From the Lagrangian to Feynman Rules with FeynRules

Neil D. Christensen
Michigan State University

Claude Duhr
Université catholique de Louvain

neil@pa.msu.edu
claude.duhr@uclouvain.be

Introduction

Physics models in high-energy physics are in general based on *quantum field theory*. In this mathematical language, the model is described by a *Lagrangian*, which contains all the particles in the model and describes their mutual interactions. Each particle is represented by a *field*, defined over the four-dimensional space-time. The interactions among the particles are then represented by non-linear terms coupling different kinds of fields at the same space–time point. The simplest example of such a model is Quantum Electrodynamics (QED), which is the quantum theory describing the electromagnetic interaction among the electron, the positron (the antiparticle of the electron) and the photon (the 'carrier' of the electromagnetic interaction). The Lagrangian of QED is

$$\mathcal{L}_{\text{QED}} = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu} + \bar{\psi}(i\gamma^\mu\partial_\mu - m)\psi - e\bar{\psi}\gamma^\mu\psi\, A_\mu, \tag{1}$$

where ψ, $\bar{\psi}$ and A_μ are the electron, positron and photon fields respectively, and $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu$ is the electromagnetic field strength tensor. The first two terms, which are quadratic in the fields, describe the noninteractive dynamics of the electron, positron and photon, while the third term couples, in a trilinear way, these three fields and hence describes their interaction (*e.g.* annihilation of an electron-positron pair into a photon). A graphical representation of this interaction is given in Fig. 1 (a) and is called a vertex. Perturbative predictions of a model can be obtained by connecting together in all possible ways the vertices and propagators (lines connecting vertices) of a model using a set of rules called 'Feynman rules', where the vertices and propagators are derived from the Lagrangian. An example of such a Feynman diagram contributing to the scattering of an electron-positron pair in QED is shown in Fig. 1 (b).

Predictions for collider experiments often require the computation of thousands if not tens of thousands of Feynman diagrams. Such a monumental task is especially suited to computers. There are several computer programs available that calculate Feynman diagrams automatically, both numerically and analytically. A few of these are CalcHEP/CompHEP, FeynArts/FormCalc, Herwig, MadGraph/MadEvent, Sherpa, and Omega/Whizard. These packages are responsible for generating the Feynman diagrams and making the predictions that will be compared with the results of experiment. However, only a very limited set of the possible physics models beyond the Standard Model is implemented in these tools, and hence their ability to determine which of all these models is correct is likewise limited. Implementing a new model into one of these codes requires to work out beforehand all the vertices that appear in the Lagrangian describing the model, and very often the input of one vertex at a time is necessary, making this a very tedious and error prone process for complicated models containing a large number of fields and interactions. On the other hand, deriving all the vertices and propagators from a Lagrangian is an algorithmic procedure very well suited to automation on a computer.

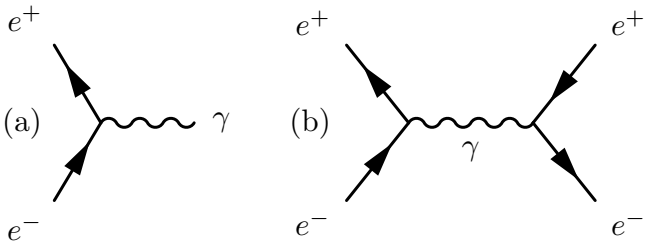


Fig. 1
(a) The vertex describing the interaction of the electron (e^-), positron (e^+) and photon (γ).
(b) A Feynman diagram contributing to electron-positron scattering.

FeynRules is a new package based on *Mathematica* which takes a model file with the Lagrangian as input and derives the Feynman rules associated with it [1]. The package contains a set of functions allowing the user to test and build their model one piece at a time. In addition to these built-in functions, the user is invited

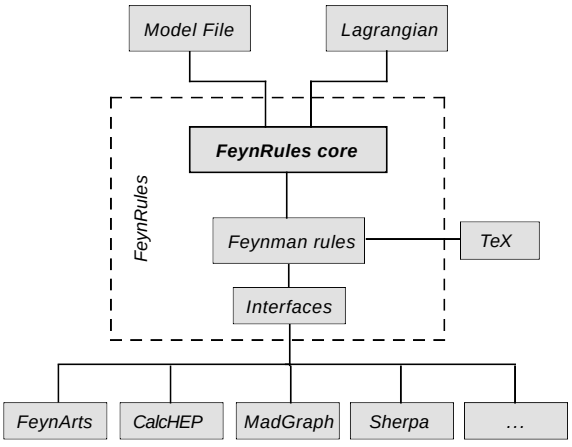
to exploit the full power of the underlying *Mathematica* code to extend these functions and to create his or her own new routines. In a second step, FeynRules can translate the model (with the derived vertices) into the model format corresponding to the Feynman diagram calculator chosen and allows in this way to implement the new model in any tool for which such a "translation interface" exists. In the following sections, we will briefly present the package, its features and the underlying algorithm.

The FeynRules model file

The basic input a user provides when implementing his or her model into FeynRules is the so called model file, a text file containing all the properties of the model (particles, parameters, *etc.*), and the Lagrangian written down using standard *Mathematica* language, augmented by some new symbols which are necessary when writing down a Lagrangian. The syntax and the structure of the model files, which is an extension of the original FeynArts model files, is based on the concept of classes. Each class consists of a physical quantity (*e.g.* a particle), together with the properties which define this quantity (*e.g.* the mass of the particle). There are three different kinds of classes which are used for a model implementation:

1. Particle classes: Each field appearing in the Lagrangian is assigned to a class, which specifies all the physical properties of the particle (*e.g.* spin, electric charge, mass, ...). In order to avoid a proliferation of particle classes, particles having similar physical properties (but possibly different masses) can be assigned to the same class.

2. Parameter classes: A physical model is also defined by its parameters. Each parameter is assigned to a class and the properties of the class of parameters are specified such as the numerical value (*e.g.* the numerical value of the electromagnetic coupling constant e in the QED Lagrangian (1)) or the functional dependence of the parameter on other parameters (*e.g.* the fine structure constant is defined in terms of the electromagnetic coupling as $\alpha = e^2/4\pi$) etc.



3. Gauge group classes: Gauge symmetries play an important role in limiting the allowed interactions in a field theory and make possible a compact way of writing the Lagrangian. The user can specify each gauge group and its properties in this class.

At run time FeynRules reads in and processes each class defined in the model file and then saves this information in various *Mathematica* lists which are used later when deriving the Feynman rules and translating the model information to the format appropriate for one of the Feynman diagram calculators.

The second input the user needs to provide is the Lagrangian, written down in *Mathematica* language in a form close to the textbook expression. We will illustrate this procedure on the QED Lagrangian defined in Eq. (1). Let us start with the first term in $\mathcal{L}_{\rm QED}$, the kinetic term for the photon field. In FeynRules, this term reads

```
-1/4 FS[A, mu, nu] FS[A, mu, nu]
```

The function `FS` appearing in this expression is predefined in FeynRules and returns the correct field strength tensor for the photon. The second term, the kinetic term for the electron field, is entered in FeynRules as

```
I psibar.Ga[mu].del[psi,mu] -
        m psibar.psi
```

where `psi` represents the electron field and `m` its mass (those symbols have been defined by the user in the model file), and `Ga[mu]` and `del[.,mu]` are special FeynRules commands representing the γ-matrices and the derivative ∂_μ. The dot-product appearing in this expression ensures that the anticommuting character of the electron field is correctly taken into account. Note that it is sufficient to define the electron field `psi` in the model file and FeynRules will automatically create the symbol `psibar` for the antiparticle. Finally, the interaction term can be entered in a similar way,

```
-e psibar.Ga[mu].psi A[mu]
```

Putting those three terms together, we obtain the *Mathematica* expression for the QED Lagrangian, which looks formally the same as the textbook expression given in Eq. (1).

Derivation of the Feynman Rules

Once FeynRules reads the model file and the Lagrangian has been defined, it is ready to compute the Feynman rules, which is done by running the command `FeynmanRules[`$\mathcal{L}$, *options*`]`, where $\mathcal{L}$ denotes the Lagrangian and *options* is a set of 'selection rules' that specify which subset of the full set of vertices should be computed (*e.g.* only those vertices involving a given number or combination of fields). The evaluation of the vertices from the Lagrangian involves two main steps outlined below.

In a first step, FeynRules prepares the Lagrangian. It does this by first checking that the Lagrangian fulfills the basic requirements of quantum field theory such as Lorentz and gauge invariance. It next applies the selection rules defined by the user. Applying the selection rules at this point allows to speed up the algorithm by avoiding the computation of undesired vertices. FeynRules then scans the remaining Lagrangian terms, identifying the fields in each, and groups terms with the same field content.

Once the Lagrangian is prepared, FeynRules applies standard quantum field theory rules in order to extract the interaction vertex. It multiplies on the right of each term by the creation operators for the fields, and then moves them all to the left of the expression by making recursive use of the canonical quantization commutation relations[1]

$$\left[\psi_i(x), a_j^\dagger(p)\right] = \delta_{ij} u_i(p) e^{-ipx}, \qquad (2)$$

where ψ_i denotes a field of type i, and $a_j^\dagger(p)$ is the creation operator associated to the field ψ_j. After having moved all creation operators to the left and dropping the external wave functions $u_i(p)e^{-ipx}$ for the fields, we are left with the interaction vertex. We note that this algorithm takes care of the (anti-)symmetrization required if identical fields are present in the vertex.

The vertices are then stored in a *Mathematica* list and can be directly used in further computations inside *Mathematica*, exported into a LaTeX file or translated to the model file format appropriate for one of the Feynman diagram calculator as described in the next section.

Translation interfaces

Although the information about a model is generic, the way each Feynman diagram calculator stores that information is different and not compatible with one another. For this reason, FeynRules stores the information in a generic way and allows translation interfaces to be written. Currently translation interfaces for FeynArts/FormCalc, CalcHep/CompHep, MadGraph/MadEvent and Sherpa are available, but we hope to have many more in the future. Although the file formats for the different Feynman diagram calculators can be quite different, the translation interfaces have nevertheless a common structure which will be described in the following.

A translation interface begins by scanning the field and parameter classes, dropping information which is not used in its Feynman diagram calculator and warning the user about possible incompatabilities. If the scan is successful, it stores the condensed information in new temporary lists and writes them to a file in the format appropriate for the particular Feynman diagram calculator. The translation interfaces also call the function `FeynmanRules` with the appropriate options for its Feynman diagram calculator and stores the resulting vertices in a list. It then scans and filters the list of vertices, keeping only the vertices that are supported in the Feynman diagram calculator being written to and warns the user about the vertices that are not supported. Finally, it writes the vertices to file in the appropriate format. The files are then ready for use in the Feynman diagram calculator[2].

Conclusion

In this article we have presented FeynRules, a *Mathematica* package which extracts Feynman rules from a Lagrangian, and allows to implement new physics models into several Feynman diagram calculators in an automated way. We have briefly reviewed the structure of the FeynRules input files, the algorithm used to derive Feynman rules from a Lagrangian, and how translation interfaces write this information in a format appropriate for one of the Feynman diagram calculators. Such interfaces are currently available for CalcHEP/CompHEP, FeynArts/FormCalc, MadGraph/MadEvent and Sherpa, but we hope that we can extend this list in the future. Several models have been implemented in FeynRules, translated to Feynman diagram calculators and successfully compared with the model implementations previously existing. The FeynRules code, the translation interfaces and a set of implemented models are available and can be downloaded from the FeynRules website [1].

Acknowledgments

C. Duhr is a research fellow of the "*Fonds National de la Recherche Scientifique*", Belgium. N. D. Christensen was supported by the US National Science Foundation under grants PHY-0354226 and PHY-0555544.

Literatur

[1] N. D. Christensen and C. Duhr, arXiv:0806.4194 [hep-ph], `http://feynrules.phys.ucl.ac.be/`

[1] We give here the commutation relations for bosons. For fermions anticommutation relations are applied.

[2] In some cases, the output files have to be compiled.