FeynRules Tutorial

The aim of this tutorial is to implement a simple extension of the SM into FeynRules The model itself is described in a separate handout. In the following we give detailed instructions of how to implement the model into FeynRules and how to obtain the corresponding model files for CalcHEP, MadGraph 5 and Sherpa.

1 Preparation of the model file

As the model we are going to implement is a simple extension of the SM, it is not necessary to start from scratch, but we can use the implementation of the SM included in the folder `/Models/SM`. We therefore start by making a copy of this folder, in order to keep a clean version of the SM. To do so, change directory to the `Models` subdirectory and make a copy of the SM folder, before going into the new directory

```
cd Models
cp -r SM MC4BSM
cd MC4BSM
```

Even though the implementation is based on the model file `SM.fr` for the Standard Model, the SM sector of the model will be implemented into a separate file that will simply be loaded on top of `SM.fr`. We therefore start by opening a blank text file called `MC4BSM.fr`. You can start by personalizing the model file by including a name for you model, the name of the author, *etc.*,

```
M$ModelName = "MC4BSM_2012";

M$Information = {Authors      -> {"C. Duhr"},
                 Version      -> "1.0",
                 Date         -> "27. 02. 2012",
                 Institutions -> {"ETH Zurich"},
                 Emails       -> {"duhrc@itp.phys.ethz.ch"}
             };
```

Note that his information is purely optional and could be omitted.

2 Implementation of the new parameters

We start by implementing the new parameters. The model we are considering depends on 9 new parameters, which we assume to be real in the following. FeynRules distinguishes between two types of parameters, the so-called *external* parameters given as numerical inputs and the *internal* parameters, related to other external and/or internal parameters via algebraic expressions. All the parameters of the model, both external and internal, are given in the FeynRules model file as a list named `M$Parameters`. Note that if an internal parameter x depends on some other parameter y, then y should appear in `M$Parameters` *before* the internal parameter x.

The new external parameters of the model are

- 5 mass parameters: m_1, m_2, m_{12}, M_U, M_E.
- 4 coupling constants: λ_1, λ_2, λ_1', λ_2'.

Note however that there is a difference between the mass parameters in the scalar and fermonic sectors: while the masses in fermonic sector are physical masses, the mass matrix in the scalar sector is not diagonal. For this reason, we will not discuss in the following the fermion masses M_U and M_E, as they will be defined together with the particles rather than as parameters of the model.

Let us now turn to the definition of the mass parameters in the scalar sector. The masses m_1, m_2 and m_{12} will be denoted in the FeynRules model file by `MM1`, `MM2` and `MM12`. In the following we only show how to implement `MM1`, all other cases being similar. `MM1` corresponds to the following entry in the list `M$Parameters`,

```
M$Parameters = {
    ...
    MM1 == {
           ParameterType -> External,
           Value         -> 200
         },
    ...
}
```

The first option tags `MM1` as an external parameter, while the second option assign a value of 200GeV to m_1. We stress that this numerical value can be changed later on in the matrix element generators.

The masses in the scalar sector are not the physical masses, because the mass matrix is not diagonal. In order to obtain the physical masses, we need to diagonalize the mass matrix

$$\begin{pmatrix} m_1^2 & m_{12}^2 \\ m_{12}^2 & m_2^2 \end{pmatrix}. \tag{1}$$

In the following, we denote the eigenvalues by `MPe1` and `MPe2`. In addition, we need to introduce a mixing angle θ (`th`) relating the fields ϕ_i to the mass eigenstates Φ_i by,

$$\begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \begin{pmatrix} -\sin\theta & \cos\theta \\ \cos\theta & \sin\theta \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \end{pmatrix}. \tag{2}$$

As in this case the mass matrix is only two-dimensional, we can compute the eigenvalues and the mixing angle analytically, and simply implement the analytical formulas into FeynRules. The implementation follows exactly the same lines as for the masses m_1, m_2, m_{12}, with the only differences that

1. the `ParameterType` is `Internal` (as these parameters are dependent on the external mass parameters,
2. the `Value` is given by an analytical expression (in Mathematica syntax).

Next we turn to the implementation of the new coupling constants, which we will call `lam1`, `la21`, `lam1p`, `lam2p`. They are all external parameters, and thus the implementation follows exactly the same lines as the implementation of the mass parameters, with only one modification: some matrix element generators, like for example MadGraph, keep track of the types of couplings that enter a process. This allows for example to generate a process by only taking into account QCD-type vertices, and to neglect all QED-type vertices. For this reason, it is mandatory to tell the matrix element generator how the new coupling constants should be counted. As in this case we are dealing with new classes of couplings which are *a priori* independent of QCD or QED interactions, we simply assign a new tag, called *interaction order*, to the coupling via the option

```
InteractionOrder ->  {NP, 1}
```

The name of the tag (`NP` for "new physics" in this case) can be chosen freely. The above option instructs the matrix element generator to count one unit of "`NP`" for each new coupling.

3 Implementation of the fields

In this section we discuss the implementation of the new fields. The implementation is similar to the implementation of the parameters, *i.e.*, all the fields are entries of a list called `M$ClassesDescription`. In Tab. 1 we show the names of the fields used in the implementation[1].

U	E	ϕ_1	ϕ_2	Φ_1	Φ_2
uv	ev	pi1	pi2	p1	p2

Table 1: Symbols used for the fields in the FeynRules implementation.

We illustrate the implementation of a new field on the example of the particle U (`uv`). The definition of the particle corresponds to an entry in `M$ClassesDescription` of the following form

```
M$ClassesDescription = {
      ...
      F[10] == {
          ClassName      -> uv,
          SelfConjugate  -> False,
          Indices        -> {Index[Colour]},
          QuantumNumbers -> {Y -> 2/3, Q -> 2/3},
          Mass           -> {Muv, 500},
          Width          -> {Wuv,1}
        },
       ...
 }
```

The meaning of this definition is as follows: each particle class has a name of the form `X[i]`, where `X` is related to the spin of the field (See Tab. 2), and `i` is an integer that labels the classes. Note that `i` can be chosen freely, as long as there is no name clash with an already existing class (in this case, there could be a name clash with the SM particles already defined in `SM.fr`). Each class has a series of options

1. `ClassName`: the symbol by which the particle will be represented in the Lagrangian.
2. `SelfConjugate`: a boolean variable, indicating whether the particle has an antiparticle (`False`) or not (`True`). If the field is not selfconjugate, a symbol for the antiparticle is automatically defined by appending "`bar`" to the name of the particle. In the above example the antiparticle associated to `uv` will be denoted by `uvbar`. Note that in the case of fermions the symbol for the antiparticle refers to the quantity $\bar{U}$ rather than $U^\dagger$.

[1] Note that the symbol `u`, `e` and `phi` are already in use in the SM implementation. We also avoid using simply uppercase letters, as some matrix element generators are case insensitive.

3. `Indices`: All indices carried by the field. The available types of indices from the SM implementation are
   - `Generation`: fermion flavor index ranging from 1 to 3,
   - `Colour`: fundamental color index ranging from 1 to 3,
   - `Gluon`: adjoint color index ranging from 1 to 8,
   - `SU2W`: adjoint $SU(2)_L$ index ranging from 1 to 3.
4. `QuantumNumbers`: a list of all $U(1)$ charges carried by the field. In the SM implementation the following $U(1)$ charges are already defined
   - `Y`: weak hypercharge,
   - `Q`: electric charge.
5. `Mass`: the mass of the particle. It is a list of two elements, the first being the symbol used to represent the mass, and the second its value (in GeV). If the value of the mass is obtained from some analytic expression defined as an internal parameter with the same symbol (as is the case for example in the scalar sector of the model), the value is set to `Internal`.
6. `Width`: the width of the particle. The definition is similar to `Mass`. Note that as we do not yet know the widths of the new particles, we simply set it for now to 1GeV, and will determine its exact value later using one of the matrix element generators.

The implementation of the other mass eigenstates (`ev`, `p1`, `p2`) is similar, so we do not discuss it here.

Spin	0	1/2	1	2	ghost
Symbol	S	F	V	T	U

Table 2: Available particle classes in FeynRules.

Let us comment on the implementation of the interaction eigenstates ϕ_i. Indeed, while the matrix element generators work exclusively at the level of the mass eigenstates, the interaction eigenstates are in general useful to write the Lagrangian in a compact form. It is therefore useful to define also the fields for the interaction eigenstates ϕ_i. The definition of these fields is similar to the mass eigenstates, *e.g.*,

```
S[10] == {
      ClassName     -> pi1,
      SelfConjugate -> True,
      Indices       -> {},
      Unphysical    -> True,
      Definitions   -> {pi1  ->  - Sin[th] p1 + Cos[th] p2}
     },
```

First, note that the `Mass` and `Width` options are omitted[2], as these fields are not mass eigenstates. This last fact is made explicit by the option

[2]The `QuantumNumbers` option is also omitted, but for the simple reason that the fields ϕ_i do not carry any $U(1)$ charges.

```
Unphysical    -> True,
```

which instruct FeynRules not to output this field to a matrix element generator. Finally, the relation of the field `pi1` to the mass eigenstates is simply given as a Mathematica replacement rule in the `Definitions` option.

4 Implementation of the Lagrangian

The definitions in the model file being complete, we now turn to the implementation of the Lagrangian. This can be done either in the model file, or directly in a Mathematica notebook. Here we use the latter approach, and we start by opening a new notebook and load the FeynRules package (see the preinstallation instructions). Next we have to load the model files, both for the SM and for the new sector,

```
LoadModel["SM.fr", "MC4BSM.fr"]
```

Note that the new model file should be loaded after `SM.fr`. Furthermore, we also load two additional files, which restrict the first two fermion generations to be massless and the CKM matrix to be diagonal,

```
LoadRestriction["DiagonalCKM.rst", "Massless.rst"]
```

The new Lagrangian consists of three parts,

$$\mathcal{L} = \mathcal{L}_{scalar,kin} + \mathcal{L}_{fermion,kin} + \mathcal{L}_{Yuk}\,. \tag{3}$$

The kinetic terms for the new scalars can easily be implemented by using the symbols for the gauge eigenstates and the mass parameters defined in the model file, as well as the symbol for the space-time derivative ∂_μ in FeynRules, `del[ ..., mu]`. As an example, we have

$$\frac{1}{2}\partial_\mu\phi_1 + \partial^\mu\phi_1 - \frac{1}{2}m_1^2\phi_1^2$$

```
1/2 del[p1, mu] del[p1, mu] - 1/2 MM1^2 p1^2
```

The kinetic terms for the fermions can be implemented in a similar way. However, as the fermions are charged under the SM gauge group, we have to use the covariant derivative `DC` rather than the space-time derivative `del`. Furthermore, we have to use a "." instead of an ordinary multiplication in order to take the non-commuting nature of the fermions into account. As an example, we have

$$i\,\bar{U}\gamma^\mu D_\mu U - M_U\bar{U}U$$

```
I uvbar.Ga[mu].DC[uv, mu] - Muv uvbar.uv
```

where `Ga[mu]` is the FeynRules symbol for the Dirac matrix γ^μ. Finally, the Yukawa interactions can be implemented in the same way as the kinetic terms for the fermions, *e.g.*,

$$\lambda_1\,\phi_1\,\bar{U}P_+u$$

```
lam1 pi1 uvbar.ProjP.u
```

where `u` denotes the u quark field defined in `SM.fr` and `ProjP` denotes the right chiral projector (the left projector is denoted by `ProjM`). Note that FeynRules contains a function `HC[ ]` which allows to obtain the hermitian conjugate of an expression in an automated way.

5 Computing the Feynman rules and running the interfaces

Our model implementation is now complete, and so we can compute the Feynman rules. The Feynman rules of the new sector can be obtained by issuing the command

```
FeynmanRules[ LNew ]
```

where `LNew` is the name of the variable that contains the new Lagrangian.

The Feynman rules can be written to file in a format suitable to various matrix element generators by using the FeynRules interfaces. In this tutorial, we will use the interfaces to CalcHEP, MadGraph 5 and Sherpa, which can be called via

```
WriteCHOutput[ LSM + LNew ];
WriteUFO[ LSM + LNew ];
WriteSHOutput[ LSM + LNew ];
```

where `LSM` is the SM Lagrangian implemented in `SM.fr`. Note that the SM implementation is available in both Feynman gauge and unitary gauge. A boolean variable `FeynmanGauge` allows to switch between both gauges. While CalcHEP and MadGraph 5 support both gauges, Sherpa only supports unitary gauge, and the `FeynmanGauge` variable should correspondingly be set to `False`.