Improving CutLang with New Features

Arpon Paul
Summer Student, CERN, Geneva, Switzerland

Abstract
This document explains the new features added to the analysis description language CutLang as a part of the Summer student project. This summer student project is supervised by Gökhan Ünel, Sezen Sekmen and David Francis.

1 Introduction
This project is about an Analysis Description Language (ADL), a domain specific computer language that aims to provide a clear, human readable way to define collider data analyses in high energy particle physics (HEP) along with an interpretation framework of that language. The ADLs are becoming popular as proven by a dedicated workshop organized Fermilab (https://indico.cern.ch/event/769263/).

On one hand, LHADA (Les Houches Analysis Description Accord proposal), was designed by a group of LHC experimentalists and phenomenologists as a text-based language that can express analysis components for a wide range of physics analyses. The analyses are expressed as human readable text files including simple variable definitions, object selections, and event selections, supplemented by external functions for calculation of more complex analysis quantities. Several prototype interfaces are under development for analysis frameworks such as TheNtupleMaker, CheckMate and Rivet. On the other hand, a proof of principle (PoP) implementation of the interpreted ADL idea, CutLang, achieved using C++ as a layer over the CERN data analysis framework ROOT, is presently available. The ADL in CutLang is more than 90% similar with LHADA. In CutLang, HEP analyses can be written in a simple manner, as a set of commands in human readable text files, which are interpreted by the framework at runtime. The initial experience with CutLang has shown that a just-in-time interpretation of a human readable HEP specific language is a practical and efficient alternative to analysis writing using compiled languages such as C++. The interpreted approach is seen to speed up analysis design especially for beginners and phenomenologists. Moreover, as it is a declarative language, the user does not need to have coding skills.

In addition, CutLang’s ability to process events from multiple data sources such as Delphes, ATLAS and CMS Open data could prove useful also to the CERN OpenData project. In this project, some new features and corresponding example analysis have been added to CutLang.

2 NOT Command for Selection Criteria
The use of Boolean operators (AND, OR, NOT) can make it easy to write the event-selection criteria. In CutLang, logical AND and logical OR operator have already been used to combine multiple event-selection criteria. The logical NOT is newly implemented. This simplifies the way to write the criteria of event-selections in the analysis code to a great extent. The simplest example code to understand the syntax:

```select NOT Size(ELE) > 4```

This command selects events which do NOT have number of electrons greater than 4. We can also combine event selection criteria using the logical AND, OR, NOT. For example:
Now let us look at another code:

```adl
select Size(ELE) == 2
select NOT ( {ELE[0] ELE[1]}q == 0 AND {ELE[0] ELE[1]}m [] 80 100)
```

( {ELE[0] ELE[1]}q == 0 AND {ELE[0] ELE[1]}m [] 80 100) - this criteria can be used for the analysis related to Z bosons. As we have set NOT, we veto events with Z boson while looking for other dilepton signatures. Without using the NOT command, this analysis would not be so straightforward.

### 3 REJECT Command

In simple words, REJECT $\equiv$ SELECT NOT

i.e, REJECT command rejects the events that match the given criteria. The last example of the previous section can also be written using the REJECT command as below:

```adl
REJECT {ELE[0] ELE[1]}q == 0 AND {ELE[0] ELE[1]}m [] 80 100
```

### 4 PDGID of Particles

Each type of particle in particle physics is assigned a unique code by the Particle Data Group (PDG). These codes are known as PDGID (or PDG ID). The numbering includes elementary particles such as, electrons, neutrinos, Z bosons etc, composite particles (mesons, baryons etc) and atomic nuclei. Hypothetical particles beyond the Standard Model also have PDGID. Particles have a positive PDGID whereas antiparticles a negative one. Here is the list of PDGID of some particles:

<table>
<thead>
<tr>
<th>Quarks</th>
<th>Leptons</th>
<th>Bosons</th>
</tr>
</thead>
<tbody>
<tr>
<td>d  1</td>
<td>$e^-$</td>
<td>$\gamma$ 22</td>
</tr>
<tr>
<td>u  2</td>
<td>$\mu^-$</td>
<td>Z  23</td>
</tr>
<tr>
<td>s  3</td>
<td>$\tau^-$</td>
<td>$W^+$ 24</td>
</tr>
</tbody>
</table>

In CutLang, particles can be selected using their PDGID. For example:

```adl
select PDGID( LEP[0]) == -11
```

This command selects positrons. (Positron is the antiparticle of electron, so it has negative PDGID)

Internally, a new attribute p_pdgid has been added to the dbx_particle class for the PDGID

### 5 WEIGHT Command

In many analysis of high energy particle physics, events are needed to be weighted. For example, the overall trigger efficiency in an analysis can be 0.95 instead of 1. In this case, we can weight the events using the following ADL code:

```adl
select ALL
weight trigEff 0.95
```

This command sets the weight of the selected events to 0.95, i.e, if the number of selected events is 1000 in the beginning, now it will be counted as 950.
6 SORT Command

By default, the particles are sorted according to their transverse momentum, $p_t$. For example, ELE[0] denotes the electron having the highest transverse momentum. For some particular analysis, particles may be needed to be sorted according to some other properties, such as, energy, pseudorapidity etc. In the latest version, this can be done in the following way:

```
Sort {ELE_}E descend
```

This command sorts electrons according to their energy in descending order, i.e, ELE[0] will have the least energy.

Sorting has been implemented using two algorithms: [5], [6]

6.1 Selection Sort

Selection sort is the simplest sorting algorithm. In this algorithm, it selects the smallest element from the unsorted part of the list and put it in the beginning of the unsorted list in each iteration. Though the idea is simple, the computational complexity for selection sort is $O(n^2)$. So, it is very inefficient for large lists. This algorithm was used in CutLang for testing purposes and to make a comparison with the Quick Sort algorithm. In the final version, this algorithm has been deactivated.

6.2 Quick Sort

Quick Sort is a Divide and Conquer algorithm. It selects an item as pivot and partitions the given list around the picked pivot. Now the list is reordered so that all elements with values less than the pivot are placed before the pivot, whereas all elements with values greater than the pivot are placed after it. And then this algorithm is iterated recursively on those partitions. This is one of most efficient sorting algorithms. The computational complexity for this algorithm is $O(n \log n)$. In the final version, this algorithm has been implemented for sorting. Computation time is compared for the two algorithms, and for 50000 events, no significant difference is observed.

A new C++ class sortnode has been introduced to implement sorting in CutLang.

7 Conclusion

After the inclusion of these new features in CutLang, the analysis description language has become more powerful. Some example ADL code have also been added to the new version so that users can understand the syntax and its uses. Still, there are lots of commands yet to be implemented, such as, plotting 2D Histogram, saving selected events in a ROOT file, and most importantly, a generic way for building composite objects. Apart from technical developments, I have also recently started a phenomenological analysis of heavy BSM quarks which will be performed completely using CutLang.

Acknowledgement

I would like to express my deepest appreciation to my supervisors, Gökhan Ünel and Sezen Sekmen whose contribution in stimulating suggestions and encouragement, helped me to complete this project.

References


https://github.com/unelg/CutLang/
[3] Monte Carlo Particle Numbering Scheme

[4] Particle Wiki: PDG particle numbering scheme
https://particle.wiki/wiki/PDG_particle_numbering_scheme


https://en.wikipedia.org/wiki/Quicksort