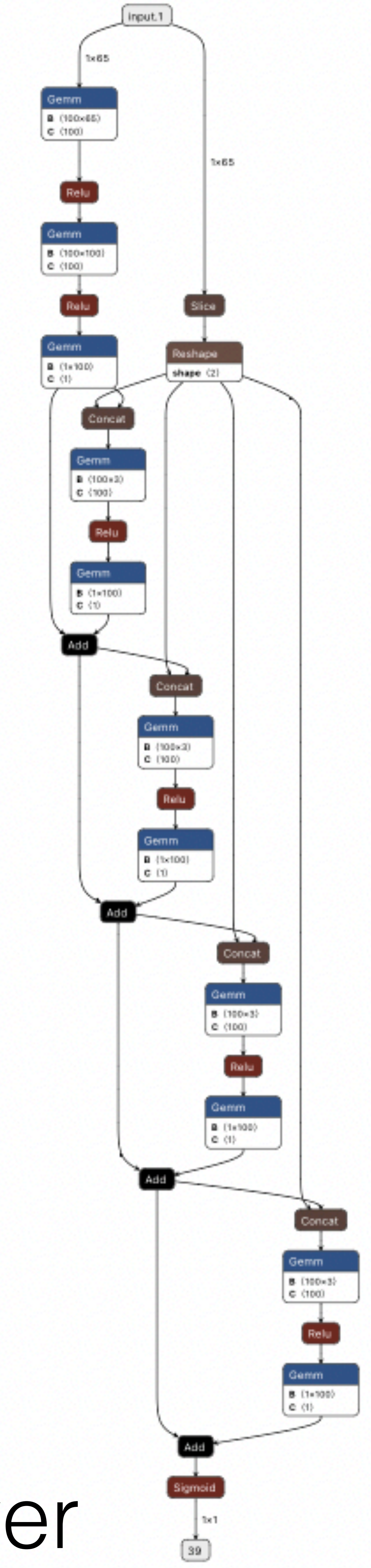
Select the configuration you want to use and run the corresponding installation script.
ONNX Runtime supports a variety of hardware and architectures to fit any need.

Optimize Inferencing | **Optimize Training**

Platform	Windows	Linux	Mac	Android	iOS	Web Browser		
API	Python	C++	C#	C	Java	JS	Obj-C	WinRT
Architecture	X64	X86	ARM64	ARM32	IBM Power			
Hardware Acceleration	Default CPU	CoreML	CUDA	DirectML				
	MIGraphX	NNAPI	oneDNN	OpenVINO				
	ROCm	QNN	TensorRT	ACL (Preview)				
	ArmNN (Preview)	Azure (Preview)	CANN (Preview)	Rockchip NPU (Preview)				
	TVM (Preview)	Vitis AI (Preview)	XNNPACK (Preview)					

Onnx Runtime Execution

OME details



onnx model viewer