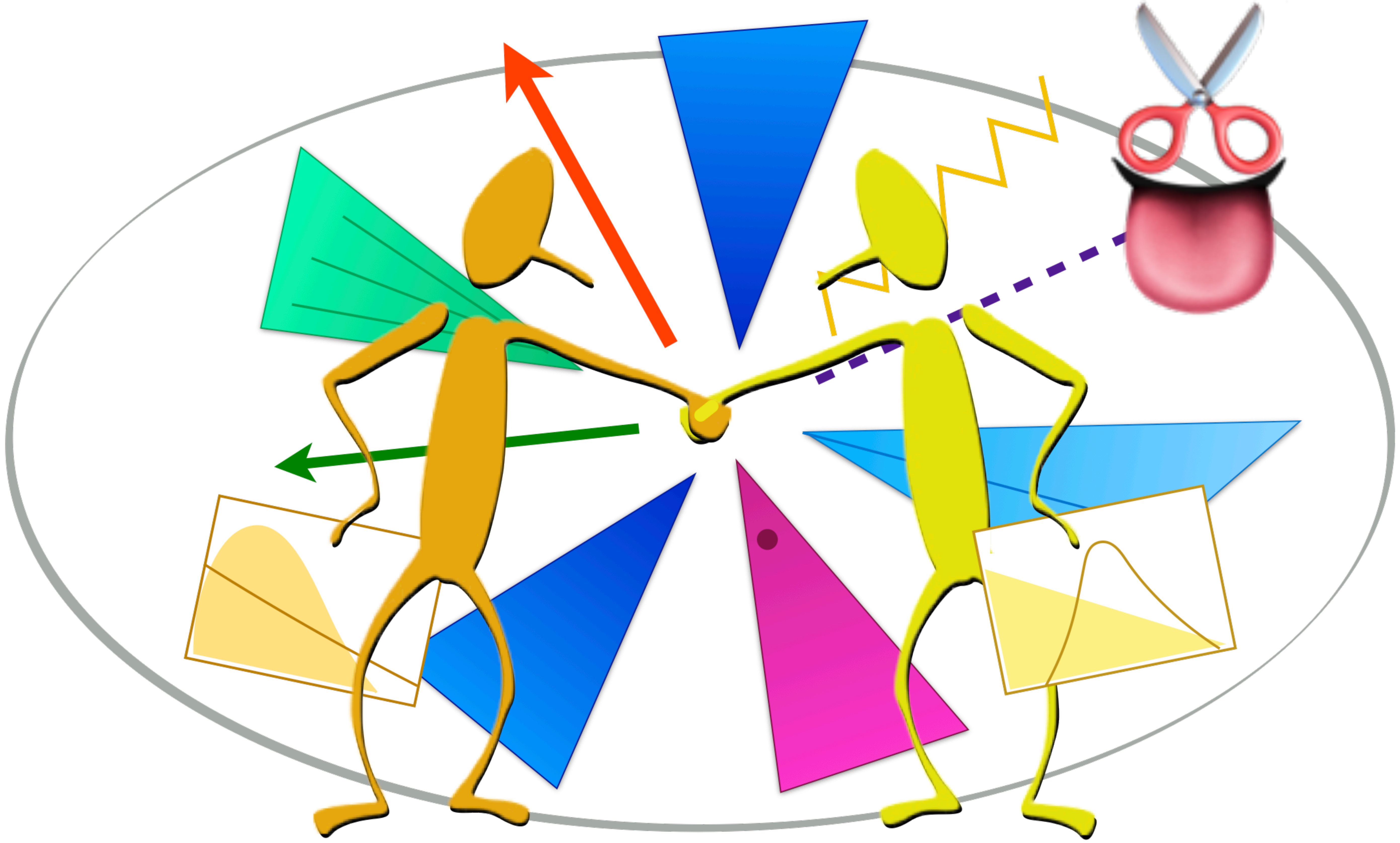


Run2 new physics search example with ADL/CutLang

Sezen Sekmen (Kyungpook Nat. U., CMS)
Gökhan Ünel (UC Irvine - ATLAS)
Burak Şen (METU)

CMS Open Data Workshop 2022
1-4 August 2022, CERN

Tutorial page : link
ADL Documentation and references : cern.ch/adl

Other contributors: Ronja Ohrnberg (Helsinki), Junghyun Lee (KNU)

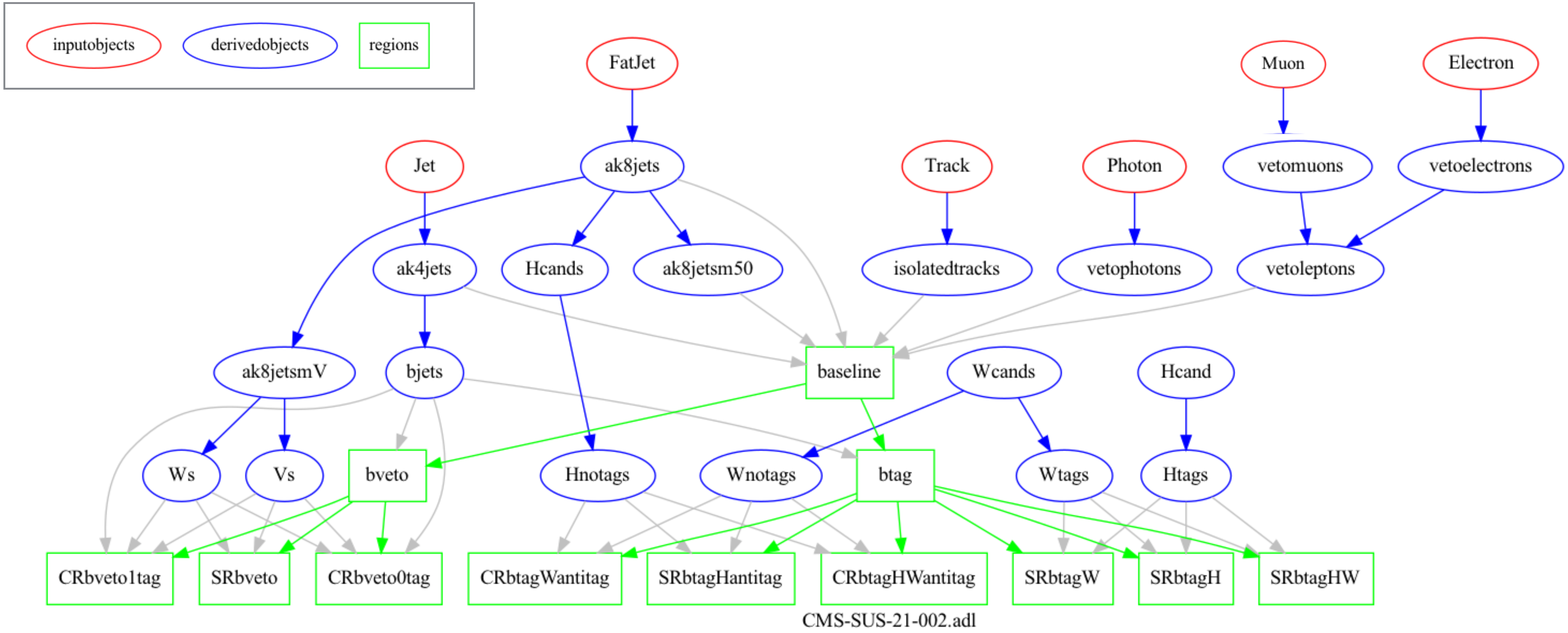
The traditional approach

We usually perform analyses using using analysis software frameworks based on general purpose languages like C++ or Python.

- Flexible method.
- Straightforward for simple analyses
- New systems based on Python allow easy access to libraries —> we can easily perform a large set of analysis tasks.

What if we have a more complex analysis?

Different objects, different regions, most depending on each other.



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

Emerging alternative: ADL

We can of course write this analysis with any GPL-based system.

However, it becomes increasingly hard to visualize and keep track of the physics algorithm details.

Main reason for complexity: Physics content and technical operations are intertwined and handled together.

More intricate physics algorithm —> more complex code.

Emerging approach: Analysis Description Language (ADL):

- Write the physics logic with a customized, self-describing syntax.
- Decouple the physics algorithm from purely technical tasks
- Describe analyses in a more intuitive and physics-focused way.

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.

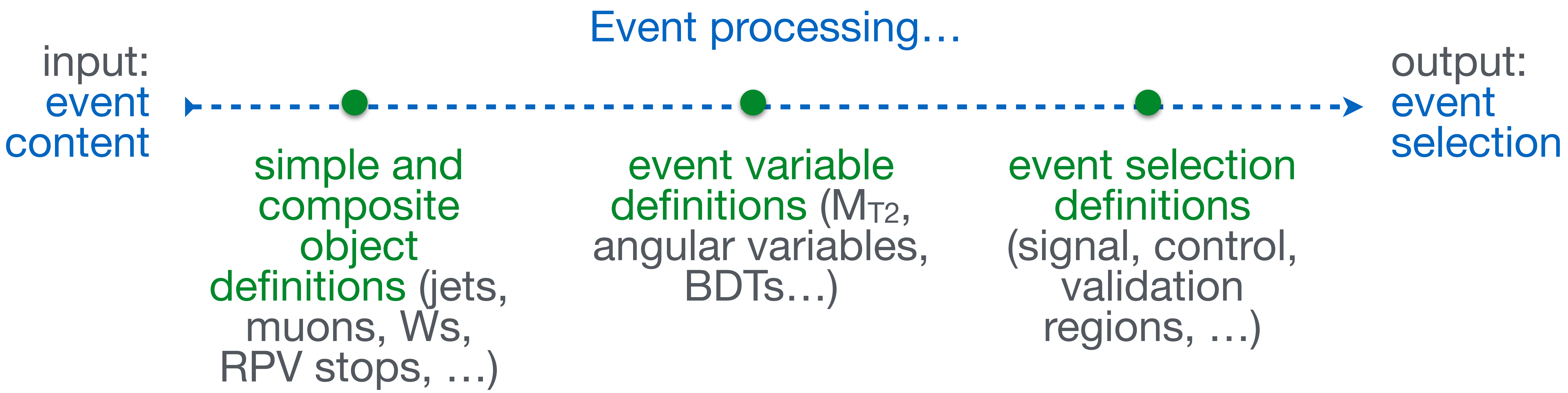
- External DSL: Custom-designed syntax to express analysis-specific concepts. Reflects conceptual reasoning of particle physicists. Focus on physics, not on programming.
- Declarative: Tells what to do, but not how to do it.
- Human-readible: Clear, self-describing syntax rules.
- Designed for everyone: 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, …
- Efficient communication between groups: exp, pheno, referees, students, public, …
- Accessible preservation of analysis physics 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 within the scope yet.

The ADL construct

cern.ch/adl

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

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

CutLang

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

This is achieved by CutLang , the runtime interpreter that reads and understands the ADL syntax and runs it on events.

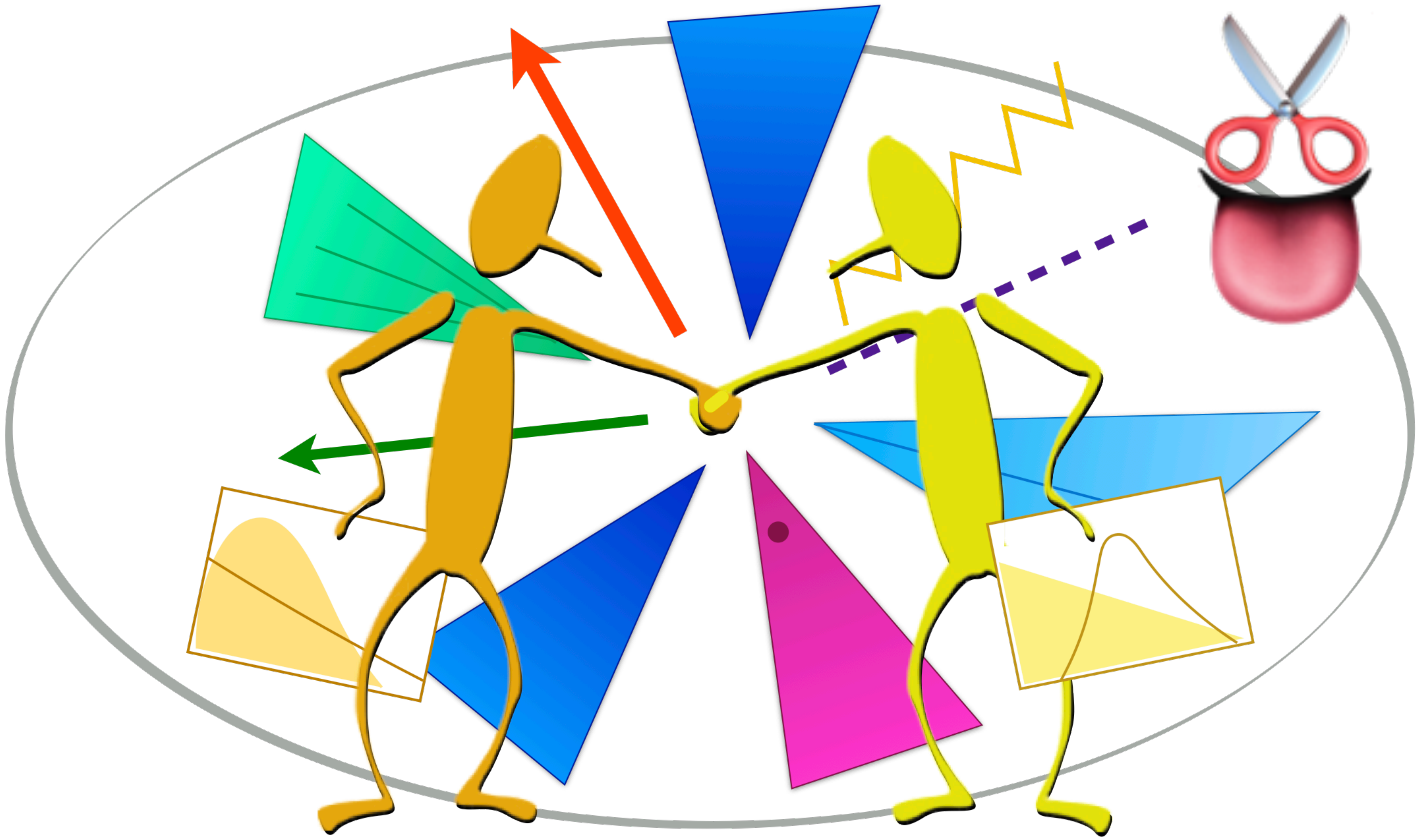
- A runtime interpreter does not require to be compiled.
- The user only modifies the ADL description, and runs CutLang.
- CutLang is also a framework which automatically handles many tedious tasks as reading input events, writing output histograms, etc.
- CutLang runs on various environments such as linux, mac, conda, docker, jupyter, etc.

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 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.

An ADL analysis database is a great source of information and can be used for performing physics studies.



Tutorial time!

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