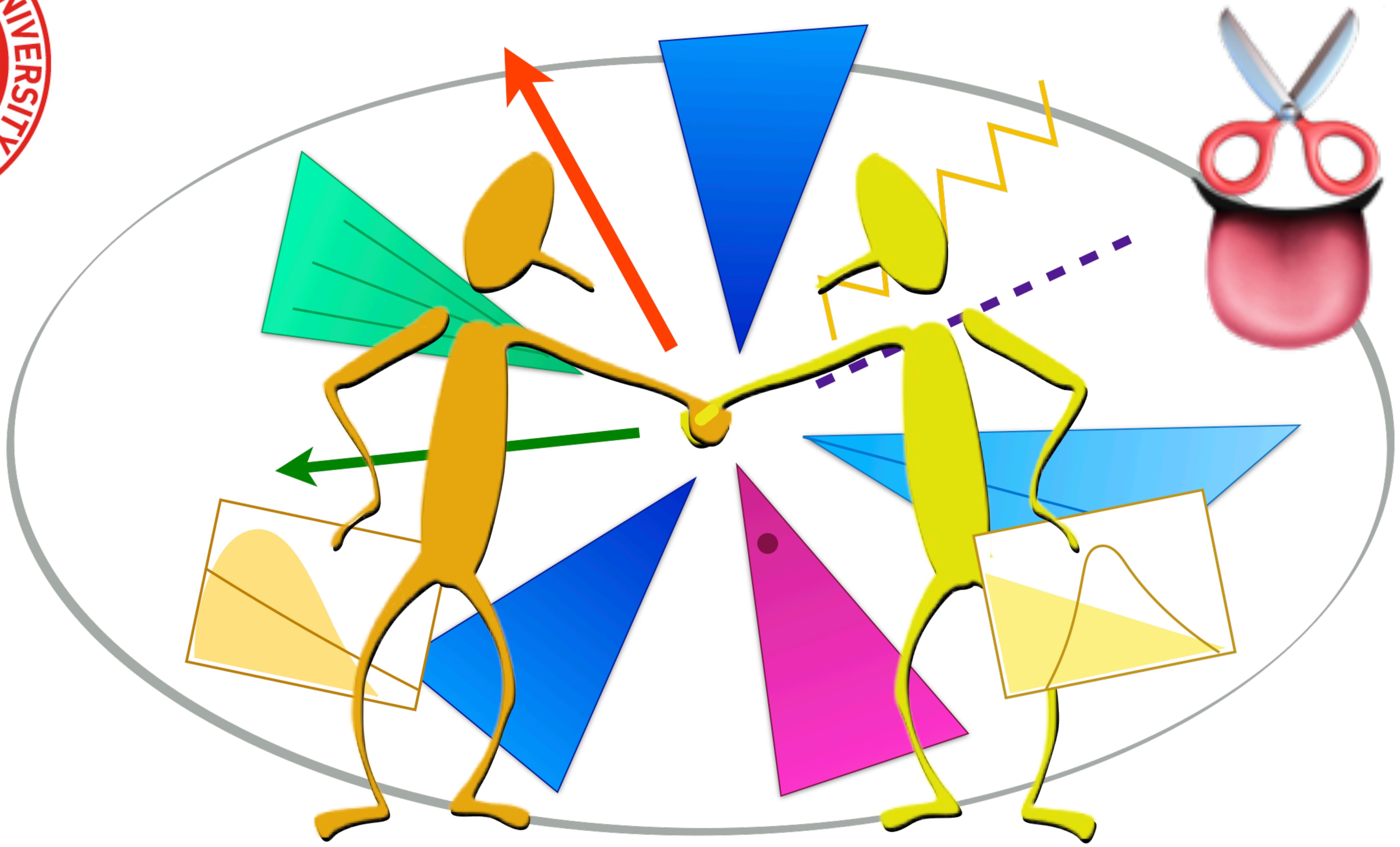
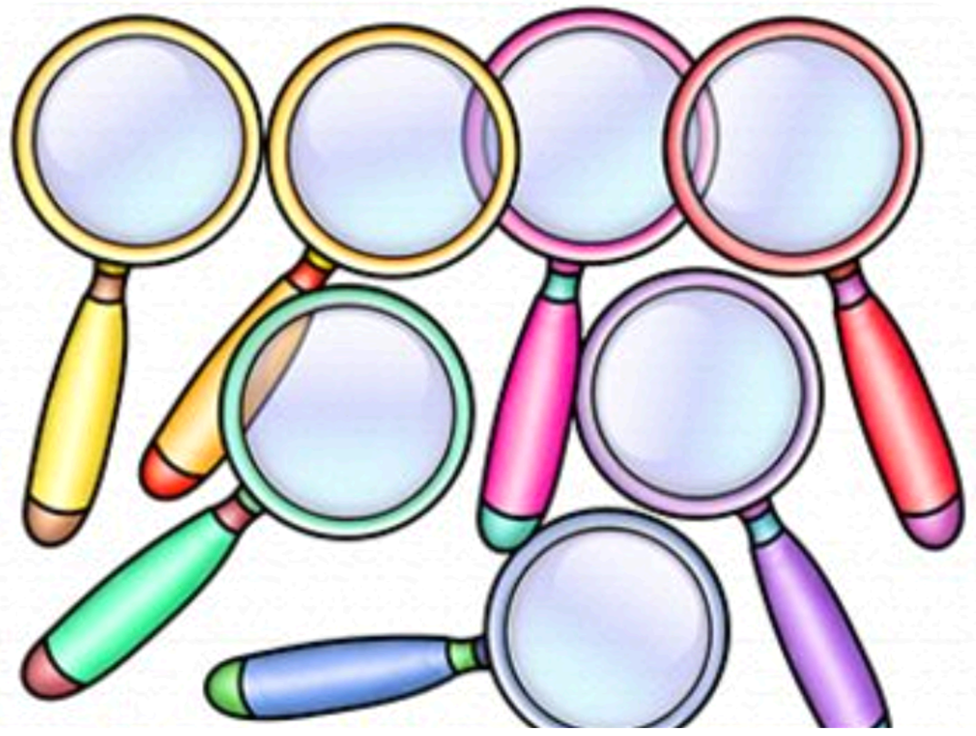


Introducing **A**nalysis **D**escription **L**anguage

Sezen Sekmen (KNU)

Analysis Description Language Tutorial & Hackathon
21-22 Nov 2022, Kyungpook National University, Center for HEP

Towards physics-focused HEP data analyses



Traditionally we write analysis physics algorithms in analysis software frameworks:

- Frameworks are based on general purpose languages like C++ / Python.
- They host the most complete description of an analysis BUT
- Physics algorithm and technical operations are intertwined and handled together.
- Physics algorithm hard to read, extract and communicate.

Could there be an alternative way of encoding analyses that

- Allows more direct interaction with data
- Decouples the physics information from purely technical tasks, thereby shifting the focus to the physics algorithm
- Improves the clarity and accessibility of analysis logic, and thereby its communicability and preservation?

According to computational science, yes.

Concept 1: Domain specific languages

General purpose language (GPL): A computer language that is broadly applicable across many application domains (C++, Python, …).

- Used for solving very wide range of problems

Domain-specific language (DSL): a computer language specialized to a particular application domain (regular expressions, Make, SQL, …).

- Designed to model how domain experts think about specific problems they wish to solve.

Embedded DSL: Implemented in a host GPL

- Benefits from already existing syntax and infrastructure, but also limited by syntax.
- Familiarity of community with syntax.
- Not fully immune to intermixing content with technical operations.

External DSL: Has custom syntax (awk, latex)

- Tailored to the semantics of the context it describes.
- Requires own interpreter / compiler infrastructure.

Concept 2: Declarative languages

Imperative programming - the current state-of-the-art in particle physics: implements algorithms in explicit steps of commands to perform, describes control flow —> focuses on how to execute. (C++, Python, …)

- Implements algorithms in explicit steps, explicit looping.

Declarative programming - the candidate paradigm for particle physics: expresses the logic of a computation without describing its control flow —> focuses on what to execute. (LISP, SQL, Prolog, Haskell, …)

- Higher level programming
- No implementation details, no explicit looping.
- clear correspondence to math logic.
- clear, concise definitions.

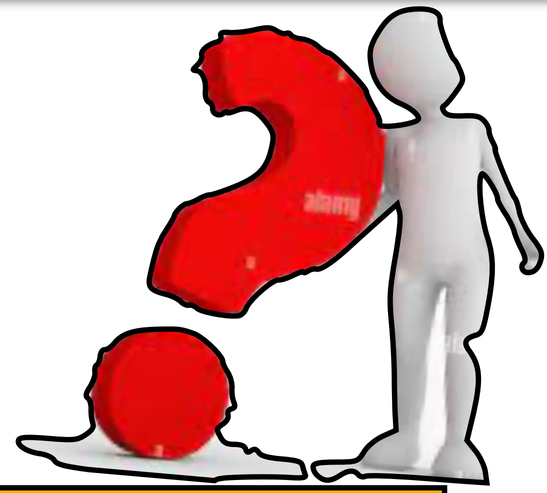
HOW → WHAT

A look at computing history:

Trends change to minimize effort, and maximize efficiency

PC domination & rise of Python

programming by voice?



80s: Age of Unix and C

1989-2000
LEP

2010- ...
LHC

1983-2011
Tevatron

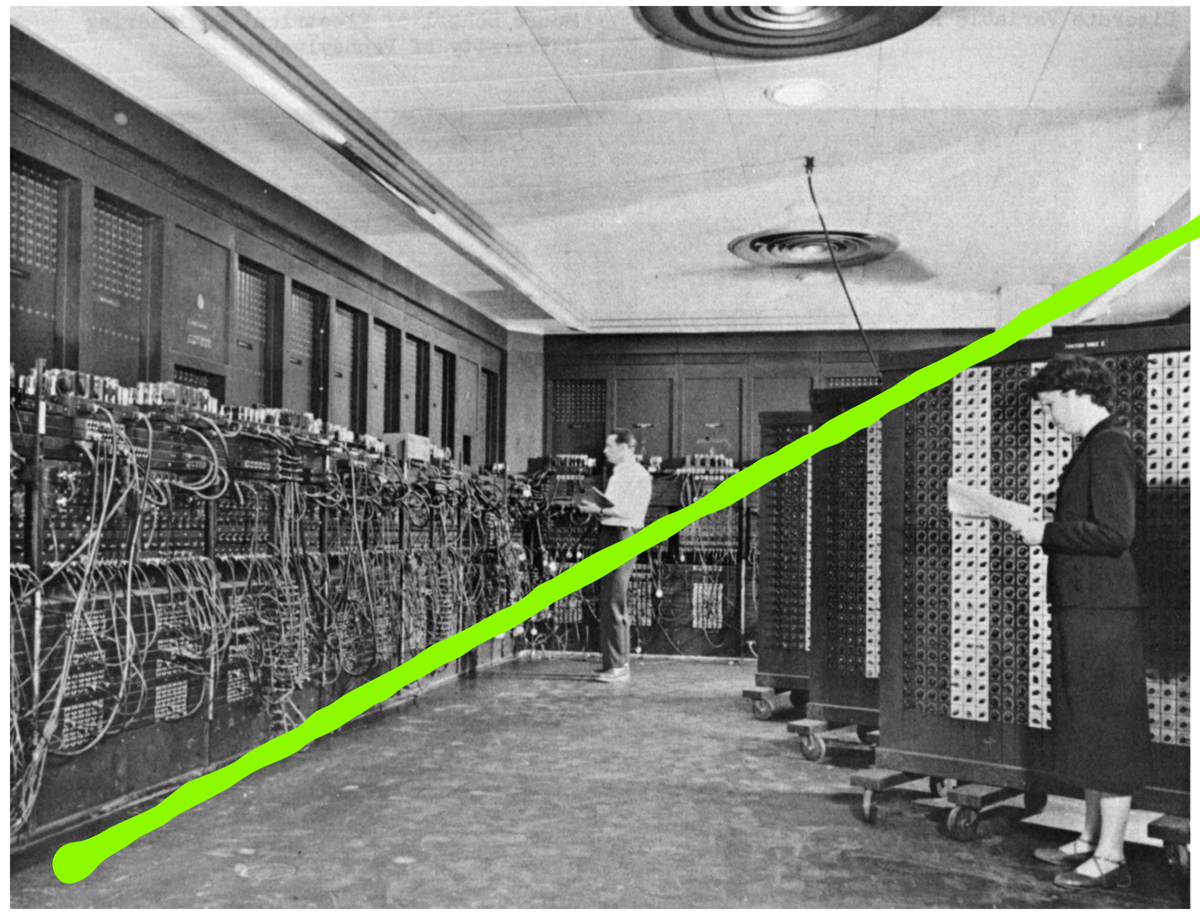
…what

1981-1991
SPPS

the OOP paradigm and C++

how…

1945: programming by wire



60s & 70s: Age of Fortran

slide by G. Unel

Analysis description language

We can apply these concepts to particle physics analysis:

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: Tells what to do in an analysis but not how to do it.
- Easy to read and understand: Clear, self-describing syntax rules.
- Designed for everyone: experimentalists, phenomenologists, students, interested public…

Analysis description language

ADL is a language. It is framework-independent

—> Analyses written with ADL can be translated / integrated into any language or framework to various analysis-related tasks.

This leads to various advantages:

- Multi-purpose use: Analysis algorithms written in ADL 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 different groups: exp, pheno, referees, students, public.
- Easy preservation of analysis physics algorithm for future use.

A little history: 2 parallel efforts towards ADL

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: Why and how we started

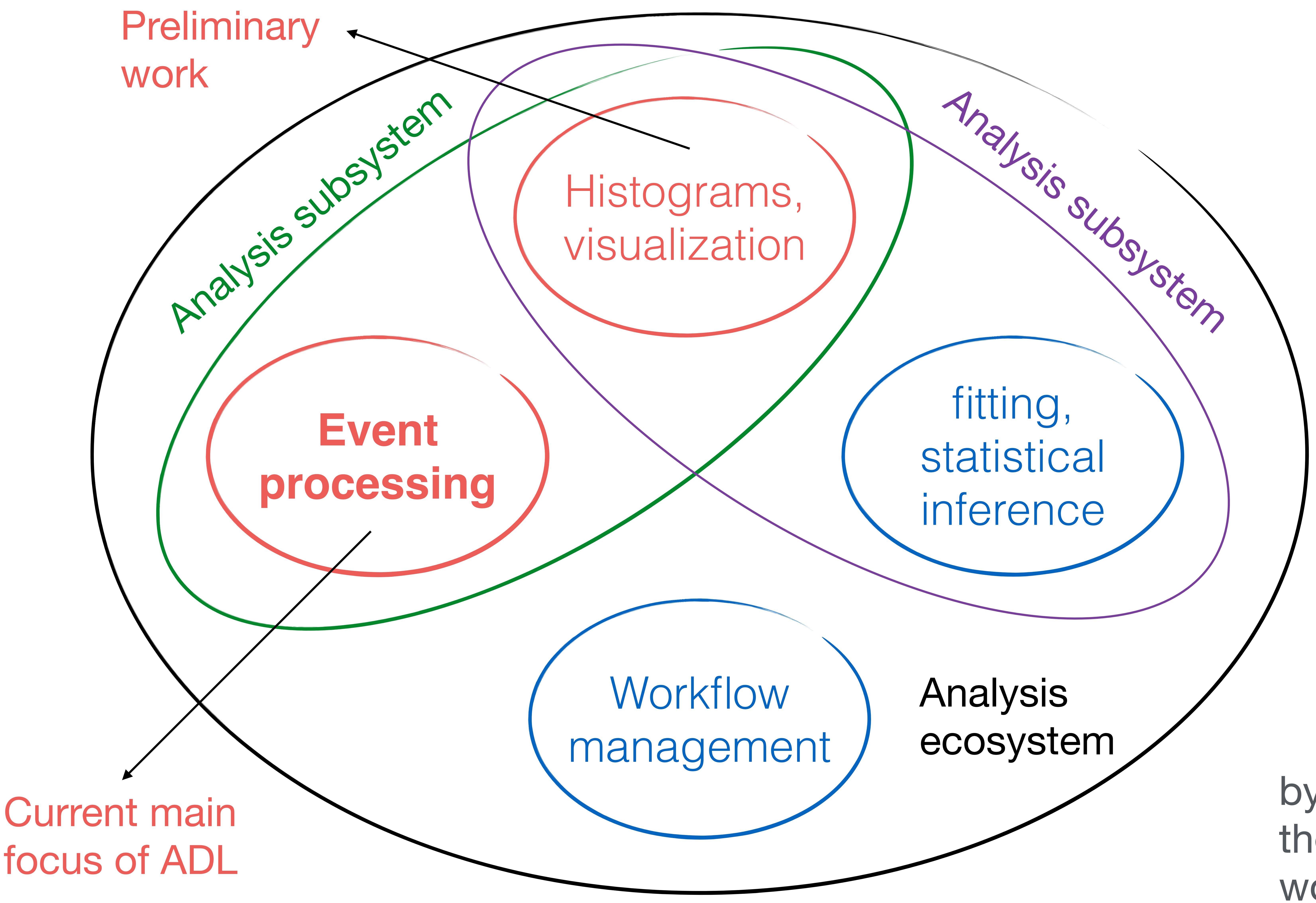
CutLang: A DSL (2015-2019) + runtime interpreter (2015-…)
arXiv:1801.05727 , arXiv:1909.10621 , arXiv:2101.09131 .

- 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.

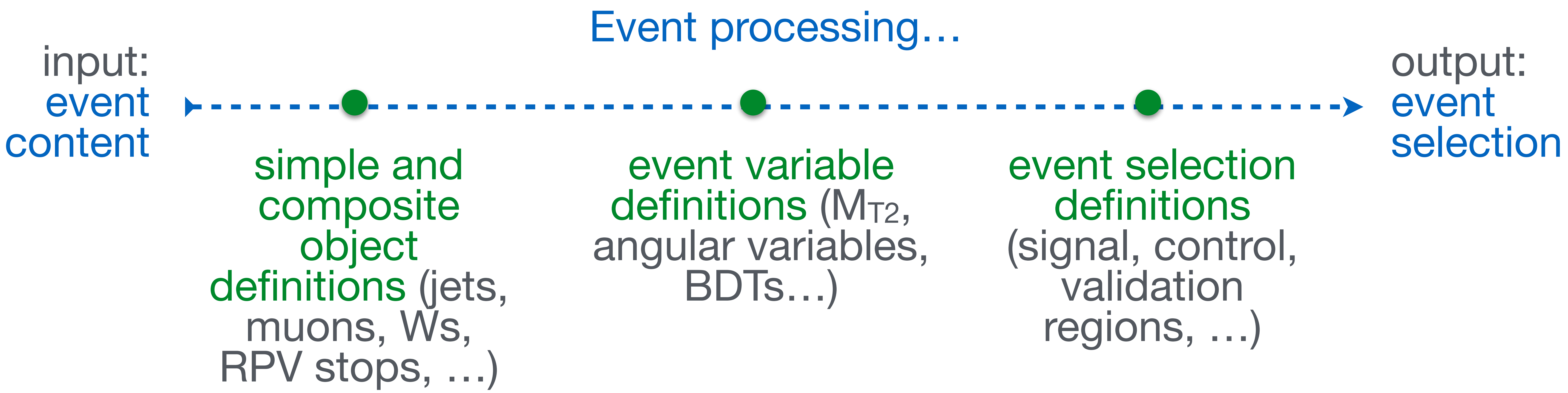
ADL scope



by Jim Pivarski from the Fermilab ADL workshop

ADL scope

- Event processing: Priority focus.



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

The ADL construct

ADL consists of

- a plain text ADL file describing the analysis algorithm 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.
  - 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 rules 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

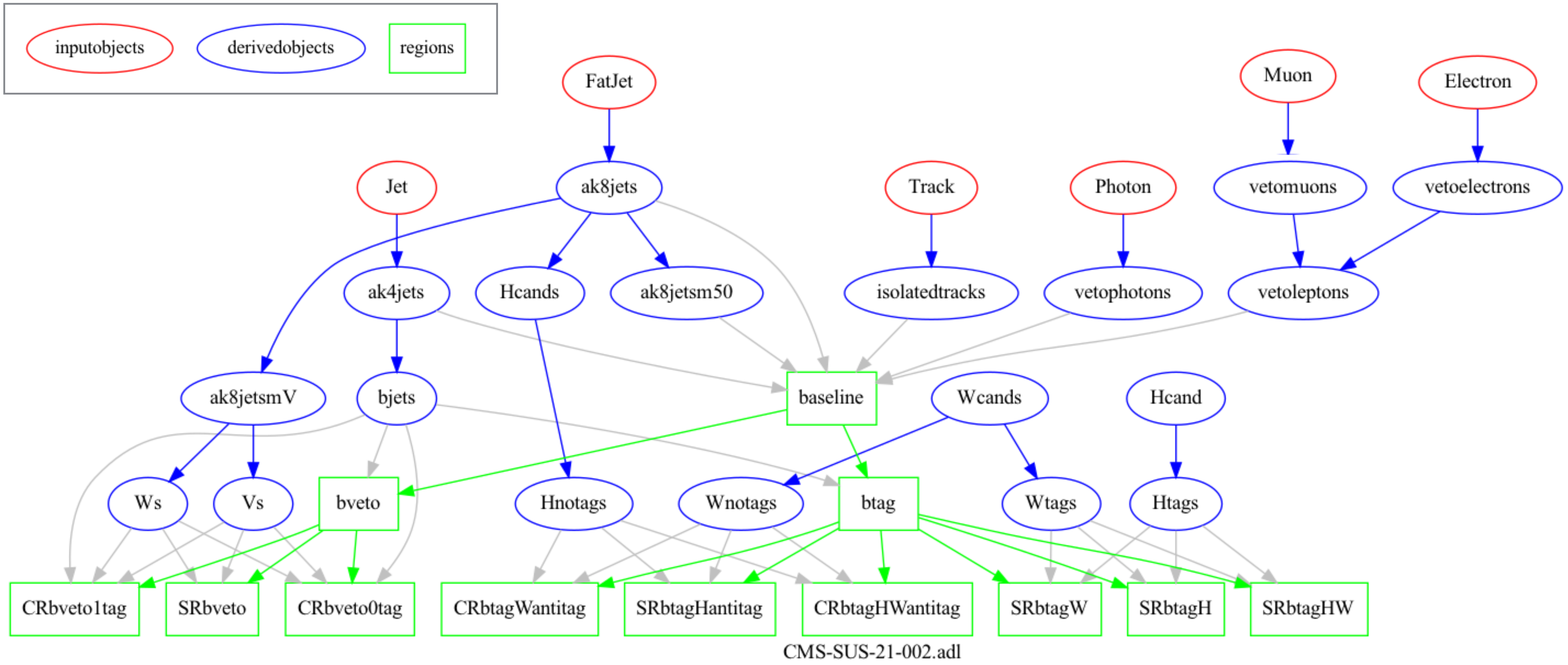
Simple SUSY analysis with ADL

```
# OBJECTS
object goodJet
  take jet
  select pT(jet) > 30
  select abs(eta(jet)) < 2.4

object goodMuon
  take muon
  select pT(muon) > 30
  select abs(eta(muon)) < 2.4

object goodEle
  take Ele
  select pT(ele) > 30
  select abs(eta(ele)) < 2.5

object goodLep
  take union(goodEle, goodMuo)
```

```
# EVENT VARIABLES
define HT = fHT(jets)
define MTl = Sqrt( 2*pT(goodLep[0]) * MET*(1-cos(phi(METLV[0]) - phi(goodLep[0]) )))

# EVENT SELECTION
region SR
  select size(jets) >= 2
  select HT > 200
  select MET > 200
  select MET / HT <= 1
  select Size(goodEle) == 0
  select Size(goodMuon) == 0
  select dphi(METLV[0], jets[0]) > 0.5
  select dphi(METLV[0], jets[1]) > 0.5
  select size(jets) >= 3 ? dphi(METLV[0], jets[2]) > 0.3 : ALL
  select size(jets) >= 4 ? dphi(METLV[0], jets[3]) > 0.3 : ALL
  histo hMET , "met (GeV)", 40, 200, 1200, MET
  histo hHT , "HT (GeV)", 40, 200, 1600, HT
```

ADL analyses on github: https://github.com/ADL4HEP/ADLLHCanalyses

ADL is particularly useful in describing complex analyses



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

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
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

Further syntax available for writing existing analysis results (e.g. data counts, BG estimation or signal counts & errors, cutflows, …)

ADL syntax rules with usage examples: https://twiki.cern.ch/twiki/bin/view/LHCPhysics/ADL
Comprehensive ADL/CutLang user's manual: Appendix of arXiv:2101.09031.

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, …

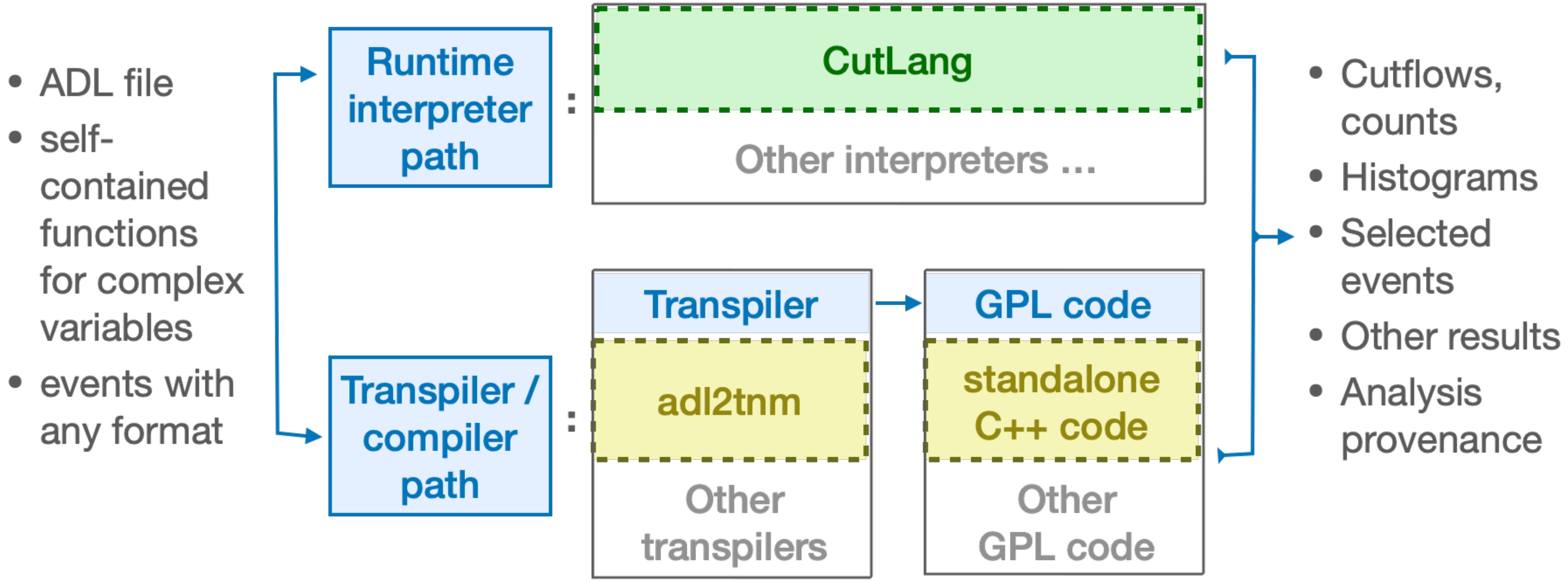
ADL syntax rules with usage examples: https://twiki.cern.ch/twiki/bin/view/LHCPhysics/ADL
Comprehensive ADL/CutLang user's manual: Appendix of arXiv:2101.09031.

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 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 ATLAS & CMS analyses: https://github.com/ADL4HEP/ADLLHCanalyses

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 analysis 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

Education with ADL / CutLang

ADL is a practical way to teach analysis to both professional and non-professional physicists:

- ADL analyses provide a learning database for analysis ideas and techniques.
- Learners can immediately start to write and run analyses with this approach.

Some training activities with ADL/CutLang:

- 5th School of Computing Applications in Particle Physics (3-7 Feb 2020, Istanbul, Turkey): ADL/CutLang school (indico link , proceedings published in Eur. J. Phys. 42 035802 (2021))
- VSOP-26 (Nov 29-Dec 11 2020, Guy Nhon / online): Hands-on ADL/CL analysis exercise for junior TH students with no analysis / unix / ROOT experience (indico link)
- Tutorials at CMS Open Data workshops.

LHC Open Data analysis with ADL/CutLang

ADL/CutLang can be already used for analysis of various LHC Open Data & MC.

Working with ATLAS & CMS OD teams to establish ADL/CutLang as an OD analysis model.

CMS Open Data

CutLang can run on the following OD formats:

- Reduced ntuples (for education)
- NanoAOD (research): Refinements ongoing.

ADL/CL OD analysis demos / tutorials:

- CMS OD4Theorists workshop - 30 Sep - 2 Oct 2020, CERN (indico link)
- CERN Summer Student Workshop - 6 Aug 2021, CERN (indico link)
- 2022 CMS Open Data Workshop - 1-4 Aug 2022, CERN (indico link)

ATLAS Open Data

Available Run1 & Run2 OD mostly used for education & outreach.

- CutLang can run on ATLAS Run1 and Run2 OD ntuple formats.

Available analysis examples:

- Z -> ll, H -> γγ (more in progress).

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.

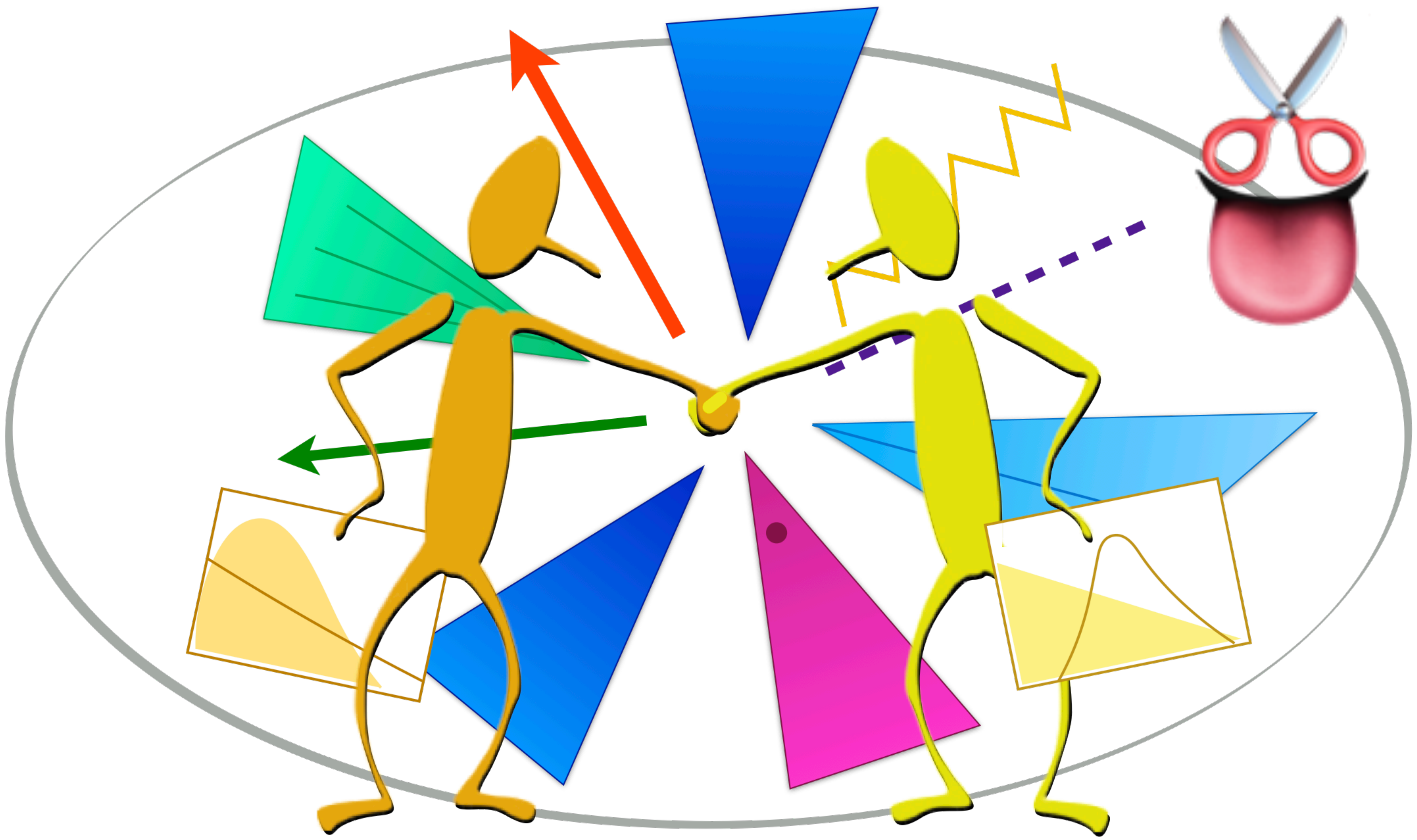
But its distinguishing strength is in navigating and exploring the multi-analysis landscape.

(more on this tomorrow)

What does the community think?

ADL is gaining increasing recognition in the HEP / LHC communities:

- Declarative languages (with ADL as an example) are recommended in the HSF IRIS-HEP Second Analysis Ecosystem Workshop Report (2022) and presented as a priority areas to be pursued.
- ADL is recommended as a common analysis documentation and preservation method by the CMS Analysis Tools Task Force (2022) (internal CMS report).
- ADL is recommended as a reinterpretation and analysis preservation method in the Snowmass 2021 Computational Frontier "Reinterpretation and Long-Term Preservation of Data and Code" report (2022).
- Advantages of DSLs for analysis are highlighted in the Snowmass 2021 Computational Frontier End User Analysis Report (2022).



In summary:

- ADL is an emerging, paradigm-shifting approach that puts physics and physicists at the center of a particle physics data analysis.
- CutLang is the first successful runtime interpreter for HEP analyses.
- Many physics and education use cases confirm the feasibility of this approach.
- ADL syntax and tools are under constant development.
- We invite the you 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.