FeynRules and UFO

Claude Duhr

in collaboration with A. Alloul, N. Christensen
C. Degrande and B. Fuks
+ many other external collaborators

Outline

- FeynRules in a nutshell.
- FeynRules 2.0: Recent developments:
  - ➡ Spin 3/2 particles.
  - ➡ ASperGe - Automatic Spectrum Generation.
  - ➡ Two-body decays
  - ➡ NLOCT - NLO counterterms.

FeynRules
in a nutshell

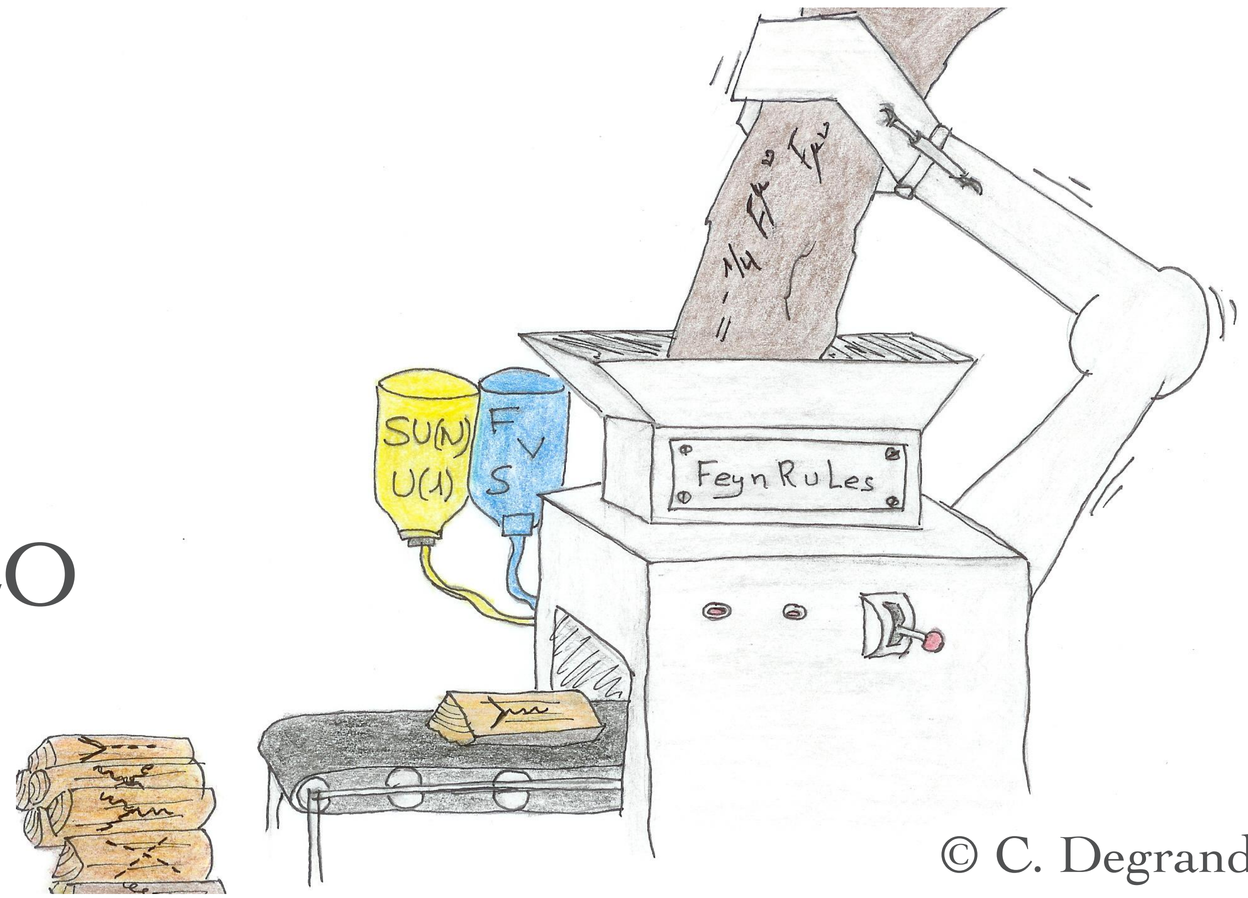
FeynRules in a nutshell

- FeynRules is a Mathematica package that allows to derive Feynman rules from a Lagrangian.
- Current public version 2.x available from

  http://feynrules.irmp.ucl.ac.be/
- The only requirements on the Lagrangian are:
  - ➡ All indices need to be contracted (Lorentz and gauge invariance).
  - ➡ Locality.
  - ➡ Supported field types: spin 0, 1/2, 1, 3/2, 2 & ghosts
  - ➡ Chiral and vector superfields are also supported.
  - ➡ Fields can be massive or massless, self conjugate or not.

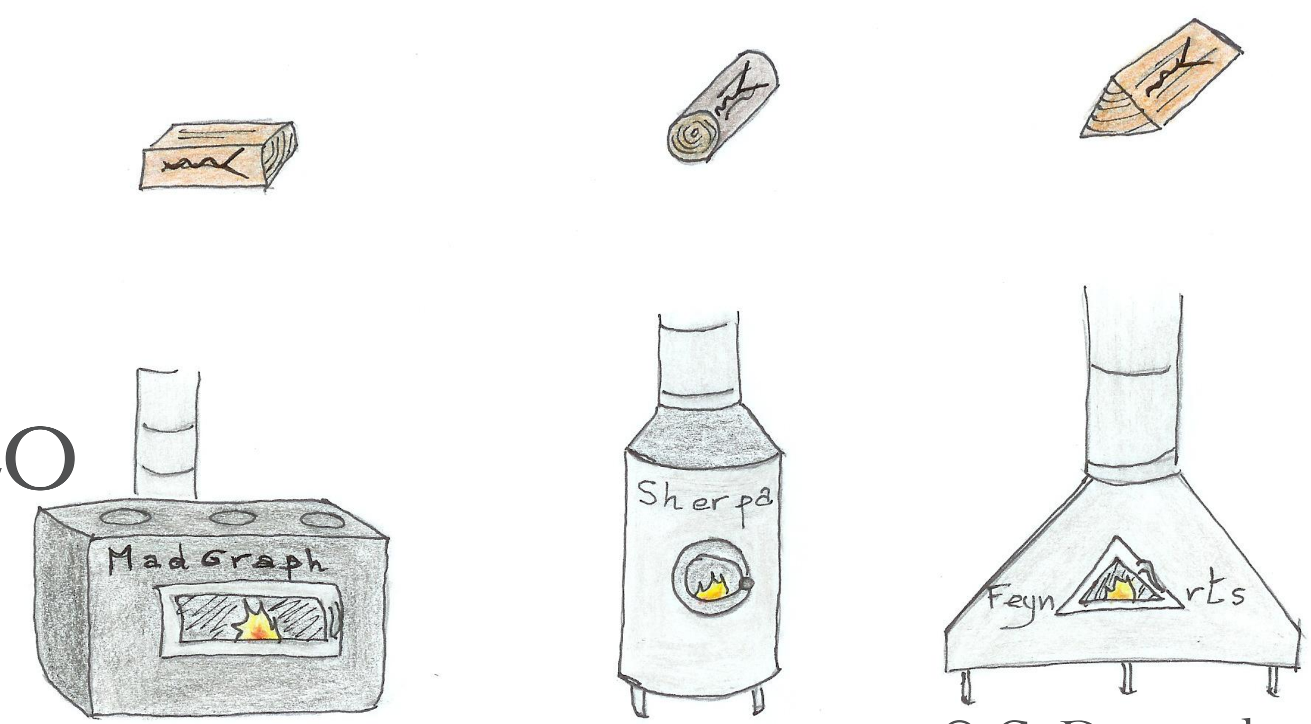
FeynRules in a nutshell

- FeynRules comes with a set of interfaces, that allow to export the Feynman rules to various matrix element generators.
- Interfaces coming with current public version
  - ➡ CalcHep / CompHep
  - ➡ FeynArts / FormCalc
  - ➡ GoSam
  - ➡ Herwig++
  - ➡ MadGraph5_aMC@NLO
  - ➡ Sherpa
  - ➡ Whizard / Omega



FeynRules in a nutshell

- FeynRules comes with a set of interfaces, that allow to export the Feynman rules to various matrix element generators.
- Interfaces coming with current public version
  - ➡ CalcHep / CompHep
  - ➡ FeynArts / FormCalc
  - ➡ GoSam
  - ➡ Herwig++
  - ➡ MadGraph5_aMC@NLO
  - ➡ Sherpa
  - ➡ Whizard / Omega



© C. Degrande

FeynRules in a nutshell

- The input requested form the user is twofold.

- The Model File:
  Definitions of particles and parameters (e.g., a quark)

```
F[1]  ==
  {ClassName      ->   q,
   SelfConjugate ->  False,
   Indices           ->  {Index[Colour]},
   Mass              ->  {MQ,  200},
   Width             ->  {WQ, 5}   }
```

- The Lagrangian:

$$\mathcal{L} = -\frac{1}{4} G^a_{\mu\nu}\, G_a^{\mu\nu} + i\bar{q}\,\gamma^\mu\, D_\mu q - M_q\,\bar{q}\,q$$

```
L =
-1/4 FS[G,mu,nu,a] FS[G,mu,nu,a]
+ I qbar.Ga[mu].del[q,mu]
- MQ qbar.q
```

FeynRules in a nutshell

- Once this information has been provided, FeynRules can be used to compute the Feynman rules for the model:

FeynmanRules[L]

Vertex 1

Particle 1 : Vector , G

Particle 2 : Dirac , $q^{\dagger}$

Particle 3 : Dirac , q

Vertex:

$$i\,\mathrm{gs}\,\gamma^{\mu_1}{}_{\mathrm{s}_2,\mathrm{s}_3}\,\delta_{\mathrm{f}_2,\mathrm{f}_3}\,T^{\mathrm{a}_1}{}_{\mathrm{i}_2,\mathrm{i}_3}$$

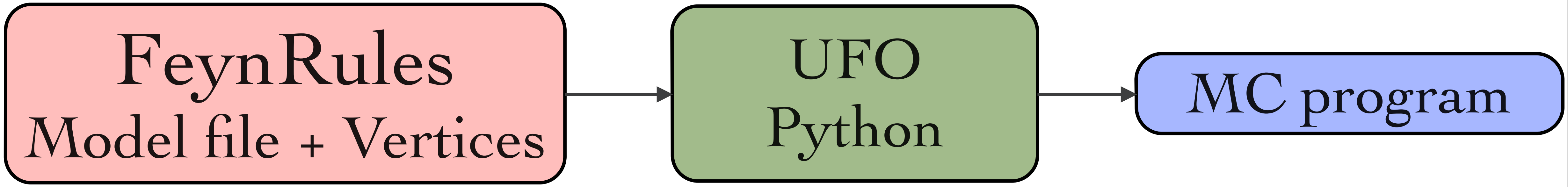
FeynRules in a nutshell

- **Idea:** Feynman rules can be exported to various matrix element generators.
- **Drawback:** Need to develop and maintain a lot of interfaces for various Monte Carlos.
- **UFO (Universal FeynRules Output):**
  - ➡ A model is a Python module that can be linked to other codes.
  - ➡ Some generators have restrictions on the type of vertices they can handle.
  - ➡ A warning is thrown if a model cannot be exported to a certain code.

The UFO

UFO = Universal FeynRules Output

- **Idea:** Create Python modules that can be linked to other codes and contain all the information on a given model.



- **By design:** No assumptions on MC program specific information.

  ➡ Lorentz/color structures, number of particles.

- UFO files can be read by GoSam, Herwig++, MadGraph5_aMC@NLO, Sherpa.

Additional Features

- Higher-dimensional operators are fully supported!
  - ➡ Only limitations come from limitations in the matrix element generators.
- Sextet color algebra fully supported.
- Supersymmetric models can be implemented using super field formalism.
- Large database of implemented models online:
  - ➡ MSSM + various extensions.
  - ➡ Extra dimensions.
  - ➡ SM + effective operators
  - ➡ ...

FeynRules 2.0

Recent developments

Spin 3/2 [Christensen, de Aquino, Deutschmann, CD, Fuks, Garcia-Cely, Mattelaer, Mawatari, Oexl, Takaesu]

- Particles of spin 3/2 are fully supported starting from v2.0.
  - ➡ Colored or non-colored.
  - ➡ Majorana or Dirac.
- Currently supported by UFO and CalcHEP interfaces.
- Example:

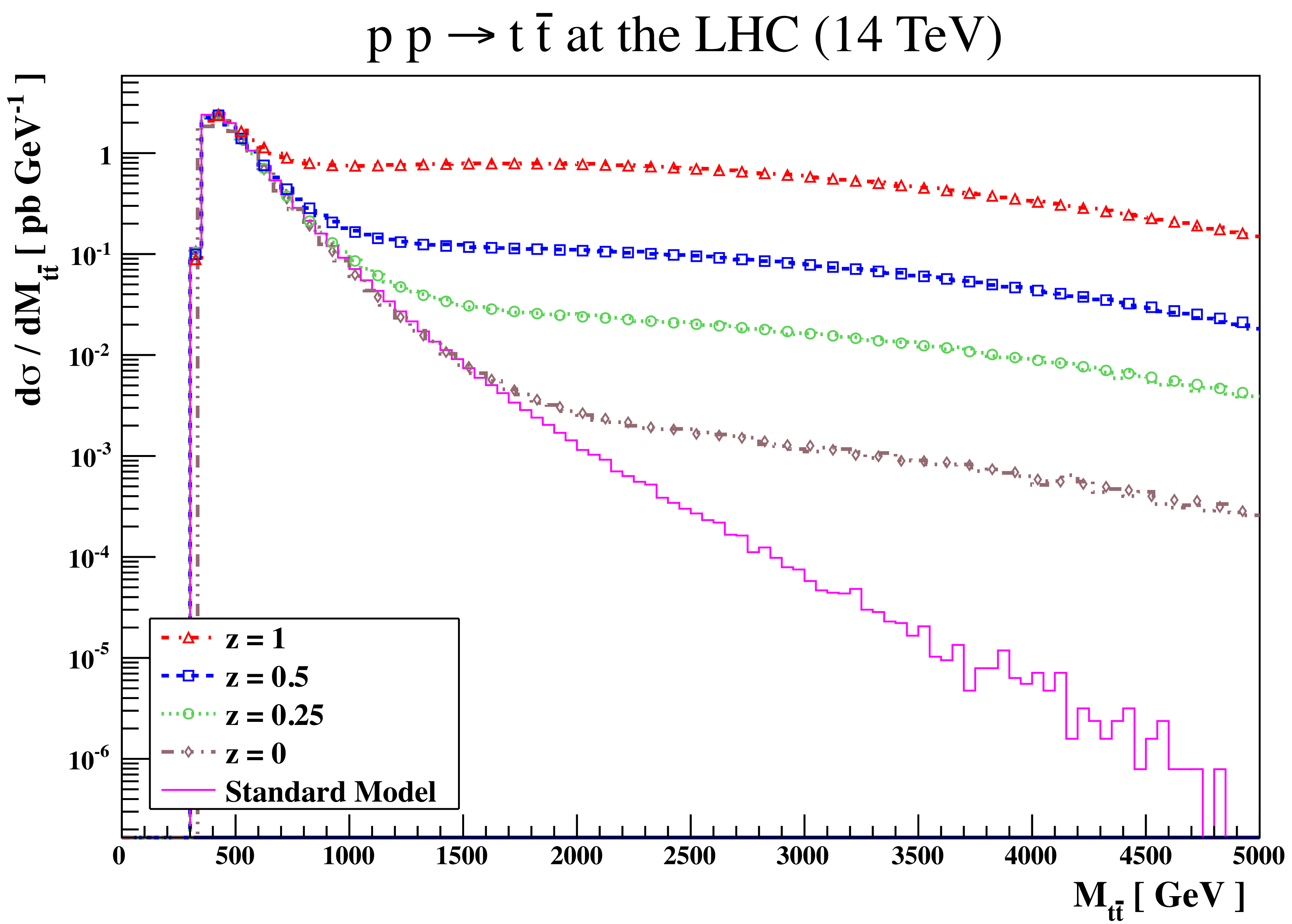
$$\mathcal{L} = \mathcal{L}_{SM} + \mathcal{L}_{3/2} + \mathcal{L}_5$$

$$\mathcal{L}_{3/2} = \epsilon^{\mu\nu\rho\sigma}\bar{\Psi}_\mu\gamma_5\gamma_\sigma D_\nu\Psi_\rho + 2iM\bar{\Psi}_\mu\gamma^{\mu\nu}\Psi_\nu$$

$$\mathcal{L}_5 = \quad i\frac{g_s}{\Lambda}\bar{\Psi}_\rho\left[\eta^{\rho\mu} + z\gamma^\rho\gamma^\mu\right]\gamma^\nu T_a t\ g^a_{\mu\nu} + \text{h.c.}$$

Spin 3/2

[Christensen, de Aquino, Deutschmann, CD, Fuks, Garcia-Cely, Mattelaer, Mawatari, Oexl, Takaesu]



- ➡ Not more difficult to implement that 'ordinary' particles.
- ➡ Perfect agreement between CalcHEP and MadGraph 5.

Diagonalisation of Mass Matrices

- In many BSM models the mass matrix is not diagonal.

$$\mathcal{L} = D_\mu \phi_i^\dagger D^\mu \phi_i - m^2 \phi_i^\dagger \phi_i \; - m_{12}^2 (\phi_1^\dagger \phi_2 + \phi_2^\dagger \phi_1) \; + \lambda (\phi_i^\dagger \phi_i)^2$$

- The gauge and mass eigenstates are then related via some unitary rotation,

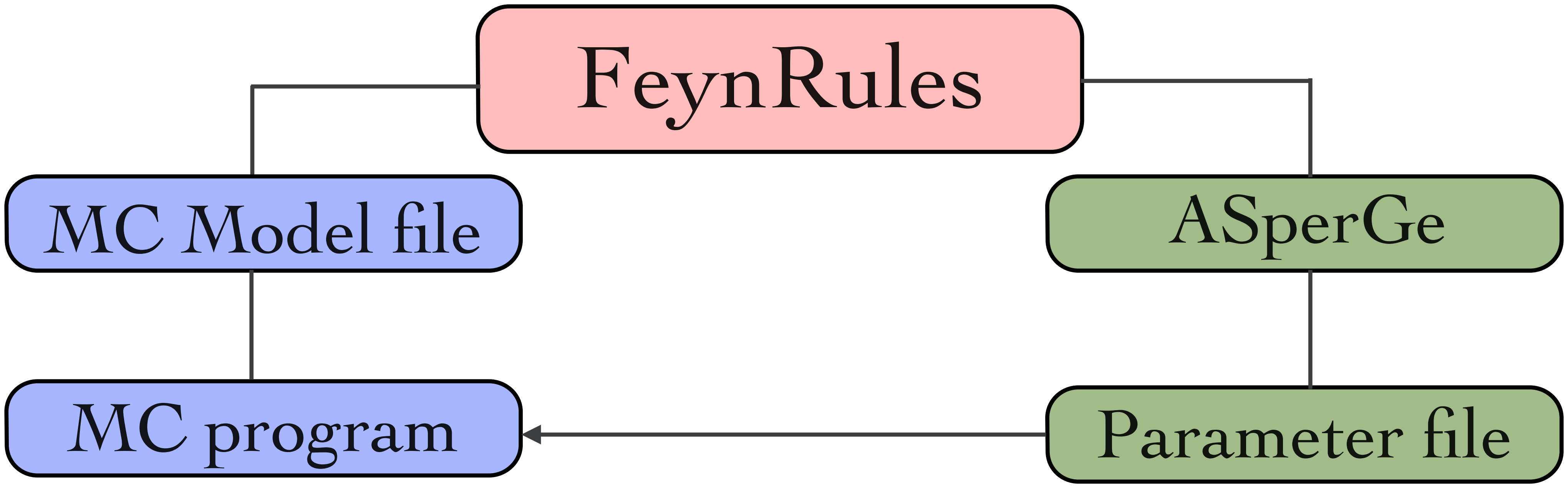
$$\begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = U \begin{pmatrix} \Phi_1 \\ \Phi_2 \end{pmatrix}$$

- Dilemma:
  - ➡ Lagrangian is simple in terms of the gauge eigenstates.
  - ➡ MC tools generically require mass eigenstates as input.
- Problem:
  - ➡ For small mixing matrices (2x2), diagonalization is easy.
  - ➡ MSSM: 6x6 squark mixing matrix.

ASperGe

[Alloul, D'Hondt, de Causmaecker, Fuks, Rausch de Traubenberg]

- ASperGe = Automatic Spectrum Generator
- ASperGe allows to
  - ➡ Extract the mass matrix from the Lagrangian.
  - ➡ Outputs a standalone numerical code taking as input a SLHA-like parameter file
    information, and returns the complete parameter file



ASperGe

[Alloul, D'Hondt, de Causmaecker, Fuks, Rausch de Traubenberg]

- Input: Definition of the mixing matrix in the model file (without specifying the numerical values!):

```
Mix["AZmix"] == {
        MassBasis -> {A, Z},
        GaugeBasis -> {B, Wi[3]},
        MixingMatrix -> UW,
        BlockName -> WEAKMIX
        }
```

$$\begin{pmatrix} A_\mu \\ Z_\mu \end{pmatrix} = U_w \begin{pmatrix} B_\mu \\ W^3_\mu \end{pmatrix}$$

- FeynRules can be used to extract the mass matrix from the Lagrangian:

```
ComputeMassMatrix[ Lagrangian ]
```

ASperGe

[Alloul, D'Hondt, de Causmaecker, Fuks, Rausch de Traubenberg]

- If the mass matrix cannot be diagonalised analytically, FeynRules can output an ASperGe-code that allows to

```
WriteASpergGe[ Lagrangian ]
```

- Result: A standalone C++ code for the diagonalization!
  - ➡ No need to return to Mathematica at this point!

```
./ASperGe   <infile>    <outfile>
```

- Extensively tested:
  - ➡ MSSM, Left-Right MSSM, Most general 2HDM ,…

Two-body decays

[Alwall, CD, Fuks, Mattelaer, Öztürk, Shen]

- MC programs generically require the total widths of all particles to be given as numerical inputs.
  - ➡ Widths are not independent input parameters!
- Most MC programs have the capability to compute the widths (on the fly).
  - ➡ Tedious, and requires to recompute the widths every time a numerical input parameter has changed.
- In many cases two-body decays are dominant.
  - ➡ Two-body decays are just 'squares' of 3-point vertices!
  - ➡ All relevant information already at FeynRules level.
  - ➡ Can use Mathematica to compute all two-body decays analytically.

Two-body decays

[Alwall, CD, Fuks, Mattelaer, Öztürk, Shen]

- FeynRules 2.0 can compute all two-body partial decay widths analytically:

```
vertices = FeynmanRules[ L ];

decays = ComputeDecays[ vertices ];
```

- The partial and total widths can easily evaluated numerically and inserted into the model file.

```
x = PartialWidth[{t,W,b}]          y = TotWidth[ t ]

NumericalValue[ x ]                NumericalValue[ y ]
```

Two-body decays

[Alwall, CD, Fuks, Mattelaer, Öztürk, Shen]

- The partial widths can be output in the UFO format, and be used when generating a process.

```
Decay_H = Decay(name = 'Decay_H',
          particle = P.H,
          partial_widths = {
                    (P.b,P.b__tilde__):'3*MH**2*yb**2',
                    (P.ta__minus__,P.ta__plus__):'MH**2*ytau**2',
                    (P.c,P.c__tilde__):'3*MH**2*yc**2',
                    (P.t,P.t__tilde__):'3*MH**2*yt**2'})
```

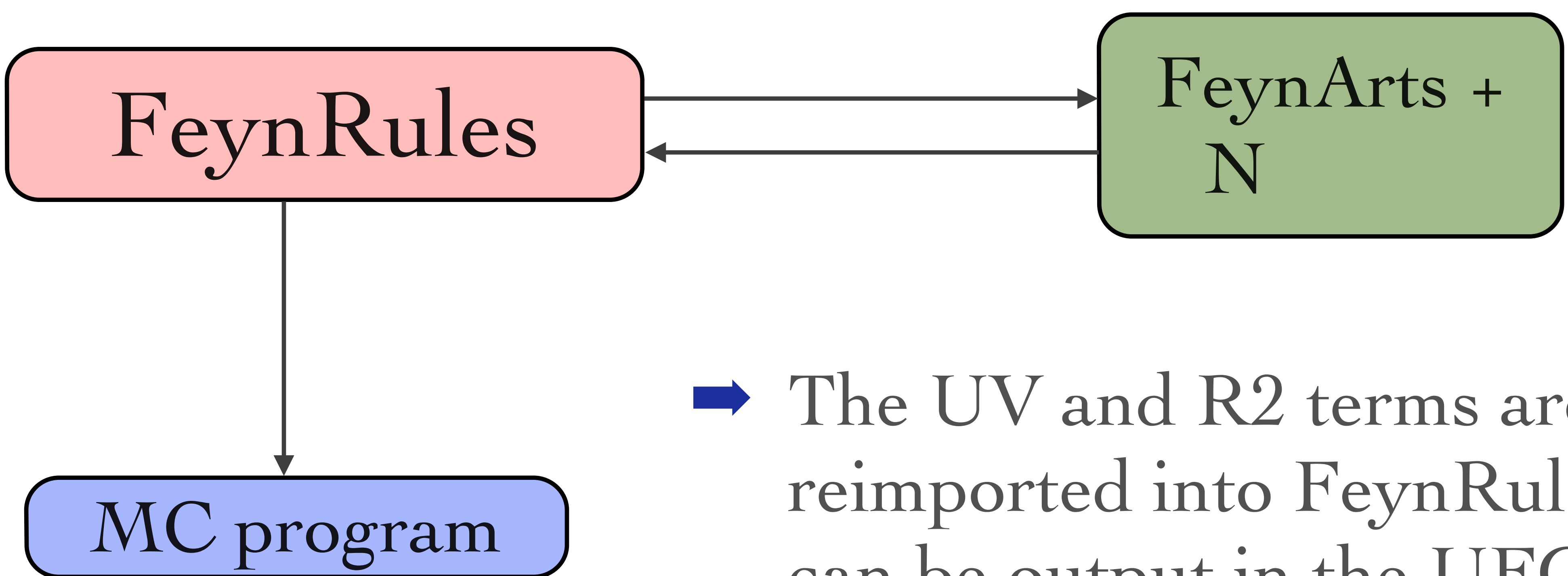
- NB: All possible analytic formulas are output, independently whether they are kinematically allowed!
  - ➡ Some channels might be open for some benchmark scenarios but not for other
  - ➡ Channels depend on spectrum.

NLOCT - NLO counterterms [Degrande]

- NLO accuracy is/will soon be the new standard for MC simulations.
- Next goal: Bring NLO models to the same level of automation than at LO!
    - ➡ UV counterterms.
    - ➡ 'R2' vertices (effective tree-level vertices arising from (d-4)-dimensional part of numerators).
- The information cannot exclusively be extracted from a tree-level Lagrangian!
    - ➡ Requires the evaluation of loop integrals.
- The NLOCT package allows to solve this problem!

NLOCT - NLO counterterms [Degrande]

- Idea of NLOCT:
  - ➡ Use FeynRules interface to FeynArts to implement the model into FeynArts.
  - ➡ Use FeynArts to write the relevant amplitudes and NLOCT to compute their R2 and UV parts.



- ➡ The UV and R2 terms are reimported into FeynRules and can be output in the UFO format.

NLOCT - NLO counterterms [Degrande]

- **Step 1:** Introduce renormalization constants for all fields and parameters, and determine renormalization constants for dependent parameters.

```
Lren = OnShellRenormalization[ L ];
```

- **Step 2:** Output the model to FeynArts.

```
WriteFeynArtsOutput[ Lren ];
```

- **Step 3:** Run FeynArts and NLOCT to compute the counterterms:

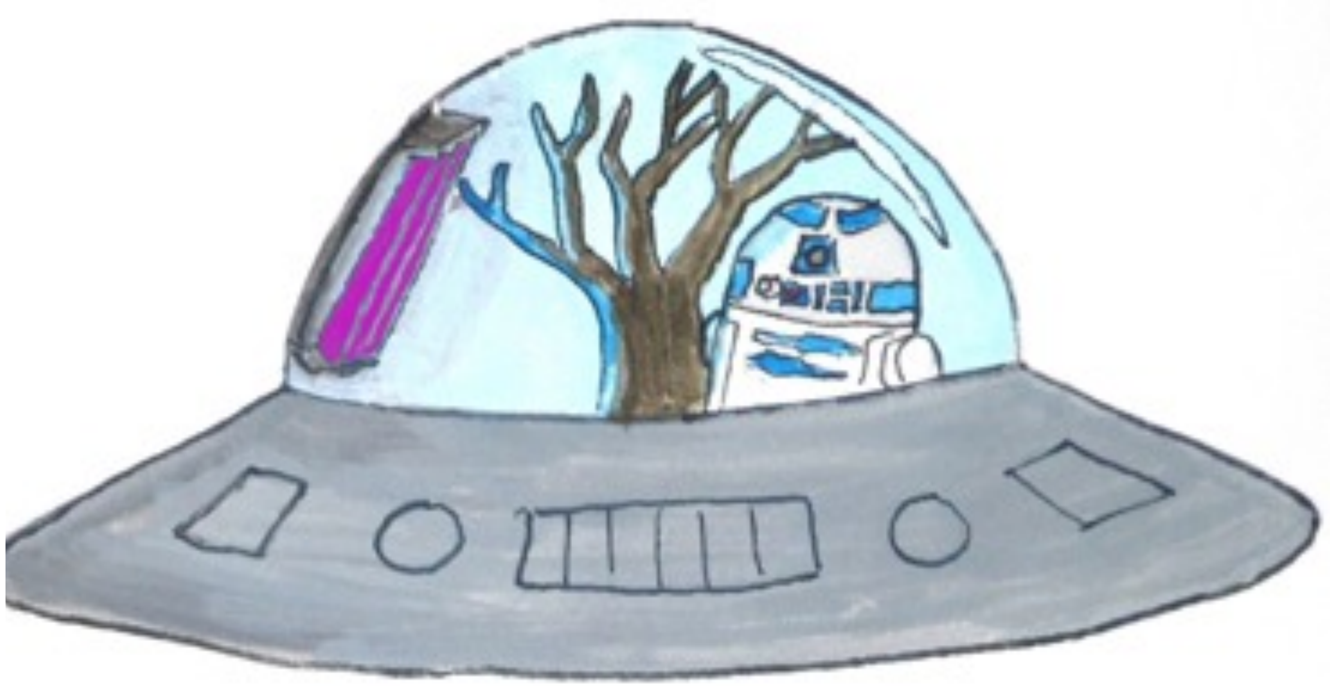
```
WriteCT[ "model", "generic_model", options ];
```

NLOCT - NLO counterterms [Degrande]

- Step 4: Reimport the counterterms into FeynRules, and export them together with the tree-level vertices in the UFO format.

```
Get[ "model.nlo" ]

WriteUFO[ L, UVCounterterms -> UV$vertlist,
             R2Vertices -> R2$vertlist ]
```



NLOCT - NLO counterterms [Degrande]

- Step 4: Reimport the counterterms into FeynRules, and export them together with the tree-level vertices in the UFO format.

```
Get[ "model.nlo" ]

WriteUFO[ L, UVCounterterms -> UV$vertlist,
             R2Vertices -> R2$vertlist ]
```

- The result is a UFO file that can immediately be used by MadGrapgh 5/aMC@NLO to produce events at NLO accuracy!
- Validation: Reproduces correctly the counterterms and R2 terms for the SM.
  - ➡ Validation of MSSM on-going.

Summary

- FeynRules 2.0 has been released 6 months ago.
- New features:
  - ➡ Spin 3/2.
  - ➡ Automatic computation of two-body decays.
  - ➡ Numeric diagonalisation of mass matrices.

    ASperGe
  - ➡ Extraction of one-loop counterterms and R2 vertices for renomalisable models (from 2.1).

    NLO
- Try it out!

  http://feynrules.irmp.ucl.ac.be/