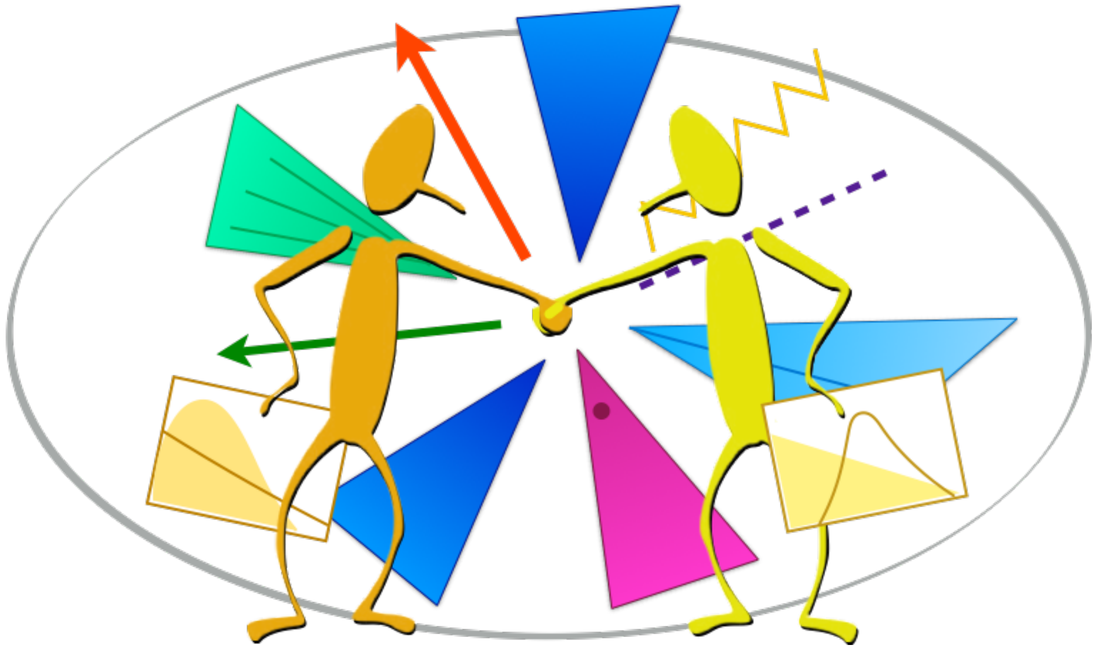


Experimental methods 2 hands-on exercise using Analysis Description Language and CutLang

Sezen Sekmen (Kyungpook National U.)
with support from Gokhan Unel (UC Irvine)

26th Vietnam School of Physics
29 Nov - 11 Dec 2020

Analysis in CMS

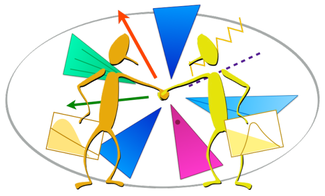
In high energy physics, we perform analyses using frameworks:

- The frameworks are based on general purpose languages like C++ / Python.
- Physics content and technical operations are intertwined and handled together.

Could there be an alternative way which

- allows for more direct interaction with data and
- decouples physics information from purely technical tasks?

Can we consider domain specific languages for LHC analyses?



Analysis description language for LHC

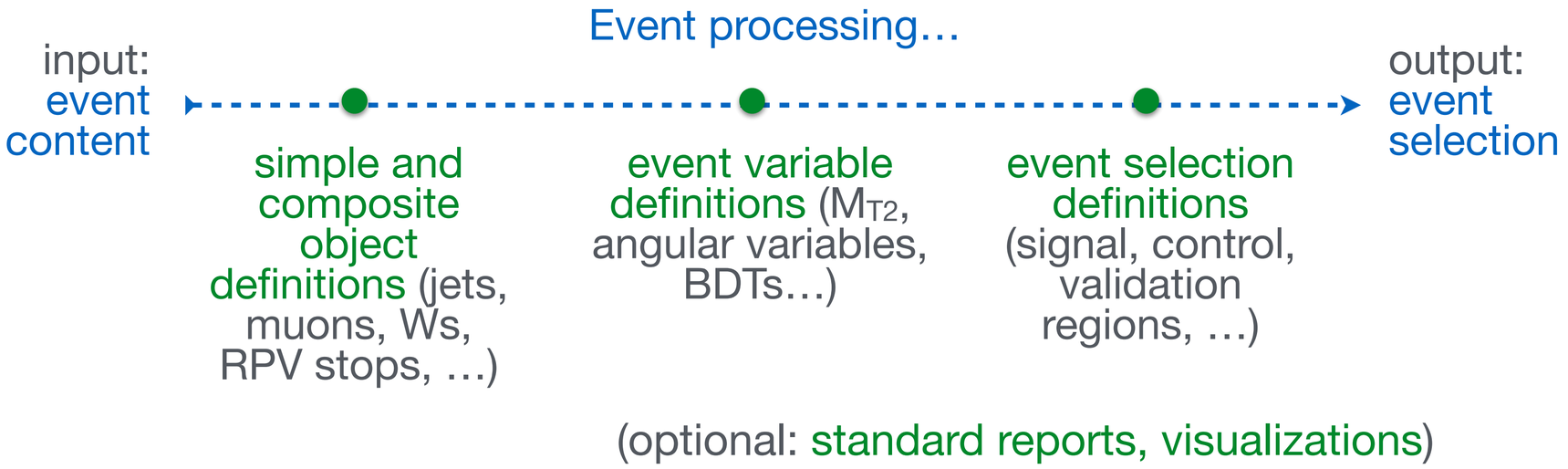
Analysis Description Language (ADL) for the LHC is:

- A domain specific and declarative language capable of describing the physics content of an LHC analysis in a standard and unambiguous way.
  - Customized to express analysis-specific concepts.
- Designed for use by anyone with an interest in, and knowledge of, LHC physics: experimentalists, phenomenologists, other enthusiasts…

- Very suitable for introducing beginners to particle physics data analysis.

Language scope

By construction, ADL is not designed to be general purpose; therefore, getting the right scope is key. The core of ADL would include:



Further operations with selected events (background estimation methods, scale factor derivations, etc.) can vary greatly, and thus may not be easily considered within a standard language scope.

LHADA/CutLang —> ADL

cern.ch/adl

ADL consists of

- a plain text file describing the analysis 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 database with 15 example LHC analyses:
https://github.com/ADL4HEP/ADLLHCanalyses

- Separate object, variable, event selection definitions into blocks with a keyword value structure, where keywords specify analysis concepts and operations.

```
blocktype blockname
  keyword1 value1
  keyword1 value2
  keyword3 value3 # comment
```

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

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), ACAT 2019 proceedings (arXiv:1909.10621)

ADL syntax: blocks, keywords, operators

Block purpose	Block keyword
object definition blocks	object
event selection blocks	region
analysis information	info
tables of results, etc.	table

Keyword purpose	Keyword
define variables, constants	define
select object or event	select
reject object or event	reject
define the mother object	take
define histograms	histo
applies object/event weights	weight
bins events in regions	bin

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

ADL syntax rules: https://twiki.cern.ch/twiki/bin/view/LHCPhysics/ADL

ADL syntax: functions

Standard/internal functions: Sufficiently generic math and HEP operations would 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: sort, comb(), union()…

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.

- Variables with non-trivial algorithms: M_{T2}, aplanarity, razor variables, …
- Non-analytic variables: Object/trigger efficiencies, variables/efficiencies computed with ML, …

ADL syntax rules: https://twiki.cern.ch/twiki/bin/view/LHCPhysics/ADL

LHADA/CutLang —> ADL: syntax

Color legend:

new object,
variable, region
existing object
object attribute
existing variable
existing region
internal function
external function

Objects

```
# AK8 jets
object AK8jets
  take FatJet
  select pt > 200
  select abs(eta) < 2.4

# mass-tagged jets
object WjetsMassTag
  take AK8jets
  select msoftdrop [] 65 105

# W-tagged jets
object Wjets
  take WjetsMassTag
  select tau2 / tau1 <= 0.4
```

Variable definitions

```
define MR = fMR(megajets)
define Rsq = sqrt(fMTR(megajets, met) / MR)
define METl = MET + leptonsVeto[0]
define Rsql = sqrt(fMTR(megajets, METl) / MR)
define MT = fMT(leptonsVeto[0], MET)
define Mll = fMll(leptonsTight[0], leptonsTight[1])
define dphimegajets = dPhi(megajets[0], megajets[1])
```

Event selection regions

```
# preselection region
region preselection
  select size(AK4jets) >= 3
  select size(AK8jets) >= 1
  select MR > 800
  select Rsq > 0.08
```

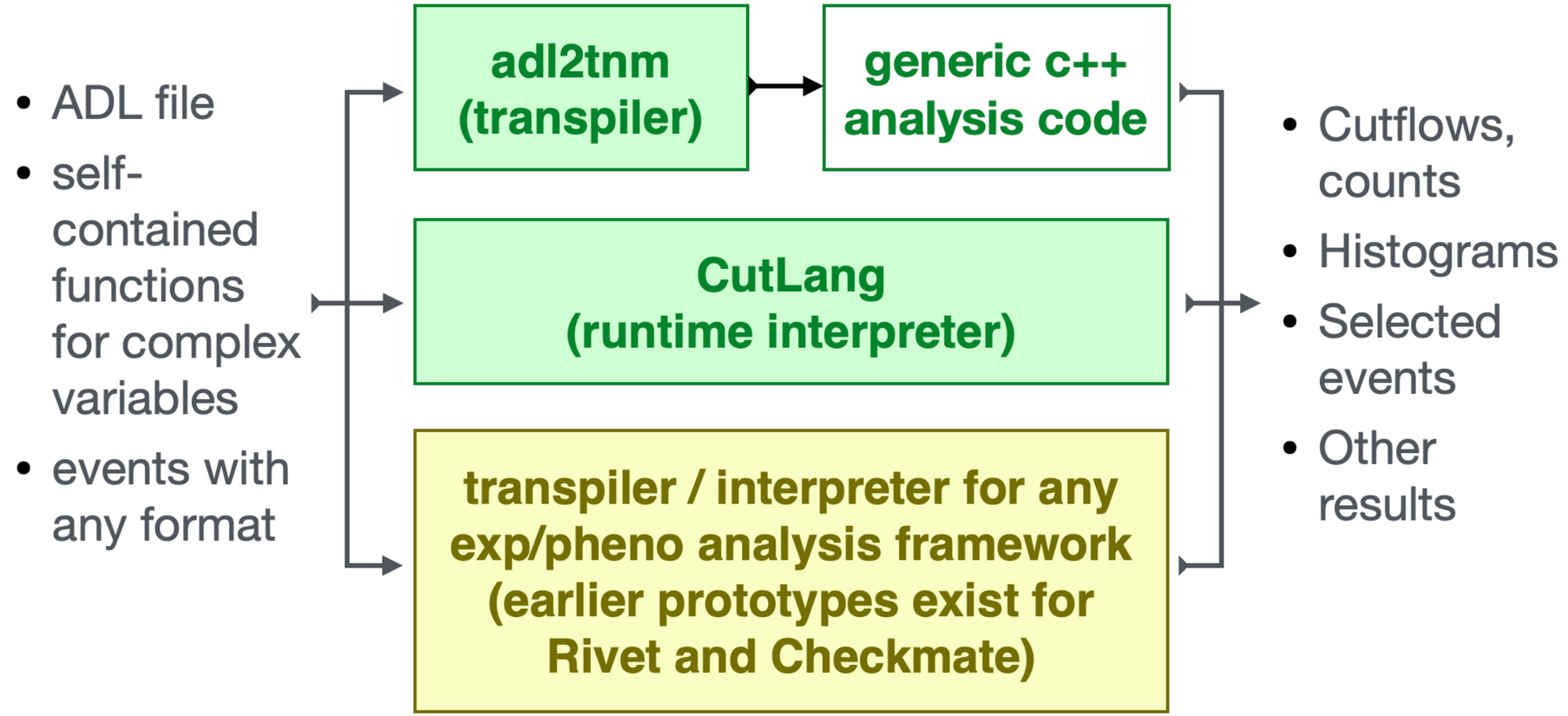
```
# control region for tt+jets
region ttjetsCR
  select preselection
  select size(leptonsVeto) == 0
  select size(Wjets) >= 1
  select dphimegajets < 2.8
  select MT > 100
  select size(bjetsLoose) >= 1
  bin MR [] 800 1000 and Rsq [] 0.08 0.1
  bin MR [] 800 1000 and Rsq [] 0.1 0.2
  bin ...
```

From CMS SUSY razor analysis CMS-SUS-16-017
Phys.Rev.D97(2018) no.1, 012007, arxiv:1710.11188

Full implementation link

Running analyses with ADL

ADL itself is analysis framework-independent. It can be interfaced with any framework capable of parsing ADL syntax.



Running analyses with ADL: CutLang

G. Unel, B. Örgen, A. Paul, N. Ravel, S. Sekmen, J. Setpal, A. M. Toon

CutLang runtime interpreter:

- No compilation. Directly runs on the ADL file.
- Written in C++, works in any modern Unix environment.
- Based on ROOT classes for Lorentz vector operations and histograms.
- ADL parsing by Lex & Yacc: relies on automatically generated dictionaries and grammar.

CutLang framework: CutLang interpreter + tools and facilities

- Reads events from ROOT files, from multiple input formats like Delphes, ATLAS / CMS Open Data ntuples, LVL0, CMSnanoAOD, FCC. More can be easily added.
- All event types converted into predefined particle object types.
- Includes many internal functions.
- Output in ROOT files. Analysis algorithms, cutflows, variable and object definitions, histograms for each region in a separate TDirectory.

CutLang Github: https://github.com/unelg/CutLang

CutLang publications: arXiv:1801.05727, (arXiv:1909.10621)