CutLang: A Particle Physics Analysis Description Language and Runtime Interpreter

Gökhan Ünel a, Sezen Sekmen b, *

a University of California at Irvine, Department of Physics and Astronomy, Irvine, USA
b Kyungpook National University, Daegu, South Korea

A B S T R A C T

This note introduces CutLang, a domain specific language that aims to provide a clear, human readable way to define analyses in high energy particle physics (HEP) along with an interpretation framework of that language. A proof of principle (PoP) implementation of the CutLang interpreter, achieved using C++, as a layer over the CERN data analysis framework ROOT, is presently available. This PoP implementation permits writing HEP analyses in an unobfuscated manner, as a set of commands in human readable text files, which are interpreted by the framework at runtime. We describe the main features of CutLang and illustrate its usage with two analysis examples. Initial experience with CutLang has shown that a just-in-time interpretation of a human readable HEP specific language is a practical alternative to analysis writing using compiled languages such as C++.

Program summary
Program title: CutLang
Program Files doi: http://dx.doi.org/10.17632/pym39s7vy7.1
Licensing provisions: GNU General Public License 3 (GPL)
Programming language: C++
Nature of problem: Obtaining a physics result from high energy collider (e.g. LHC) data typically involves complex software frameworks for data analysis. Operating such analysis frameworks usually demands high computational expertise from the physicists, which makes trying simple analysis ideas difficult. The difficulty can be overcome by decoupling the description of all physics components of an analysis from the software framework. Analyses would thus be fully and unambiguously described by a framework-independent domain specific language. Defining such a language, which would be text-based and human readable is the first problem this paper investigates. Once a text-based language is available, a major task is to write an interpreter (or compiler) that converts it into instructions that can actually be executed, perhaps in some framework. The second problem addressed here is to write an automated, just-in-time interpreter to perform this task.
Solution method: CutLang addresses the above problems by first constructing a domain specific language which aims to provide a clear, human readable way to define high energy physics analyses. The language is capable of defining various components of the analysis like object selection, event selection, and allows analysis tools like variable definition, $\chi^2$ minimization, multiple selection regions definition and many others. CutLang also provides an automated interpretation framework for the language, which is achieved using C++ classes as a layer over the CERN ROOT framework. The interpreter evaluates each component of the analysis description at runtime, without compilation. It handles event selection by classes which consist of a function that evaluates event-by-event object or event related quantities, a comparison operator, and one or two limit values. For each evaluation, the function result is compared to the limit values to find a boolean result of 0 or 1. Multiple selection criteria are combined together using logical operators. CutLang evaluates such expressions numerically by first converting them into Reverse Polish Notation, and then by parsing them with the Shunting Yard algorithm. Several properties of the interpreter are being able to work with multiple particle types, allowing particle combinations to make new particles, offering standard functions, histogramming, etc.

* This paper and its associated computer programme are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail addresses: gokhan.unel@cern.ch (G. Ünel), sssekmen@cern.ch (S. Sekmen).

https://doi.org/10.1016/j.cpc.2018.06.023
0010-4655/© 2018 Elsevier B.V. All rights reserved.
Additional comments including restrictions and unusual features: The CutLang language and interpreter can work for a large number of physics analyses at the LHC. However it would benefit from some generalizations which would make it eligible for a larger variety of analysis. These generalizations could be achieved by inclusion of external functions for computing analysis-related variables, and allowing user-defined particle collections.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

Since the era of the LEP experiments, particle physicists have been performing large scale computing tasks to analyze particle physics data in order to obtain physics results. A typical analysis requires extensive manipulation of real and simulated particle collision events, principally defining analysis objects, defining quantities that help classify events as signal or background, selecting events, reweighting simulated events to improve the agreement between the simulated and real events, and interpreting experimental results by comparing them to predictions. Performing all of these tasks in a systematic manner generally requires an analysis framework that organizes and sequences these tasks appropriately.

As a consequence, a physicist who wishes to engage in a particle physics data analysis needs to be well acquainted with computing at the level of both a system programmer and a software developer. System level expertise is needed because the frameworks generally comprise multiple software components, while software level expertise is needed in order to write these components. The coding needs for analysis tasks include compiled languages like C, C++ [1], and interpreted languages like Python [2,3], awk [4] and bash [5]. This list, combined with the structural complexity and diversity in analysis framework design makes data analysis a rather daunting task, and erects a barrier between data and the physicist who may simply wish to try out an analysis idea. The complexities and technical difficulties have grown considerably in the LHC era, both due to the unprecedented amount of data collected by the LHC experiments, and to the increasingly elaborate analyses inspired by these data.

In recent years, these challenges have motivated the exploration of ways to tame the complexity. For example, LHC experiments are converging towards fewer analysis frameworks that are developed and maintained by dedicated experts. On the phenomenology side, frameworks such as CheckMate [6-8] and MadAnalysis [9-12] exist, which implement a sizable number of LHC new physics search analyses ready for use in reinterpretation studies. Similarly, Rivet [13,14] hosts a large repository of LHC analyses implemented by the original analysts. Using centrally maintained frameworks to implement and run analyses already helps to automate and reduce technical burdens in analysis work. However, this approach is still severely biased towards physicists with considerable coding skills.

An alternative approach is to decouple the description of all physics components of an analysis from the software framework. Analyses would thus be fully and unambiguously described by a framework-independent domain specific language. Previous attempts at defining such languages (based on the LHCO format) were made in [15,16]. More recently, a thorough open discussion was started at the Les Houches PhysTeV 2015 workshop to define the elements and general structure of a language that could be broadly used by LHC physicists for describing analyses. An initial proposal, called the Les Houches Analysis Description Accord (LHADA) was released to the LHC community [17]. LHADA is a domain specific language with a strict set of syntax rules and a limited number of operators. A LHADA description of an analysis together with the associated self-contained functions, encapsulating non-trivial variables and, or, algorithms, provide a complete and unambiguous description of the analysis.

A framework-independent analysis description language has many advantages. First, it makes the writing of analyses significantly more accessible by eliminating coding complexities. Second, having a standard, framework independent language allows analysis preservation beyond the lifetimes of the experiments or analysis software, and facilitates the abstraction, visualization, validation, combination, reproduction, interpretation and overall communication of the contents of particle physics analyses. Third, such a language benefits not only the physicist working on an analysis within an experimental collaboration, but also colleagues from other experiments and phenomenologists. However, like other computer languages, an analysis description language is just that: a description. In order to execute a LHADA described analysis, an interpreter (or compiler) is needed to convert the description into instructions that can actually be executed, perhaps in a framework. The key point is that the mapping from the analysis description language to executing code must be automatic for the reasons discussed above.

In this paper, we introduce CutLang, which is both an analysis description language and the name of its interpreter, much the same way that Python is both a programming language and the name of its interpreter. The CutLang language follows the LHADA principles, but currently uses a syntax different from that of LHADA, as firstly, since work on CutLang had started earlier, and secondly, although a LHADA proposal exists, details of its syntax have not yet been fully finalized. The unique strength of the CutLang interpreter lies in its capability to perform interpretation at runtime, without the need for compilation. Runtime interpretation allows making rapid changes in an analysis, and hence is a very practical feature especially in the phase of analysis design. Therefore, this tool is intended primarily for analysis design by experimentalists and phenomenologists.

CutLang can be used in collider experiments such as ATLAS or CMS since it can easily be adapted to read in event data from a variety of different formats. Currently CutLang is being actively used in a full fledged ATLAS exotics analysis, and is capable of handling all object and event selection requirements. Another raison d’etre of CutLang is to provide a tool for the phenomenology community for studies on new analysis ideas, new kinematic variables, or tests of sensitivity to future experiments such as HL-LHC or the FCC CutLang. Furthermore it intends to serve physicists and physics enthusiasts using open data, who may not necessarily be expert programmers. With its easy to learn human readable syntax, the authors hope that CutLang would break the barrier between the collider data and the analysts. Such a tool would undoubtedly increase the number of analysts and democratize access to the data collected using public funds.

In the remainder of this note, we describe CutLang in detail and present several analyses examples written in CutLang syntax together with their runtime interpretation using the CutLang interpreter. Section 2 lays out the CutLang concept and design principles, followed by Section 3, which explains the implementation details. Section 4 illustrates the CutLang language with two analysis examples. Section 5 discusses the runtime execution speed of CutLang versus that of the corresponding C++ implementation. Our conclusions and the outlook are given in Section 6. A detailed user manual is given in Appendix A and example implementations of two published ATLAS analyses are presented in Appendix B.