LHADA-to-Rivet translator

Philippe Gras

CEA/IRFU - Saclay

Nov 15, 16

Introduction

- Support for BSM analyses added in Rivet with version 2.5.2
  - Reproducing the analyses, with the smearing implemented in Rivet. No need for Delphes.
- Implemented a prototype from lhda to rivet in the context of a Rivet developer workshop
  - Study the feasibility of an automatic translation from LHADA to code

Prototype implementation

- Implementation done using python: suitable for fast developement and for text processing.
- Takes CMS MT2 LHCP16 analysis, CMS-SUS-15-003 as use case,

"Search for new physics with the M_{T2} variable in all-jets final state produced in pp collisions at $\sqrt{s} = 13\,\text{TeV}$"

- Information for reinterpretation, as for other SUSY analysis, is public and available from CMS Public Result pages.

- Usage:

`lhada2rivet.py CMS-PAS-SUS-16-015.lhada CMS-PAS-SUS-16-015`
(the second argument is used for the analysis label used by Rivet)

- Extra input to build the Rivet module: c++ code of the user functions

Input

```
function btag_eff
       #Returns the b-tagging efficiency of a b jet
       arg bjet #b jet to apply efficiency to
       code CMS-PAS-SUS-015-functions.cc

function mt2
       #Computes the M_{T2} observable, http://arXiv.org/abs/hep-ph/9906349v1
       arg particle1 #quadrivector of first visible particle (only m, px, py are used)
       arg particle2 #quadrivector of first visible particle (only m, px, py are used)
       arg met       #quadrivector of missing energy (only px, py are used)
       code CMS-PAS-SUS-015-functions.cc

object jets_eta47
       take external
       apply antikt(dR=0.4, ptmin=30, etamax=4.7)

object jets
       take jets_eta47
       select eta < 2.4.

object bjets
       take external
       apply antikt_b(dR=0.4, ptmin=30, etamax=2.5)
       weight btag_eff

object met
       take external

object ht
       take jets
       apply scalar_pt_sum

cut mt2_cut
    select (jets.size < 2) or (mt2(particle1 = jets[1], particle2 = jets[2], met = met) > 200)

cut deltaphi_etmiss_jet
    select (dphi(met, jets[1]) > 0.3) and (dphi(met, jets[2]) > 0.3) and(dphi(met, jets[3]) > 0.3) and(dphi(met, jets[4])
=> 0.3)


cut preselection
#   select trigger
    select mt2_cut
    select deltaphi_etmiss_jet
#   select etmiss per bad fac
```

Output

```
// -*- C++ -*-
#include "Rivet/Analysis.hh"

#include ""
#include "Rivet/Projections/FastJets.hh"
#include "Rivet/Projections/FastJets.hh"

namespace Rivet {


  class RivetAnalysis : public Analysis {
  public:

    /// Constructor
    RivetAnalysis()
      : Analysis("RivetAnalysis")
    {  }


    /// Book histograms and initialise projections before the run
    void init() {

      FinalState fs;
      VisibleFinalState visfs(fs);

      addProjection(FastJets(fs, FastJets::ANTIKT, 0.4), "jets_eta47");
      addProjection(FastJets(fs, FastJets::ANTIKT, 0.4), "bjets");

    }

    bool cut_mt2_cut(){
        bool r = true;
        r &= (jets.size() < 2)  || (mt2(jets[1 - 1], jets[2 - 1], met) > 200);
        return r;
    }
    bool cut_deltaphi_etmiss_jet(){
        bool r = true;
        r &= (dphi(met, jets[1 - 1]) > 0.3)  && (dphi(met, jets[2 - 1]) > 0.3)  && (dphi(met, jets[3 - 1]) > 0.3)  &&
(dphi(met, jets[4 - 1]) > 0.3);
        return r;
    }
    bool cut_preselection(){
```

Note: a function is created for each cut block

Prototype status

- Implemented blocks: function, object, cut
- Blocks still to implement: **table**, (info), units
- Takes an lhada file and produce a .cc file with the Rivet code.

Complement the format specification

Needs for more rules

- Implementing an automatic translator requires extra assumptions not specified in Les Houces proceedings
  - $\rightarrow$ Define a common set of assumptions
- The tool to validate LHADA discussed this morning will be very useful.
- A LHADA format version number should be included in the LHADA file to handle evolutions

Several points addressed in the following.

LHADA inputs: `take external`

How to deal with "take external"?

- For tools like Rivet which do not use detector simulation (`Delphes`), final state particles and possibly vertices are sufficient. More object can be added for convenience, those can be defined using LHADA format.
- Tools using `Delphes` starts from higher level object
  - Need for two entries: start from generator-level quantities and start from reconstucted quantities.

More on objects

Objet attributes

- `object/LHAParticle`
  - define a set of attributes. E.g: pt, eta, y, px, py, pz, m, pdgid, charge, phi, theta

Single observable

How to defined a single observable, like H_T

- Use object block that returns a collection with a single element
  - $\rightarrow$ difficult for the translator to guess it's a single observable
  - $\rightarrow$ can still be guessed from the object usage (absence of index)
- Make it explicit with an attribute in the object block or with a different block name?

Function

Programming language

- Obviously a translator understand all existing programming language
- Functions to be provided in the programming language of the target tool (Checkmate, MadAnalysis, Rivet,..)
  - Nice thing: the current tools all uses the same language.

Implicit arguments

In the `apply` statement, the object collection (union of previous $\widehat{\text{take}}$'s) is an implicit argument of function

- A function can have several arguments and not requires this collections: e.g. building a candidate from two different collections.
  - The translator can guess if it should pass the collection and to which argument..
  - ...but it's error prone
- $\rightarrow$ Define a name for THE collection and use explicit argument for multiple argument functions

More on specifications: cuts

Cut line

- cut Define a set of allowed operators and the precedence.
  - E.g.: $+, -, /, *, **$,'and,or, $<, >, <=, >=$ + few common mathematical functions.

Miscellaneous

- Specifying the order of blocks will make translation easier
  - `function, object, cut, table`
- One swiss-knife tool or one per recasting tool?
- Weights: current Rivet approach for selection efficiency is random rejection. LHADA approach is to weight events. Support both?
- Add a field for the LHADA format version
- Detector effect (efficiency/smearing) is currently mixed with the analysis workflow.
  - Separate the two can make the description more legible
  - Not the time to change the format,

Lessons from the exercise and next steps

- Conclusion of the study:
  - Automatic translation is feasible and even less chalenging than I was expecting first.
- Next steps:
  - Complete the prototype to produce a working rivet analysis.
  - Compare codes with the checkMate translator