Analysis Description Language for Particle Physics

H. B. Prosper[a], S. Sekmen[b], G. Unel[c]

[a] Florida State University, USA. [b] Kyungpook National University, Korea.
[c] University of California, Irvine, USA.

Since the era of the Large Electron Positron (LEP) collider experiments, physicists have been performing large-scale analyses of particle physics data. A typical analysis requires extensive manipulation of independent data ensembles, called events, of real and simulated particle collisions. This effort consists principally of defining analysis objects and quantities used in classifying events as signal or background, selecting events, re-weighting simulated events to improve their agreement with real events, and interpreting experimental results by comparing them to theoretical predictions.

Performing all of these tasks in a systematic manner generally requires an analysis framework that organizes the tasks into a computational pipeline. As a consequence, a physicist who wishes to engage in the analysis of particle physics data, needs to be acquainted with the framework both as a user and as a developer. Generally, the framework comprises multiple software components: while running such a setup requires system level expertise, software level expertise is also needed in order to write these components. Currently, the expertise required includes familiarity with compiled languages such as C++ and interpreted languages such as Python, awk and bash. This list, combined with the structural complexity and diversity of analysis frameworks, makes data analysis in particle physics a daunting task, and erects a barrier between data and the physicist who may simply wish to explore the data and try out an idea.

The complexity and technical difficulty of analyses have grown considerably in the era of the Large Hadron Collider (LHC). This is due, in part, to the unprecedented amount of data collected by the LHC experiments and, in part, to the increasingly elaborate analyses inspired by these data. Furthermore, it is anticipated that analyses will become even more elaborate as a consequence of the increasing creativity of the younger generation of physicists and of the requirements associated with the challenges of the forthcoming new colliders. In addition to the complexity of the analyses, there is also considerable complexity in the interface between the physicist and the analysis frameworks. It is that complexity that we plan to address in this contribution.

We propose a domain specific language (DSL), which we call the *analysis description language* (ADL), whose semantics reflect, but also inform, the conceptual reasoning of particle physicists [1]. ADL will allow a physicist to develop an analysis in a manner that is closer to how she or he thinks about what is being done in an analysis. By definition, a general purpose language (GPL) can do everything, a guarantee that is an enormous point in its favor. However, a language that can do everything may not be able to do so in a manner that is natural to

the domain specialist, hence our proposal to create a DSL. Many such languages exist taking advantage of the modes of thought, concepts, and algorithms within a given domain in order to simplify their expression. Given the success of this approach in other domains, we argue that this approach could be beneficial and even transformative in the domain of collider physics. A well-designed ADL would be empowering and would help the analysts clarify their thinking about analysis design and structure.

In this approach, the description of the components of an analysis is decoupled from the software frameworks through the use of an ADL that fully and unambiguously describes the whole algorithm. An analysis description language has many advantages. First, it makes the writing of analyses significantly more accessible by eliminating coding complexities, thereby allowing analysts with differing levels of computing skill to focus on the design aspect. This would be especially practical in the phase where the physics potential of various future collider options are being investigated through quick phenomenological studies. As the bulk of an analyzer's time is spent optimizing the signal significance, being able to quickly define new "derived objects" and new selection algorithms will dramatically reduce the time needed for optimization studies. Second, having a standard language allows easier communication of the analysis content, thus making its validation and reviewing easier. Third, such a language benefits not only the physicist working on an analysis within an experimental collaboration, but also colleagues from other experiments, or theoretical physicists interested in interpreting the results. Finally, the ability to describe an analysis in a framework-independent manner simplifies analysis preservation beyond the lifetime of software frameworks and even the experiments.

In order to evaluate these ideas, in particular the capability of an ADL to express analysis components and logic, a functioning ADL with detailed syntax rules has been under development since 2015 [1] and a large number of LHC analyses have been implemented in this language [2, 3, 4]. In an ADL document, written as plain-text, object, variable and event selection definitions are separated into blocks that follow a keyword-value structure, where keywords specify analysis concepts and operations. The syntax includes mathematical and logical operations, comparison and optimisation operators, reducers, four-vector algebra and common HEP-specific functions (e.g. $\Delta\Phi$, ΔR). ADL files can also refer to self-contained functions encapsulating variables defined through complex algorithms (e.g. MT2, aplanarity), complicated functional forms (e.g, machine learning classifiers) or non-analytic variables (e.g. efficiency tables).

To interpret this language, a C++ tool called CutLang has also been developed [5, 6]. In its current state, CutLang is capable of reading simple analysis algorithms written in ADL files and interpreting these at runtime, thereby eliminating the code-compile-execute cycle. The output from CutLang can be fed into statistical analysis tools. The CutLang interpreter prototype uses the standard lexical analysis and parsing tools, `lex` and `yacc`, which render the language flexible and robust against errors in the ADL file. CutLang can also handle standard tasks including histogramming, access to external user functions, event selection region overlap determination and input data type handling. Furthermore, a Python transpiler called adl2tnm that converts ADL to executable C++ analysis code is also under development [2].

The output of an analysis written in ADL is a list of surviving events, and summaries such as histograms, stored in a ROOT file, the common practice in high energy physics. This analysis output file will contain the analysis description in readable form and other meta-data, hence allowing provenance tracking of analyses. Allowing an output file to be generated at any point in the analysis permits interfacing with machine learning (ML) based selection tools. For

example, an ML model to discriminate between signal and background events could receive the relevant event information via such output files for training. It could return the likelihood of an event to fit a particular physics scenario or the likelihood of a set of particles to originate from a given mother particle. Any encapsulated ML model would be treated exactly the same as any other user defined function.

ADL and CutLang are already being used in experimental and phenomenological collider studies [7, 3, 4]. Additionally they have been interfaced with SModelS, an automatized tool enabling the fast interpretation of simplified model results from the LHC within various new physics models [8, 9, 10]. ADL and CutLang are used for running analyses on events produced for large numbers of signal points to build efficiency maps that are input to limit calculation in SModelS. They will also be used for determining analysis overlaps and identifying disjoint analyses or search regions to be used for combination [3]. The many potential uses in interpretation studies will be further explored via improving the interface with SModelS and with other interpretation tools.

In summary, ADL is a modern platform-independent way to describe HEP analyses in a human readable syntax. Together with its runtime interpreter CutLang and transpiler adl2tnm, the ADL approach will be further expanded and documented during Snowmass 2021. Additionally, a number of tutorials and advanced examples will be developed and documented. The gentle learning curve of analysis development in ADL should benefit other contributors to Snowmass 2021 working on phenomenological studies of the physics potential of future colliders.

References

[1] G. Brooijmans, *et. al.* "Les Houches 2015: Physics at TeV colliders - new physics working group report," [arXiv:1605.02684 [hep-ph]].

[2] G. Brooijmans, *et. al.* "Les Houches 2017: Physics at TeV Colliders New Physics Working Group Report," [arXiv:1803.10379 [hep-ph]].

[3] G. Brooijmans, *et. al.* "Les Houches 2019 Physics at TeV Colliders: New Physics Working Group Report," [arXiv:2002.12220 [hep-ph]].

[4] https://github.com/ADL4HEP/ADLLHCanalyses

[5] S. Sekmen and G. Unel, "CutLang: A Particle Physics Analysis Description Language and Runtime Interpreter," Comput. Phys. Commun. **233** (2018), 215-236 doi:10.1016/j.cpc.2018.06.023 [arXiv:1801.05727 [hep-ph]].

[6] G. Unel, S. Sekmen and A. M. Toon, "CutLang: a cut-based HEP analysis description language and runtime interpreter," J. Phys. Conf. Ser. **1525** (2020) no.1, 012025 doi:10.1088/1742-6596/1525/1/012025 [arXiv:1909.10621 [hep-ph]].

[7] A. Paul, S. Sekmen and G. Unel, "Down type iso-singlet quarks at the HL-LHC and FCC-hh," [arXiv:2006.10149 [hep-ph]].

[8] S. Kraml, S. Kulkarni, U. Laa, A. Lessa, W. Magerl, D. Proschofsky-Spindler and W. Waltenberger, "SModelS: a tool for interpreting simplified-model results from the LHC and its

application to supersymmetry," Eur. Phys. J. C **74** (2014), 2868 doi:10.1140/epjc/s10052-014-2868-5 [arXiv:1312.4175 [hep-ph]].

[9] F. Ambrogi, S. Kraml, S. Kulkarni, U. Laa, A. Lessa, V. Magerl, J. Sonneveld, M. Traub and W. Waltenberger, "SModelS v1.1 user manual: Improving simplified model constraints with efficiency maps," Comput. Phys. Commun. **227** (2018), 72-98 doi:10.1016/j.cpc.2018.02.007 [arXiv:1701.06586 [hep-ph]].

[10] F. Ambrogi, J. Dutta, J. Heisig, S. Kraml, S. Kulkarni, U. Laa, A. Lessa, P. Neuhuber, H. Reyes-González, W. Waltenberger and M. Wolf, "SModelS v1.2: long-lived particles, combination of signal regions, and other novelties," Comput. Phys. Commun. **251** (2020), 106848 doi:10.1016/j.cpc.2019.07.013 [arXiv:1811.10624 [hep-ph]].