CutLang: a text-based particle physics analysis description language and run time interpreter

S. Sekmen[1] and G. Unel[2] —December 7, 2017

[1]Kyungpook National University, Physics Dept., Daegu, South Korea
[2]University of California at Irvine, Department of Physics and Astronomy, Irvine, USA

The *CutLang* User Manual

(V2 / November 2017 - For users with some experience in ROOT)

• Pre-requisites to run *CutLang*:

Any Unix, Linux or OSX machine is fit to run *CutLang*. Basic knowledge of terminal operations such as editing a text file and moving files around is also needed in addition to running the analysis executable. Basic knowledge on ROOT macros would be good for manipulating histograms if preferred, but is not essential. Typically a user will have to find and work with at least these files:

- The text file `CLA.ini` (this is the example analysis configuration file, although it can also be specified at the command line)
- The shell script file `CLA.sh` (this is the main executable)
- The ROOT file to be analyzed (this is the "events" file) (A ttbar example can be downloaded from http://cutlang.hepforge.org for a quick start). Multiple comma separated input files and their paths can be specified at the command line; e.g. `./CLA.sh ../roots/atlas1ttbar.root,../roots/atlas2ttbar.root ATLASOD`

• Analysis scope in *CutLang*:

Only electron and muon triggers are implemented in this version. Wherever the term 'lepton' or the abbreviation "LEP" is used, they refer to either electron or muon depending on the selected trigger. Tau channel is not available in this version.

• Obtaining and compiling *CutLang* analysis binary:

The latest version of the *CutLang* Analysis package can be downloaded from HepForge at

`http://cutlang.hepforge.org`

The downloaded file should be opened with

`tar -xzf CLA_V200.tgz}`

This will automatically create the `CutLang` directory. To compile,

```
cd CutLang/CLA
make
```

The runs should be executed from the `CutLang/runs` subdirectory.

• How to run an example *CutLang* analysis:

This section provides the 8 steps to follow for running an analysis in *CutLang*.

- Step0- Open a Terminal
- Step1- Go to Directory '*CutLang*/runs'
- Step2- Edit the analysis description file, e.g. `CLA.ini`
- Step3- Go back to the same Terminal
- Step4- Execute the analysis description edited in Step2 using the following command:

```
./CLA.sh cms-opendata-ttbar.root CMSOD [-i youranalysis.ini]
         [-e number_of_events_to_process] [-v verbosity]
```

- Step5- *CutLang* lists the analysis evaluation results on screen.
  - If everything is correctly entered in the analysis description file in Step2, the user will see the following message on the screen: "`End of analysis initialization`". All is well, proceed to Step6.
  - If there are any errors in the analysis description, *CutLang* notifies the user about the unknown parameter(s) as "`UFO`"(s). Go back to Step2, verify and correct the '`ini`' file.
- Step6- `CLA` lists every 1000 processed event messages until it reaches the end of the ROOT file.
- Step7- Efficiency table for the analysis is displayed on the screen.
- Step8- `CLA` displays the message: "`saving...finished`" at the end of the analysis. Output file is saved under the same directory. Its name is given by the system as `histoOut-NAME.root` ; where NAME is the name of the analysis description ini file. If no name is specified, the default value of `histoOut-CLA.root` is used. If users wish to keep output from previous analysis, the output file is to be renamed. Otherwise, the `CLA.sh` will overwrite the output file.

• How to see the *CutLang* output files:

There are two ways to see the contents of the *CutLang* output file:

1. Open it using ROOT : `root.exe histoOut-CLA.root;` launch a `TBrowser`
2. Run the default macro: `./showall.sh`

• File and command details:

`CLA.ini` default input file used in Step2 is the analysis description file that provides the analysis algorithm. It is available by default for all *CutLang* users and it is editable. Based on this one, similar input files can be prepared and used.

`CLA.sh` The script that executes *CutLang* analysis as shown in Step4. The first **two** arguments are mandatory: the first is the name of the ROOT data file to be analyzed, e.g. `cms-opendata-ttbar.root`. The analyst has to get it correctly from the relevant source. The second argument is the input data format. There are several readable ROOT file formats. The user needs to specify one among the following: ATLASOD (ATLAS open data), LVL0, FCC, LHCO, Delphes or CMSOD (CMS open data), or add a new format as described in the last section of this document. The other arguments are optional and can be listed as:

`-e|--events` the number of events to be processed. By default all events (represented by 0) are processed.

`-i|--inifile` the initialization file to be processed. By default `CLA.ini` is processed.

`-v|--verbose` the verbosity frequency. By default after each 1000 events the current event number is written on screen.

`-h|--help` displays these explanations as reminders.

`histoOut-NAME.root` output file produced in Step8 where NAME is the name of the analysis description ini file. If no name is specified, the default value of `histoOut-CLA.root` is used. In case of multiple signal regions, each region will have its own directory inside the output file marked with `BP_i` where `i` is index number.

• How to display/edit the analysis description file '`CLA.ini`' (Step2):

The initialization file is made of three sections:

- **Objets thresholds:** This section is mandatory. It contains the η and p_T threshold values for a particle to be accepted.
- **User definitions:** This is a non-mandatory section containing user definitions starting with keyword '`def`' for new composite particles and variables. These definitions are practical as they create shorthand names for otherwise long expressions for later use.
- **Event selection algorithm:** This section is mandatory. It consists of lines starting with keyword '`cmd`' which define event operation or selection criteria. It also consists of lines starting with '`histo`' which signify the histogram definitions.

Common rules for all sections

- All lines start with one of the `def, cmd`, or `histo` keywords. No space should exist before the keywords. Note that there are no keywords for the object thresholds section.
- Indefinite amount of space is allowed between the keyword and the command/description.
- Every command/description must be enclosed within double quotation marks, and there should be a space before the ending quotation mark, e.g. "`mLL : { LEP_1 LEP_0 }m", or "mLL [] 70 120 "`.
- There is no upper limit for the number of lines.
- At least one space must be left before and after each term; including operands, numbers.
- All units in *CutLang* are either GeV or radians (c=1, therefore mass, momentum, energy are all in GeV)
- All variable, function and particle names are case sensitive.

Additional rules in editing the ini file

R1 : " ~ = " and " ! = " cannot be combined with any other operator or function. For example:

" `nJET` >= 6 AND `nBJET` >= 2 " ***OK***

" `mTopb` ~= 175 " ***OK***

" `mTopb` ~= 175 AND `nBJET` >= 2 " ***not OK***

" `mTopb` ~= 175 AND `mTopb2` ~= 175 " ***not OK***

R2: Any expression after a "#" is considered as a comment. If the user likes to temporarily skip a line in the program, it is sufficient to add a "#" in the beginning of the line.

R3: Names of the user defined composite particles and variables are unique. They cannot be redefined within the same analysis.

R4: "{ }" are used to mention the properties of particles, e.g. mass, charges etc. One can add as many particles as required in the term within the curly braces.

R5: The order of particles in a particle combination does not have an impact on the outcome; i.e. : `LEP_0 LEP_1` is not different than `LEP_1 LEP_0`

An example ini file with multiple regions

```
###### OBJECT THRESHOLDS
minpte = 15.0 # min pt of electrons
minptm = 15.0 # min pt of muons
minptj = 15.0 # min pt of jets
maxetae = 2.47 # max pseudorapidity of electrons
maxetam = 2.5 # max pseudorapidity of muons
maxetaj = 5.5 # max pseudorapidity of jets
TRGm = 0 # muon Trigger Type: 0=dont trigger, 1=1st trigger (data) 2=2nd trigger (MC)
TRGe = 2 # electron Trigger Type: 0=dont trigger, 1=1st trigger (data) 2=2nd trigger (MC)

###### USER DEFINITIONS
def "WH1 : JET_-1 JET_-1 " # W boson of the first top
def "WH2 : JET_-11 JET_-11 " # W boson of the second top
def "mWH1 : { WH1 }m " # mass of W boson of the first top
def "mWH2 : { WH2 }m " # mass of @ boson of the second top
def "mTopH1 : { WH1 JET_-2 }m " # first top quark's mass
def "mTopH2 : { WH2 JET_-4 }m " # second top quark's mass

###### EVENT SELECTION
algo __preselection__
cmd "ALL " # to count all events
cmd "nJET >= 6 " # events with 6 or more jets
cmd "MET < 100 " # fully hadronic events should have small MET
#cmd "FillHistos "
#histo "Basics "

algo __teknik1__
__preselection__
cmd "mTopH1 - mTopH2 / 4.2 ^ 2 + mWH1 - 80.4 / 2.1 ^ 2 + mWH2 - 80.4 / 2.1 ^ 2 ~= 0 " # 2 topHads
cmd "FillHistos "
histo "mWHh1 , Hadronic W reco (GeV), 50, 50, 150, mWH1 "
histo "mWHh2 , Hadronic W reco (GeV), 50, 50, 150, mWH2 "
histo "mTopHha1 , Hadronic top reco (GeV), 70, 0, 700, mTopH1 "
histo "mTopHhb1 , Hadronic top reco (GeV), 70, 0, 700, mTopH2 "

algo __teknik2__
__preselection__
cmd "mWH1 - 80.4 / 2.1 ^ 2 + mWH2 - 80.4 / 2.1 ^ 2 ~= 0 " # 2 WHads
cmd "mTopH1 - mTopH2 / 4.2 ^ 2 ~= 0 "
cmd "FillHistos "
histo "mWHh1 , Hadronic W reco (GeV), 50, 50, 150, mWH1 "
histo "mWHh2 , Hadronic W reco (GeV), 50, 50, 150, mWH2 "
histo "mTopHha1 , Hadronic top reco (GeV), 70, 0, 700, mTopH1 "
histo "mTopHhb1 , Hadronic top reco (GeV), 70, 0, 700, mTopH2 "
```

• How to add a new file format to *CutLang* Analysis:

- An example root ntuple file for the new file format has to be acquired and loaded into ROOT.
- The MakeClass command from ROOT should be called on the new ntuple with a new class name, e.g. `newNT->MakeClass("XFormat");`
- The resulting header file (`XFormat.h`)has to reside in the `analysis_core` subdirectory, and it has to be included by the main code `CLA.C`
- The implementation macro (`XFormat.C` file) has to be in the `CutLang/CLA` directory, and has to include these needed headers:

  ```
  #include "lhco.h"
  ```

```
#include <TH2.h>}
#include <TStyle.h>}
#include <TCanvas.h>}
#include <signal.h>}
#include "dbx_electron.h"
#include "dbx_muon.h"
#include "dbx_jet.h"
#include "dbx_a.h"
#include "DBXNtuple.h"
#include "analysis_core.h"
#include "AnalysisController.h"
```

- In the event loop, it has to fill the predefined electron, muon, photon and jet particles vectors, without forgetting any available event-wide information like RunNumber, EventNumber etc... An example for the LHCO format is:

```
TLorentzVector alv; dbxMuon *adbxm; vector<dbxMuon> muons;
for (unsigned int i=0; i<Muon_; i++) {
  alv.SetPtEtaPhiM( Muon_PT[i], Muon_Eta[i], Muon_Phi[i], (105.658/1E3) ); // all in GeV
  adbxm= new dbxMuon(alv);
  adbxm->setCharge(Muon_Charge[i] );
  adbxm->setEtCone(Muon_ETiso[i] );
  adbxm->setPtCone(Muon_PTiso[i] );
  adbxm->setParticleIndx(i);
  muons.push_back(*adbxm);
  delete adbxm;
  }
```

- The last steps in this file should be as follows:

```
AnalysisObjects a0={muons, electrons, photons, jets, met, anevt};
aCtrl.RunTasks(a0);
} // end of event loop
aCtrl.Finalize();
} // end of Loop function
```