ETH

Eidgenössische Technische Hochschule Zürich

Swiss Federal Institute of Technology Zurich

The UFO

Claude Duhr

Taipei School on MadGraph for LHC physics
National Taiwan University Taipei, 24-27 May 2012

Recap of previous lecture

- In the previous lecture we saw how to
  - ➡ implement a model into FeynRules.
  - ➡ export the model to MadGraph 5.
  - ➡ check a model implementation.
- So far we have used the input files for MadGraph as 'black boxes' without caring to much about what is inside.
- For most applications this is good enough (this is the whole goal of FeynRules!).
- But sometimes it is good to know what is going on behind the scenes...

UFO..?

- UFO = Universal FeynRules Output

  [C. Degrande, CD, B. Fuks, D. Grellscheid, O. Mattelaer, T. Reiter]
- The UFO is a generic format for BSM models that is not tied to any matrix element generator.
- The UFO is the default model format of MadGraph 5 (but not of MadGraph 4!).

UFO..?

- UFO = Universal FeynRules Output

  [C. Degrande, CD, B. Fuks, D. Grellscheid, O. Mattelaer, T. Reiter]
- The UFO is a generic format for BSM models that is not tied to any matrix element generator.
- The UFO is the default model format of MadGraph 5 (but not of MadGraph 4!).
- Question: Why switch to a new model format in MadGraph 4..?
  - ➡ the old format did a good job for many years.
  - ➡ users are familiar with this format.
  - ➡ never change a winning team!!!!
  - ➡ so why did we change it nevertheless..?

Why the UFO..?

- Reason 1: The UFO is ‘agnostic’, and does not make any assumption about any matrix element generator downstream in the code.

Why the UFO..?

- Reason 1: The UFO is 'agnostic', and does not make any assumption about any matrix element generator downstream in the code.
  - ➡ clear factorization between what is model and what is generator!
  - ➡ in other words, clear separation between what FeynRules should provide, and what generators should do with this information.

Why the UFO..?

- Reason 1: The UFO is 'agnostic', and does not make any assumption about any matrix element generator downstream in the code.
  - ➡ clear factorization between what is model and what is generator!
  - ➡ in other words, clear separation between what FeynRules should provide, and what generators should do with this information.
  - ➡ the UFO is not exclusively used by MadGraph 5, but also by other codes.
  - ➡ makes it possible to use the same model file with various generators.

Why the UFO..?

- Reason 2: The old format was not flexible enough:
  - ➡ most model formats are based on text files:

```
# FFV (llZ)
b b g    GG QCD
t  t g   GG QCD

g g g    G    QCD

g g g g G G QCD QCDD
```

Why the UFO..?

- Reason 2: The old format was not flexible enough:
  - ➡ most model formats are based on text files:

```
# FFV (l1Z)
b b g   GG QCD
t  t g   GG QCD

g g g   G   QCD

g g g g G G QCD QCDD
```

- ➡ Fixed number of particles per vertex.
- ➡ Implicit Lorentz structures.
- ➡ Implicit color structures.
- ➡ In other words, everything beyond the ordinary is very hard to implement.

Why the UFO..?

- Reason 2: The old format was not flexible enough:
  - ➡ most model formats are based on text files:

```
# FFV (llZ)
b b g    GG QCD
t  t g    GG QCD

g g g    G    QCD

g g g g G G QCD QCDD
```

- ➡ Fixed number of particles per vertex.
- ➡ Implicit Lorentz structures.
- ➡ Implicit color structures.
- ➡ In other words, everything beyond the ordinary is very hard to implement.

- The UFO does not have these restrictions by design!

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.
- The UFO is a self-contained Python code, and not tied to a specific matrix element generator.
- The content of the FR model files, together with the vertices, is translated into a library of Python objects, that can be linked to other codes.
- By design, the UFO does not make any assumptions on Lorentz/color structures, or the number of particles.
- GoSam and MadGraph 5 use the UFO as the default model format for BSM, Herwig++ will use it in the future.

Plan of the lecture

- What is inside the UFO?
  - ➡ structure of the model files.
  - ➡ what makes it so flexible.

- From the Lagrangian to events:
  - ➡ The FeynRules-UFO-ALOHA-MadGraph chain.

Inside the UFO

Inside the UFO

- The UFO is a fully fledged Python module that can be linked to other codes.
  ➡ no parsing of text files.
- Content of the module:

```
__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py
```

Inside the UFO

```
__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py
```

- Let's have a closer look at these files!

Inside the UFO

__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py

- Standard initialization file present in every Python module.

Inside the UFO

__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py

- File containing the definitions of the classes (Particle, Parameter, ...) used in this Python module.

Inside the UFO

- __init__.py
- object_library.py
- **write_param_card.py**
- function_library.py
- particles.py
- parameters.py
- vertices.py
- lorentz.py
- couplings.py
- coupling_orders.py

- Built-in function to produce Les Houches-style input parameter files (‘param_card.dat’).
- Will be discussed later.

Inside the UFO

__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py

- It is possible to use functions from the cmath library when writing algebraic expressions in the UFO (e.g. cmath.sqrt(x)).
- Sometimes however these functions are insufficient or too cumbersome to use.
- function_library.py allows you to define new customized (algebraic) functions.

Inside the UFO

__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py

➡ Example:

cmath does not contain a definition of sec(x).

- It is possible to use functions from the cmath library when writing algebraic expressions in the UFO (e.g. cmath.sqrt(x)).
- Sometimes however these functions are insufficient or too cumbersome to use.
- function_library.py allows you to define new customized (algebraic) functions.

Inside the UFO

- __init__.py
- object_library.py
- write_param_card.py
- function_library.py
- particles.py
- parameters.py
- vertices.py
- lorentz.py
- couplings.py
- coupling_orders.py

- It is possible to use functions from the cmath library when writing algebraic expressions in the UFO (e.g. cmath.sqrt(x)).
- Sometimes however these functions are insufficient or too cumbersome to use.
- function_library.py allows you to define new customized (algebraic) functions.

➡ Example:

cmath does not contain a definition of sec(x).

```
sec = Function(name = 'sec',
        arguments = ('z',),
        expression = '1./cmath.cos(z)')
```

Inside the UFO

__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py

```
u = Particle(pdg_code = 2,
             name = 'u',
             antiname = 'u~',
             spin = 2,
             color = 3,
             mass = Param.ZERO,
             width = Param.ZERO,
             texname = 'u',
             antitexname = 'u',
             charge = 2/3,
             LeptonNumber = 0,
             GhostNumber = 0)

u__tilde__ = u.anti()
```

➡ The Particle class comes with a predefined method to instantiate the antiparticle.

Inside the UFO

```
__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py
```

```
u = Particle(pdg_code = 2,
             name = 'u',
             antiname = 'u~',
             spin = 2,
             color = 3,
             mass = Param.ZERO,
             width = Param.ZERO,
             texname = 'u',
             antitexname = 'u',
             charge = 2/3,
             LeptonNumber = 0,
             GhostNumber = 0)

u__tilde__ = u.anti()
```

- Allowed colors: 1,3,6,8
- Allowed spins: 0,1/2,1,2

  -1 = ghosts

  3/2 under testing in MadGraph 5.

Inside the UFO

__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py

➡ Parameters are grouped into external and internal, just like in FeynRules.

Inside the UFO

- __init__.py
- object_library.py
- write_param_card.py
- function_library.py
- particles.py
- **parameters.py**
- vertices.py
- lorentz.py
- couplings.py
- coupling_orders.py

➡ Parameters are grouped into external and internal, just like in FeynRules.

➡ An internal parameter

```
v = Parameter(name = 'v',
              nature = 'internal',
              type = 'real',
              value = '(2*MW*sw)/ee',
              texname = 'v')
```

Inside the UFO

```
__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py
```

- Parameters are grouped into external and internal, just like in FeynRules.

- An internal parameter

```
v = Parameter(name = 'v',
        nature = 'internal',
        type = 'real',
        value = '(2*MW*sw)/ee',
        texname = 'v')
```

- An external parameter

```
aS = Parameter(name = 'aS',
        nature = 'external',
        type = 'real',
        value = 0.118,
        texname = '\\text{aS}',
        lhablock = 'SMINPUTS',
        lhacode = [ 3 ])
```

The param_card

- If you want to change a numerical input parameter, you do not need to do this in the UFO!
- Numerical values in the UFO are just default values.
- They can be changed at run time **after** generating the process.
- This can be done in the `param_card.dat`, stored in the directory `/Cards/` of the process directory (just like in MadGraph 4).
- The format of the `param_card.dat` is an extension of the SUSY-Les-Houches accord format.
  - ➡ All parameters are grouped into blocks.

The param_card

```
Block SMINPUTS      # Standard Model inputs
    1        1.32506980E+02   # alpha_em(MZ)(-1) SM MSbar
    2        1.16639000E-05   # G_Fermi
    3        1.18000000E-01   # alpha_s(MZ) SM MSbar
    4        9.11880000E+01   # Z mass (as input parameter)
```

The param_card

```
Block SMINPUTS      # Standard Model inputs
    1        1.32506980E+02   # alpha_em(MZ)(-1) SM MSbar
    2        1.16639000E-05   # G_Fermi
    3        1.18000000E-01   # alpha_s(MZ) SM MSbar
    4        9.11880000E+01   # Z mass (as input parameter)

Block MASS      #  Mass spectrum (kinematic masses)
#      PDG       Mass
       5     4.70000000E+00   # bottom   pole mass
       6     1.72000000E+02   # top      pole mass
```

The param_card

```
Block SMINPUTS      # Standard Model inputs
    1        1.32506980E+02   # alpha_em(MZ)(-1) SM MSbar
    2        1.16639000E-05   # G_Fermi
    3        1.18000000E-01   # alpha_s(MZ) SM MSbar
    4        9.11880000E+01   # Z mass (as input parameter)

Block MASS      #  Mass spectrum (kinematic masses)
#       PDG       Mass
        5     4.70000000E+00   # bottom   pole mass
        6     1.72000000E+02   # top      pole mass

DECAY        6     1.43947825E+00   # top width
DECAY       23     2.44140351E+00   # Z   width
```

The param_card

```
Block SMINPUTS     # Standard Model inputs
    1        1.32506980E+02   # alpha_em(MZ)(-1) SM MSbar
    2        1.16639000E-05   # G_Fermi
    3        1.18000000E-01   # alpha_s(MZ) SM MSbar
    4        9.11880000E+01   # Z mass (as input parameter)

Block MASS      #  Mass spectrum (kinematic masses)
#       PDG       Mass
        5     4.70000000E+00   # bottom   pole mass
        6     1.72000000E+02   # top      pole mass

DECAY       6     1.43947825E+00   # top width
DECAY      23     2.44140351E+00   # Z   width

#       BR          NDA       ID1       ID2
    8.27451012E-02    2          4        -4   # BR( H -> c  cbar  )
    7.17809696E-01    2          5        -5   # BR( H -> b  bbar  )
```

The param_card

```
Block SMINPUTS      # Standard Model inputs
     1        1.32506980E+02  # alpha_em(MZ)(-1) SM MSbar
     2        1.1663900
     3        1.18000000E-01  # alpha_s(MZ) SM MSbar
     4        9.11880000E+01  # Z mass (as input parameter)
```

Model Parameters

```
Block MASS      #  Mass spectrum (kinematic masses)
#       PDG       Mass
        5     4.70000000E+00   # bottom   pole mass
        6     1.72000000E+02   # top      pole mass

DECAY       6     1.43947825E+00   # top width
DECAY      23     2.44140351E+00   # Z   width

#       BR          NDA       ID1       ID2
    8.27451012E-02   2          4        -4   # BR( H -> c  cbar  )
    7.17809696E-01   2          5        -5   # BR( H -> b  bbar  )
```

The param_card

```
Block SMINPUTS      # Standard Model inputs
    1        1.32506980E+02   # alpha_em(MZ)(-1) SM MSbar
    2        1.1663900
    3        1.18000000E-01   # alpha_s(MZ) SM MSbar
    4        9.11880000E+01   # Z mass (as input parameter)
```

Model Parameters

```
Block MASS      #  Mass spectrum (kinematic masses)
#       PDG       Mass
        5     4.7000000                   pole mass
        6     1.72000000E+02   # top      pole mass
```

Masses

```
DECAY        6     1.43947825E+00   # top width
DECAY        23     2.44140351E+00   # Z   width

#       BR           NDA       ID1       ID2
    8.27451012E-02    2          4         -4    # BR( H -> c  cbar  )
    7.17809696E-01    2          5         -5    # BR( H -> b  bbar  )
```

The param_card

```
Block SMINPUTS      # Standard Model inputs
    1        1.32506980E+02  # alpha_em(MZ)(-1) SM MSbar
    2        1.166390
    3        1.18000000E-01  # alpha_s(MZ) SM MSbar
    4        9.11880000E+01  # Z mass (as input parameter)
```

Model Parameters

```
Block MASS      #  Mass spectrum (kinematic masses)
#       PDG       Mass
        5     4.700000             pole mass
        6     1.72000000E+02  # top      pole mass
```

Masses

```
DECAY       6     1.43                          dth
DECAY       23    2.4                           dth
```

Widths

```
#       BR          NDA       ID1       ID2
    8.27451012E-02   2          4        -4   # BR( H -> c  cbar  )
    7.17809696E-01   2          5        -5   # BR( H -> b  bbar  )
```

The param_card

```
Block SMINPUTS      # Standard Model inputs
    1         1.32506980E+02   # alpha_em(MZ)(-1) SM MSbar
    2         1.166390
    3         1.18000000E-01   # alpha_s(MZ) SM MSbar
    4         9.11880000E+01   # Z mass (as input parameter)
```

Model Parameters

```
Block MASS      #  Mass spectrum (kinematic masses)
#       PDG       Mass
        5     4.7000000                       pole mass
        6     1.72000000E+02   # top      pole mass
```

Masses

```
DECAY       6     1.43                        dth
DECAY       23    2.4                         dth
```

Widths

```
#       BR          NDA       ID1      ID2
    8.27451012E-02                            H -> c  cbar  )
    7.17809696E-01                            H -> b  bbar  )
```

Branching Ratios

The param_card

- The param_card consists of 4 parts:
  - ➡ Model parameters (grouped into blocks)
  - ➡ Masses
  - ➡ Widths (given as numerical inputs)
  - ➡ Branching ratios

The param_card

- The param_card consists of 4 parts:
  - ➡ Model parameters (grouped into blocks)
  - ➡ Masses
  - ➡ Widths (given as numerical inputs)
  - ➡ Branching ratios

```
aS = Parameter(name = 'aS',
          nature = 'external',
          type = 'real',
          value = 0.118,
          texname = '\\text{aS}',
          lhablock = 'SMINPUTS',
          lhacode = [ 3 ])
```

The param_card

- The param_card consists of 4 parts:
  - ➡ Model parameters (grouped into blocks)
  - ➡ Masses
  - ➡ Widths (given as numerical inputs)
  - ➡ Branching ratios
- Numerical values for the width can be given as inputs in FeynRules, and are then passed to the UFO.

The param_card

- The param_card consists of 4 parts:
  - ➡ Model parameters (grouped into blocks)
  - ➡ Masses
  - ➡ Widths (given as numerical inputs)
  - ➡ Branching ratios
- Numerical values for the width can be given as inputs in FeynRules, and are then passed to the UFO.
- However, FeynRules cannot compute the widths by itself.

The param_card

- The param_card consists of 4 parts:
  - ➡ Model parameters (grouped into blocks)
  - ➡ Masses
  - ➡ Widths (given as numerical inputs)
  - ➡ Branching ratios
- Numerical values for the width can be given as inputs in FeynRules, and are then passed to the UFO.
- However, FeynRules cannot compute the widths by itself.
  - ➡ Write out the UFO model without the widths, and use MadGraph to compute them.

The param_card

- The param_card consists of 4 parts:
  - ➡ Model parameters (grouped into blocks)
  - ➡ Masses
  - ➡ Widths (given as numerical inputs)
  - ➡ Branching ratios
- Numerical values for the width can be given as inputs in FeynRules, and are then passed to the UFO.
- However, FeynRules cannot compute the widths by itself.
  - ➡ Write out the UFO model without the widths, and use MadGraph to compute them.
- Branching ratios cannot be included into FeynRules of the UFO right now.

The decay module

- In the future, FeynRules will be able to compute all the two-body decays, and will pass them on to the UFO.

The decay module

- In the future, FeynRules will be able to compute all the two-body decays, and will pass them on to the UFO.

$$\Gamma_{1\to 2} = \frac{1}{2m}\int d\Phi_2\, |\mathcal{M}_{1\to 2}|^2$$

The decay module

- In the future, FeynRules will be able to compute all the two-body decays, and will pass them on to the UFO.

$$\Gamma_{1\to 2} = \frac{1}{2m}\int d\Phi_2\, |\mathcal{M}_{1\to 2}|^2$$
$$= \frac{1}{2m}|\mathcal{M}_{1\to 2}|^2\,\mathrm{Vol(phase\ space)}$$

The decay module

- In the future, FeynRules will be able to compute all the two-body decays, and will pass them on to the UFO.

$$\Gamma_{1\to 2} = \frac{1}{2m}\int d\Phi_2\,|\mathcal{M}_{1\to 2}|^2$$
$$= \frac{1}{2m}|\mathcal{M}_{1\to 2}|^2\,\mathrm{Vol(phase\ space)}$$

- Or in words: a two-body decay rate is just a constant: the square of a three-point vertex times the PS volume.

The decay module

- In the future, FeynRules will be able to compute all the two-body decays, and will pass them on to the UFO.

$$\Gamma_{1\to 2} = \frac{1}{2m}\int d\Phi_2\,|\mathcal{M}_{1\to 2}|^2$$
$$= \frac{1}{2m}|\mathcal{M}_{1\to 2}|^2\,\mathrm{Vol(phase\ space)}$$

- Or in words: a two-body decay rate is just a constant: the square of a three-point vertex times the PS volume.
- In FeynRules, we have
  - ➡ all the three-point vertices,
  - ➡ a high-level computer algebra system.

The decay module

- In the future, FeynRules will be able to compute all the two-body decays, and will pass them on to the UFO.

$$\Gamma_{1\to2} = \frac{1}{2m}\int d\Phi_2\,|\mathcal{M}_{1\to2}|^2$$
$$= \frac{1}{2m}|\mathcal{M}_{1\to2}|^2\,\mathrm{Vol(phase\ space)}$$

- Or in words: a two-body decay rate is just a constant: the square of a three-point vertex times the PS volume.
- In FeynRules, we have
  - ➡ all the three-point vertices,
  - ➡ a high-level computer algebra system.
- That's all we need to get the two-body decays!

The decay module

- The development version of FeynRules already computes the widths and writes them to the UFO files.
- Currently being implemented into MadGraph 5.

The decay module

- The development version of FeynRules already computes the widths and writes them to the UFO files.
- Currently being implemented into MadGraph 5.

```
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'})
```

The decay module

- The development version of FeynRules already computes the widths and writes them to the UFO files.
- Currently being implemented into MadGraph 5.

```
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'})
```

- The width and the branching ratios will then be included automatically computed when loading a UFO.

Inside the UFO

```
__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py
```

- These files are the core of the UFO.
- The contain the definition of the vertices and the couplings.
- The rest of this lecture will be about these files.

Vertices in MadGraph

- In MadGraph a vertex is represented by a threefold data:
  - ➡ a color structure
  - ➡ a Lorentz structure
  - ➡ a coupling constant (a complex number)
- The computation of the Lorentz structures is outsourced to the Helas library, which allows to compute helicity amplitudes in a fast and efficient way.

[Murayama, Hagiwara, Watanabe]

Vertices in MadGraph

- In MadGraph a vertex is represented by a threefold data:
  - ➡ a color structure
  - ➡ a Lorentz structure
  - ➡ a coupling constant (a complex number)
- The computation of the Lorentz structures is outsourced to the Helas library, which allows to compute helicity amplitudes in a fast and efficient way.

[Murayama, Hagiwara, Watanabe]

- The master formula:

$$\mathcal{V}^{a_1\ldots a_n,\ell_1\ldots\ell_n}(p_1,\ldots,p_n) = \sum_{i,j} C_i^{a_1\ldots a_n}\, G_{ij}\, L_j^{\ell_1\ldots\ell_n}(p_1,\ldots,p_n)$$

Vertices in MadGraph

- In MadGraph a vertex is represented by a threefold data:
  - ➡ a color structure
  - ➡ a Lorentz structure
  - ➡ a coupling constant (a complex number)
- The computation of the Lorentz structures is outsourced to the Helas library, which allows to compute helicity amplitudes in a fast and efficient way.

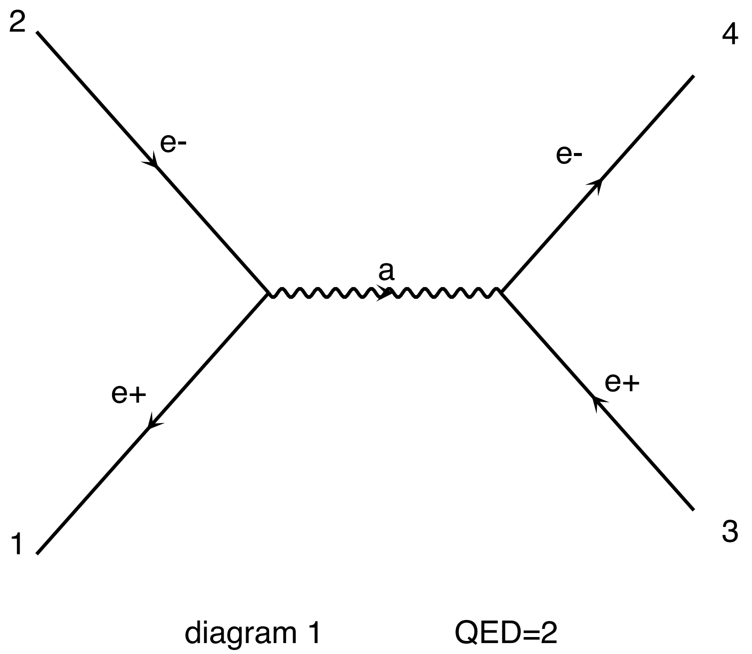
[Murayama, Hagiwara, Watanabe]

- The master formula:

$$\mathcal{V}^{a_1\ldots a_n,\ell_1\ldots\ell_n}(p_1,\ldots,p_n) = \sum_{i,j} \boxed{C_i^{a_1\ldots a_n}}\,\boxed{G_{ij}}\,\boxed{L_j^{\ell_1\ldots\ell_n}(p_1,\ldots,p_n)}$$

HELAS [Murayama, Hagiwara, Watanabe]

- **Idea:** Evaluate the matrix element for fixed helicities of the external particles

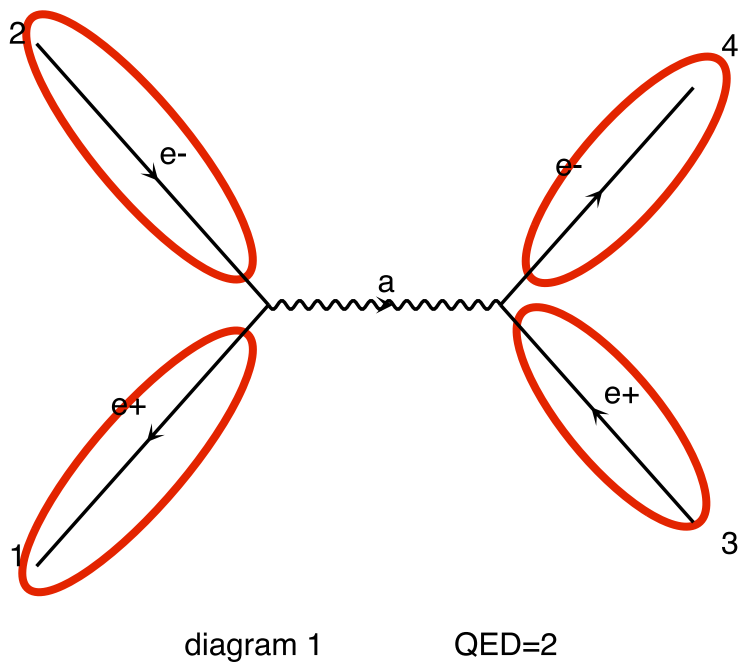


$$M = \bar{u}\gamma^{\mu}v \; P_{\mu\nu} \; \bar{u}\gamma^{\nu}v$$

[Slide from O. Mattelaer]

HELAS [Murayama, Hagiwara, Watanabe]

- **Idea:** Evaluate the matrix element for fixed helicities of the external particles



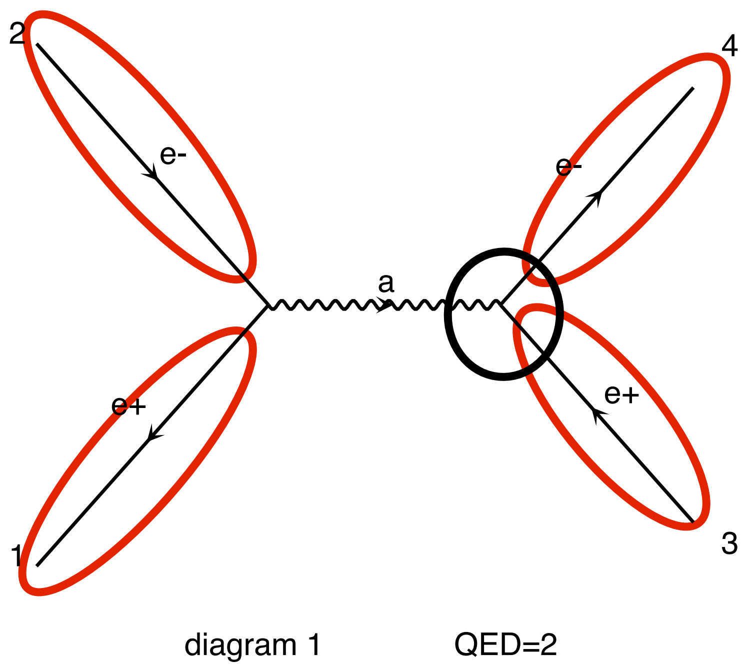
$$M = \bar{u}\gamma^{\mu}v\,P_{\mu\nu}\,\bar{u}\gamma^{\nu}v$$

→ Number for a given helicity

[Slide from O. Mattelaer]

HELAS [Murayama, Hagiwara, Watanabe]

- **Idea:** Evaluate the matrix element for fixed helicities of the external particles



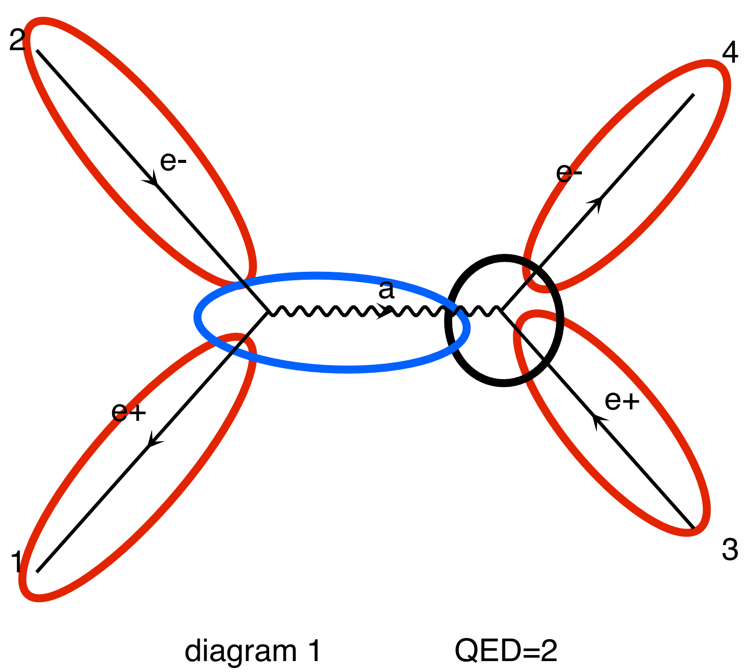
$$M = \bar{u}\gamma^\mu v \, P_{\mu\nu} \, \bar{u}\gamma^\nu v$$

→ Number for a given helicity

[Slide from O. Mattelaer]

HELAS [Murayama, Hagiwara, Watanabe]

- **Idea:** Evaluate the matrix element for fixed helicities of the external particles



$$M = \bar{u}\gamma^{\mu} v \, P_{\mu\nu} \, \bar{u}\gamma^{\nu} v$$

→ Number for a given helicity

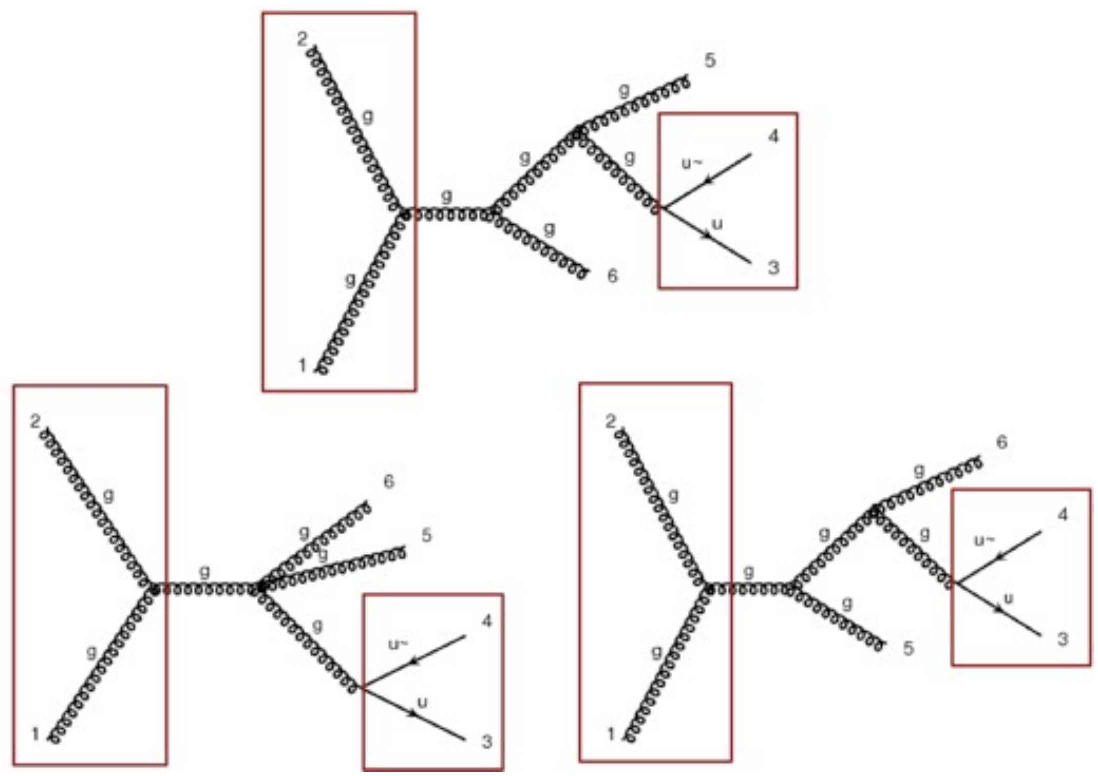
→ Evaluate interaction by interaction

```
CALL IXXXXX(P(0,1),ZERO,NHEL(1),+1*IC(1),W(1,1))
CALL OXXXXX(P(0,2),ZERO,NHEL(2),-1*IC(2),W(1,2))
CALL OXXXXX(P(0,3),MT,NHEL(3),+1*IC(3),W(1,3))
CALL IXXXXX(P(0,4),MT,NHEL(4),-1*IC(4),W(1,4))
CALL JIOXXX(W(1,1),W(1,2),GG,ZERO,ZERO,W(1,5))
CALL IOVXXX(W(1,4),W(1,3),W(1,5),GG,AMP(1))
```

[Slide from O. Mattelaer]

HELAS [Murayama, Hagiwara, Watanabe]

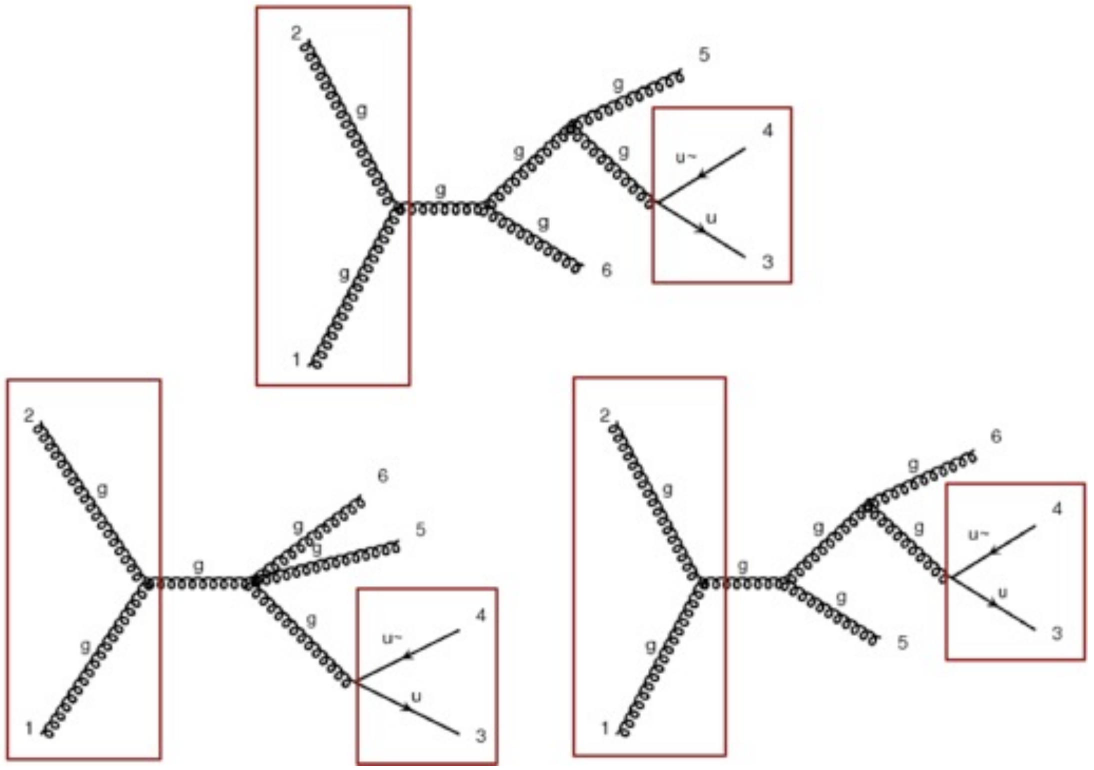
- Speed:



[Slide from O. Mattelaer]

HELAS [Murayama, Hagiwara, Watanabe]

- Speed:
    - ➡ The complexity grows linearly with the number of diagrams.
    - ➡ Recycling of diagrams: reduces the factorial growth.



[Slide from O. Mattelaer]

Vertices in MadGraph

- The master formula:

$$\mathcal{V}^{a_1\ldots a_n,\ell_1\ldots\ell_n}(p_1,\ldots,p_n) = \sum_{i,j} C_i^{a_1\ldots a_n}\, G_{ij}\, L_j^{\ell_1\ldots\ell_n}(p_1,\ldots,p_n)$$

- Example:

$$\begin{aligned} & ig_s^2\, f^{a_1a_2b} f^{ba_3a_4}\, \left(\eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4}\right) \\ & + ig_s^2\, f^{a_1a_3b} f^{ba_2a_4}\, \left(\eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4}\right) \\ & + ig_s^2\, f^{a_1a_4b} f^{ba_2a_3}\, \left(\eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4}\right) \end{aligned}$$

$$\left(f^{a_1a_2b} f^{ba_3a_4},\ f^{a_1a_3b} f^{ba_2a_4},\ f^{a_1a_4b} f^{ba_2a_3}\right) \begin{pmatrix} ig_s^2 & 0 & 0 \\ 0 & ig_s^2 & 0 \\ 0 & 0 & ig_s^2 \end{pmatrix} \begin{pmatrix} \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} \\ \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \\ \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \end{pmatrix}$$

Vertices in the UFO

$$\left(f^{a_1 a_2 b} f^{b a_3 a_4},\; f^{a_1 a_3 b} f^{b a_2 a_4},\; f^{a_1 a_4 b} f^{b a_2 a_3}\right) \begin{pmatrix} i g_s^2 & 0 & 0 \\ 0 & i g_s^2 & 0 \\ 0 & 0 & i g_s^2 \end{pmatrix} \begin{pmatrix} \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} \\ \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \\ \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \end{pmatrix}$$

- Vertices are saved in vertices.py in precisely this way:

```
V_5 = Vertex(name = 'V_5',
          particles = [ P.g, P.g, P.g, P.g ],
          color = [ 'f(-1,1,2)*f(3,4,-1)', 'f(-1,1,3)*f(2,4,-1)', 'f(-1,1,4)*f(2,3,-1)' ],
          lorentz = [ L.VVVV6, L.VVVV8, L.VVVV9 ],
          couplings = {(1,1):C.GC_8,(0,0):C.GC_8,(2,2):C.GC_8})
```

vertices.py

Vertices in the UFO

- Allowed color building blocks:

Trivial tensor (for non-colored particles)	`1`
Kronecker delta $\delta^{\bar{\jmath}_2}{}_{i_1}$	`Identity(1,2)`
Fundamental representation matrices $(T^{a_1})^{\bar{\jmath}_3}{}_{i_2}$	`T(1,2,3)`
Structure constants $f^{a_1 a_2 a_3}$	`f(1,2,3)`
Symmetric tensor $d^{a_1 a_2 a_3}$	`d(1,2,3)`
Fundamental Levi-Civita tensor $\epsilon_{i_1 i_2 i_3}$	`Epsilon(1,2,3)`
Antifundamental Levi-Civita tensor $\epsilon^{\bar{\imath}_1 \bar{\imath}_2 \bar{\imath}_3}$	`EpsilonBar(1,2,3)`
Sextet representation matrices $(T_6^{a_1})^{\bar{\beta}_3}{}_{\alpha_2}$	`T6(1,2,3)`
Sextet Clebsch-Gordan coefficient $(K_6)^{\bar{\imath}_2 \bar{\jmath}_3}{}_{\alpha_1}$	`K6(1,2,3)`
Antisextet Clebsch-Gordan coefficient $(\bar{K}_6)^{\bar{\alpha}_1}{}_{i_2 j_3}$	`K6Bar(1,2,3)`

Vertices in the UFO

$$\left(f^{a_1a_2b}f^{ba_3a_4},\ f^{a_1a_3b}f^{ba_2a_4}, f^{a_1a_4b}f^{ba_2a_3}\right)\begin{pmatrix} ig_s^2 & 0 & 0 \\ 0 & ig_s^2 & 0 \\ 0 & 0 & ig_s^2 \end{pmatrix}\begin{pmatrix} \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} \\ \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \\ \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \end{pmatrix}$$

- Vertices are saved in vertices.py in precisely this way:

```
V_5 = Vertex(name = 'V_5',
          particles = [ P.g, P.g, P.g, P.g ],
          color = [ 'f(-1,1,2)*f(3,4,-1)', 'f(-1,1,3)*f(2,4,-1)', 'f(-1,1,4)*f(2,3,-1)' ],
          lorentz = [ L.VVVV6, L.VVVV8, L.VVVV9 ],
          couplings = {(1,1):C.GC_8,(0,0):C.GC_8,(2,2):C.GC_8})
```

vertices.py

Vertices in the UFO

$$\left(f^{a_1a_2b}f^{ba_3a_4},\ f^{a_1a_3b}f^{ba_2a_4}, f^{a_1a_4b}f^{ba_2a_3}\right)\begin{pmatrix} ig_s^2 & 0 & 0 \\ 0 & ig_s^2 & 0 \\ 0 & 0 & ig_s^2 \end{pmatrix}\begin{pmatrix} \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} \\ \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \\ \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \end{pmatrix}$$

- Vertices are saved in vertices.py in precisely this way:

```
V_5 = Vertex(name = 'V_5',
        particles = [ P.g, P.g, P.g, P.g ],
        color = [ 'f(-1,1,2)*f(3,4,-1)', 'f(-1,1,3)*f(2,4,-1)', 'f(-1,1,4)*f(2,3,-1)' ],
        lorentz = [ L.VVVV6, L.VVVV8, L.VVVV9 ],
        couplings = {(1,1):C.GC_8,(0,0):C.GC_8,(2,2):C.GC_8})
```

vertices.py

```
VVVV6 = Lorentz(name = 'VVVV6',
        spins = [ 3, 3, 3, 3 ],
        structure = 'Metric(1,4)*Metric(2,3) -
                     Metric(1,3)*Metric(2,4)')
```

lorentz.py

Vertices in the UFO

- Allowed Lorentz structure building blocks:

Charge conjugation matrix: $C_{i_1 i_2}$	`C(1,2)`
Epsilon matrix: $\epsilon^{\mu_1\mu_2\mu_3\mu_4}$	`Epsilon(1,2,3,4)`
Dirac matrices: $(\gamma^{\mu_1})_{i_2 i_3}$	`Gamma(1, 2, 3)`
Fifth Dirac matrix: $(\gamma^5)_{i_1 i_2}$	`Gamma5(1,2)`
(Spinorial) Kronecker delta: $\delta_{i_1 i_2}$	`Identity(1,2)`
Minkowski metric: $\eta_{\mu_1 \mu_2}$	`Metric(1,2)`
Momentum of the N^{th} particle: $p_N^{\mu_1}$	`P(1,N)`
Right-handed chiral projector: $\left(\frac{1+\gamma 5}{2}\right)_{i_1 i_2}$	`ProjP(1,2)`
Left-handed chiral projector $\left(\frac{1-\gamma 5}{2}\right)_{i_1 i_2}$	`ProjM(1,2)`
Sigma matrices: $(\sigma^{\mu_1\mu_2})_{i_3 i_4}$	`Sigma(1,2,3,4)`

Vertices in the UFO

$$\left(f^{a_1 a_2 b} f^{b a_3 a_4},\ f^{a_1 a_3 b} f^{b a_2 a_4}, f^{a_1 a_4 b} f^{b a_2 a_3} \right) \begin{pmatrix} i g_s^2 & 0 & 0 \\ 0 & i g_s^2 & 0 \\ 0 & 0 & i g_s^2 \end{pmatrix} \begin{pmatrix} \eta^{\mu_1 \mu_4} \eta^{\mu_2 \mu_3} - \eta^{\mu_1 \mu_3} \eta^{\mu_2 \mu_4} \\ \eta^{\mu_1 \mu_4} \eta^{\mu_2 \mu_3} - \eta^{\mu_1 \mu_2} \eta^{\mu_3 \mu_4} \\ \eta^{\mu_1 \mu_3} \eta^{\mu_2 \mu_4} - \eta^{\mu_1 \mu_2} \eta^{\mu_3 \mu_4} \end{pmatrix}$$

- Vertices are saved in vertices.py in precisely this way:

```
V_5 = Vertex(name = 'V_5',
        particles = [ P.g, P.g, P.g, P.g ],
        color = [ 'f(-1,1,2)*f(3,4,-1)', 'f(-1,1,3)*f(2,4,-1)', 'f(-1,1,4)*f(2,3,-1)' ],
        lorentz = [ L.VVVV6, L.VVVV8, L.VVVV9 ],
        couplings = {(1,1):C.GC_8,(0,0):C.GC_8,(2,2):C.GC_8})
```

vertices.py

```
VVVV6 = Lorentz(name = 'VVVV6',
        spins = [ 3, 3, 3, 3 ],
        structure = 'Metric(1,4)*Metric(2,3) -
                     Metric(1,3)*Metric(2,4)')
```

lorentz.py

Vertices in the UFO

$$\left(f^{a_1a_2b}f^{ba_3a_4},\ \ f^{a_1a_3b}f^{ba_2a_4},\ f^{a_1a_4b}f^{ba_2a_3}\right)\begin{pmatrix} ig_s^2 & 0 & 0 \\ 0 & ig_s^2 & 0 \\ 0 & 0 & ig_s^2 \end{pmatrix}\begin{pmatrix} \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} \\ \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \\ \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \end{pmatrix}$$

- Vertices are saved in vertices.py in precisely this way:

```
V_5 = Vertex(name = 'V_5',
         particles = [ P.g, P.g, P.g, P.g ],
         color = [ 'f(-1,1,2)*f(3,4,-1)', 'f(-1,1,3)*f(2,4,-1)', 'f(-1,1,4)*f(2,3,-1)' ],
         lorentz = [ L.VVVV6, L.VVVV8, L.VVVV9 ],
         couplings = {(1,1):C.GC_8,(0,0):C.GC_8,(2,2):C.GC_8})
```

vertices.py

```
VVVV6 = Lorentz(name = 'VVVV6',
         spins = [ 3, 3, 3, 3 ],
         structure = 'Metric(1,4)*Metric(2,3) -
                      Metric(1,3)*Metric(2,4)')
```

lorentz.py

```
GC_8 = Coupling(name = 'GC_8',
         value = 'complex(0,1)*G**2',
```

couplings.py

Inside the UFO

__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py

- To understand the content of this file we first need to understand interaction orders.
- MadGraph has the to 'count' the number of couplings of a certain type that enter a diagram.
- Example: In the SM, there are two types of couplings (=interaction orders): QED and QCD.

➡ p p > t t~, QCD only: QED = 0, QCD = 2

➡ p p > t t~, EW only: QED = 2, QCD = 0

➡ p p > t t~, all: QED = 2, QCD = 2

Inside the UFO

__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py

Inside the UFO

```
__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py
```

- Sometimes it can be useful to define a default behavior for interaction orders.

```
QCD = CouplingOrder(name = 'QCD',
                    expansion_order = 99,
                    hierarchy = 1)

QED = CouplingOrder(name = 'QED',
                    expansion_order = 99,
                    hierarchy = 2)
```

Inside the UFO

__init__.py
object_library.py
write_param_card.py
function_library.py
particles.py
parameters.py
vertices.py
lorentz.py
couplings.py
coupling_orders.py

- Sometimes it can be useful to define a default behavior for interaction orders.

```
QCD = CouplingOrder(name = 'QCD',
                    expansion_order = 99,
                    hierarchy = 1)

QED = CouplingOrder(name = 'QED',
                    expansion_order = 99,
                    hierarchy = 2)
```

- Default interaction orders can be defined in the FeynRules model file:

```
M$InteractionOrderHierarchy = {
        {QCD, 1},
        {QED, 2}};

M$InteractionOrderLimit = {
        {QCD, 99},
        {QED, 99}};
```

Summary

- The UFO is the default model format of MadGraph 5.

- A UFO is a selfcontained Python model, and stored all the model information in the form of Python modules.

- The structure of the UFO is such that, unlike the traditional text-based model format, it allows to accommodate very large classes of models.

- Next: How is this information passed to MadGraph..?

From UFO to MadGraph

From UFO to MadGraph

- Representation of a vertex in the UFO:

$$\sum_{i,j} C_i^{a_1\ldots a_n}\, G_{ij}\, L_j^{\ell_1\ldots\ell_n}(p_1,\ldots,p_n)$$

```
VVVV6 = Lorentz(name = 'VVVV6',
                spins = [ 3, 3, 3, 3 ],
                structure = 'Metric(1,4)*Metric(2,3) -Metric(1,3)*Metric(2,4)')
```

From UFO to MadGraph

- Representation of a vertex in the UFO:

$$\sum_{i,j} C_i^{a_1\dots a_n}\, G_{ij}\, L_j^{\ell_1\dots\ell_n}(p_1,\dots,p_n)$$

```
VVVV6 = Lorentz(name = 'VVVV6',
                spins = [ 3, 3, 3, 3 ],
                structure = 'Metric(1,4)*Metric(2,3) -Metric(1,3)*Metric(2,4)')
```

- Representation of a vertex in the MadGraph: Helas routine

```
 VERTEX = COUP*( (V4(1)*( (V2(1)*( (0, -1)*(V3(2)*V1(2))
$ +(0, -1)*(V3(3)*V1(3))+(0, -1)*(V3(4)*V1(4))))+(V1(1)*( (0, 1)
$ *(V3(2)*V2(2))+(0, 1)*(V3(3)*V2(3))+(0, 1)*(V3(4)*V2(4))))))
$ +( (V4(2)*( (V2(2)*( (0, -1)*(V3(1)*V1(1))+(0, 1)*(V3(3)*V1(3))
$ +(0, 1)*(V3(4)*V1(4))))+(V1(2)*( (0, 1)*(V3(1)*V2(1))+(0,
$ -1)*(V3(3)*V2(3))+(0, -1)*(V3(4)*V2(4))))))+( (V4(3)*( (V2(3)
$ *( (0, -1)*(V3(1)*V1(1))+(0, 1)*(V3(2)*V1(2))+(0, 1)*(V3(4)
$ *V1(4))))+(V1(3)*( (0, 1)*(V3(1)*V2(1))+(0, -1)*(V3(2)*V2(2))
$ +(0, -1)*(V3(4)*V2(4))))))+(V4(4)*( (V2(4)*( (0, -1)*(V3(1)
$ *V1(1))+(0, 1)*(V3(2)*V1(2))+(0, 1)*(V3(3)*V1(3))))+(V1(4)
$ *( (0, 1)*(V3(1)*V2(1))+(0, -1)*(V3(2)*V2(2))+(0, -1)*(V3(3)
$ *V2(3))))))))
 END
```

From UFO to MadGraph

- Representation of a vertex in the UFO:

$$\sum_{i,j} C_i^{a_1\ldots a_n}\, G_{ij}\, L_j^{\ell_1\ldots\ell_n}(p_1,\ldots,p_n)$$

```
VVVV6 = Lorentz(name = 'VVVV6',
                spins = [ 3, 3, 3, 3 ],
                structure = 'Metric(1,4)*Metric(2,3) -Metric(1,3)*Metric(2,4)')
```

- Representation of a vertex in the MadGraph: Helas routine

```
 VERTEX = COUP*( (V4(1)*( (V2(1)*( (0, -1)*(V3(2)*V1(2))
$ +(0, -1)*(V3(3)*V1(3))+(0, -1)*(V3(4)*V1(4))))+(V1(1)*( (0, 1)
$ *(V3(2)*V2(2))+(0, 1)*(V3(3)*V2(3))+(0, 1)*(V3(4)*V2(4))))))
$ +( (V4(2)*( (V2(2)*( (0, -1)*(V3(1)*V1(1))+(0, 1)*(V3(3)*V1(3))
$ +(0, 1)*(V3(4)*V1(4))))+(V1(2)*( (0, 1)*(V3(1)*V2(1))+(0,
$ -1)*(V3(3)*V2(3))+(0, -1)*(V3(4)*V2(4))))))+( (V4(3)*( (V2(3)
$ *( (0, -1)*(V3(1)*V1(1))+(0, 1)*(V3(2)*V1(2))+(0, 1)*(V3(4)
$ *V1(4))))+(V1(3)*( (0, 1)*(V3(1)*V2(1))+(0, -1)*(V3(2)*V2(2))
$ +(0, -1)*(V3(4)*V2(4))))))+(V4(4)*( (V2(4)*( (0, -1)*(V3(1)
$ *V1(1))+(0, 1)*(V3(2)*V1(2))+(0, 1)*(V3(3)*V1(3))))+(V1(4)
$ *( (0, 1)*(V3(1)*V2(1))+(0, -1)*(V3(2)*V2(2))+(0, -1)*(V3(3)
$ *V2(3)))))))))
 END
```

- Want to combine the flexibility of the UFO with the efficiency of Helas.
- We need a translater UFO-Helas.

The UFO & ALOHA

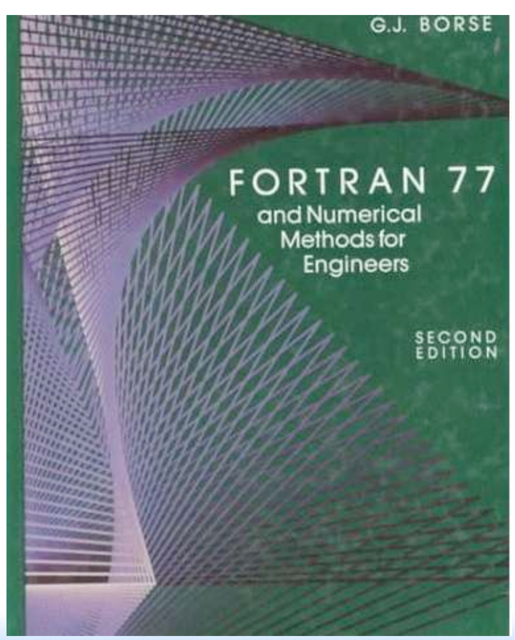
- The development of the UFO goes hand in hand with the development of ALOHA.
- ALOHA = Automatic Language-independent Output of Helicity Amplitudes.
- Idea: ALOHA uses the information contained in the UFO to create the library of Lorentz structures for MadGraph on the fly.

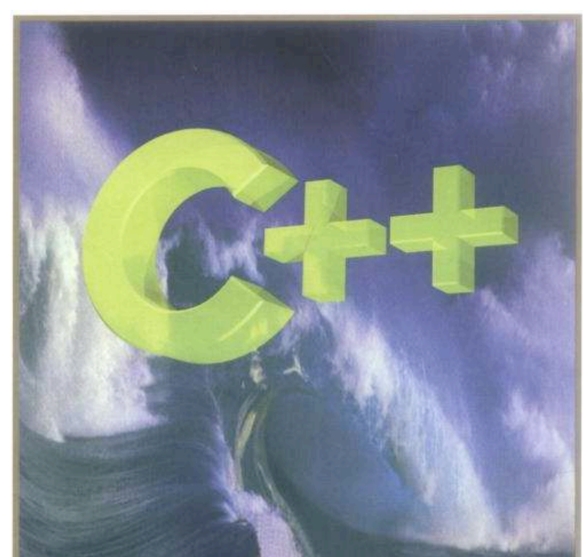
ALOHA

ALOHA
~~Google~~ translate

From: UFO To: Helicity Translate

Type text or a website address or translate a document.





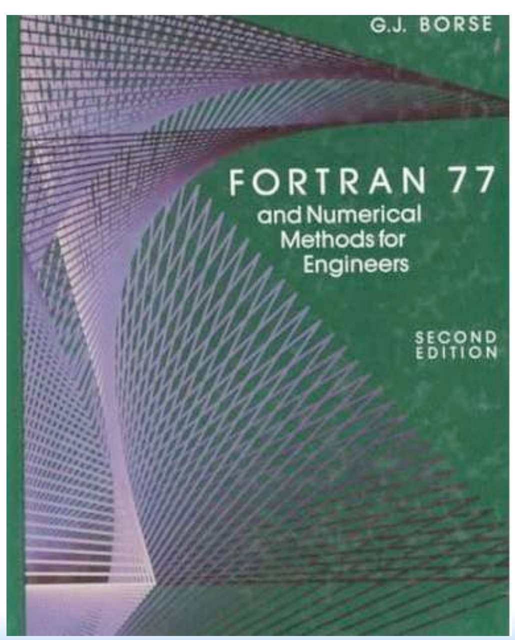
ALOHA

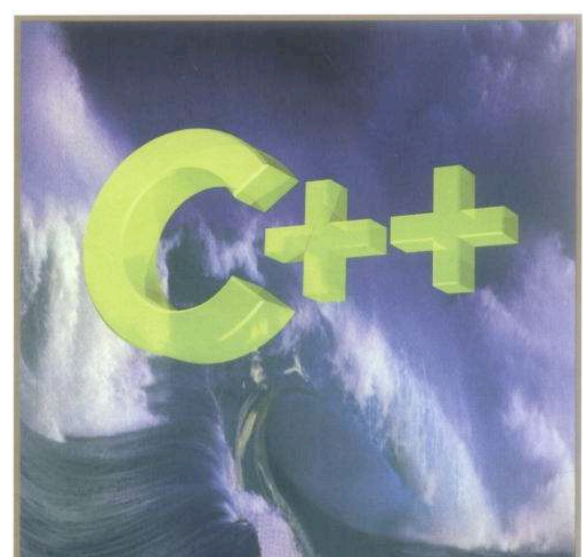
ALOHA
~~Google~~ translate

From: UFO To: Helicity Translate

```
VVVV6 = Lorentz(name = 'VVVV6',
            spins = [ 3, 3, 3, 3 ],
            structure = 'Metric(1,4)*Metric(2,3) -Metric(1,3)*Metric(2,4)')
```

Type text or a website address or translate a document.





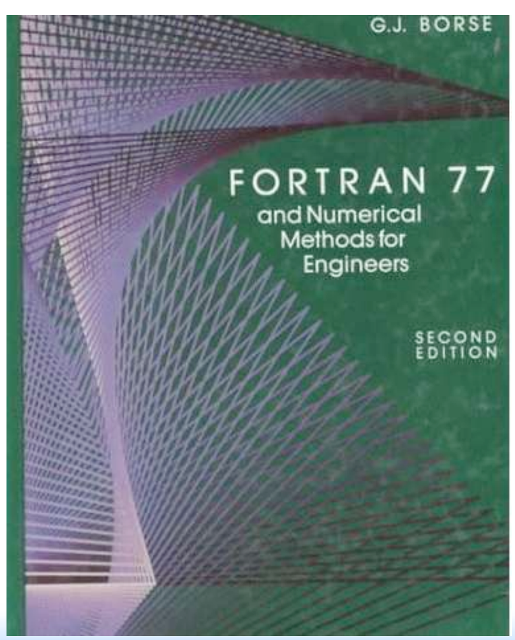
ALOHA

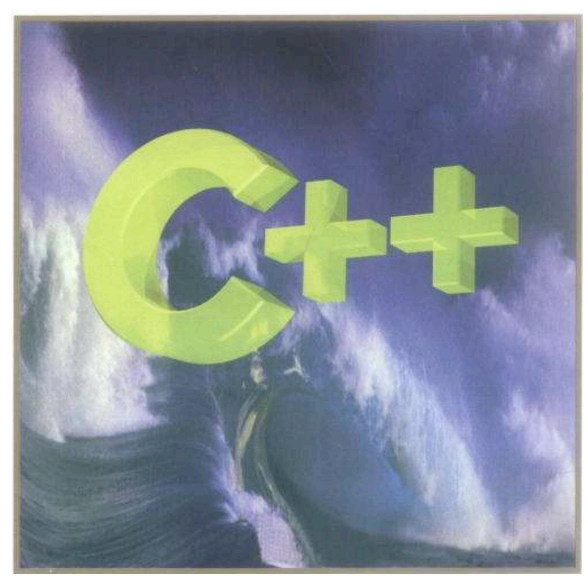
ALOHA ~~Google~~ translate

From: UFO   To: Helicity   Translate

```
 VERTEX = COUP*( (V4(1)*( (V2(1)*( (0, -1)*(V3(2)*V1(2))
$ +(0, -1)*(V3(3)*V1(3))+(0, -1)*(V3(4)*V1(4))))+(V1(1)*( (0, 1)
$ *(V3(2)*V2(2))+(0, 1)*(V3(3)*V2(3))+(0, 1)*(V3(4)*V2(4))))))
$ +( (V4(2)*( (V2(2)*( (0, -1)*(V3(1)*V1(1))+(0, 1)*(V3(3)*V1(3))
$ +(0, 1)*(V3(4)*V1(4))))+(V1(2)*( (0, 1)*(V3(1)*V2(1))+(0,
$ -1)*(V3(3)*V2(3))+(0, -1)*(V3(4)*V2(4))))))+( (V4(3)*( (V2(3)
```

Type text or a website address or translate a document.





News on ALOHA

- ALOHA is optimizing the way to do analytic computations:
  - ➡ SM computed in 1.2s (instead of 3s).
  - ➡ MSSM computed in 1.4s (instead of 5s).
  - ➡ Randall-Sundrum computed in 90s (instead of 15mins).
- More optimized output:
  - ➡ Faster to compile.
  - ➡ Faster to run (up to 40% faster).
- Possibility to create ALOHA routines from the MG5 shell:

```
mg5> output aloha FFV1_3
```

[Slide from O. Mattelaer]

News on ALOHA

- New Outputs/Options in progress:
  - ➡ Spin 3/2.
  - ➡ Feynman gauge.
  - ➡ Complex mass scheme.
  - ➡ Quadruple precision.
  - ➡ Open Loops.

[Slide from O. Mattelaer]

O
L
D

A
L
O
H
A

O
U
T
P
U
T

```fortran
C     This File is Automatically generated by ALOHA
C     The process calculated in this file is:
C     Gamma(3,2,1)
C
      SUBROUTINE FFV1_3(F1, F2, COUP, M3, W3, V3)
      IMPLICIT NONE
      DOUBLE COMPLEX F1(*)
      DOUBLE COMPLEX F2(*)
      DOUBLE COMPLEX V3(*)
      DOUBLE COMPLEX COUP
      DOUBLE COMPLEX DENOM
      DOUBLE PRECISION M3, W3
      DOUBLE COMPLEX OM3
      DOUBLE PRECISION P3(0:3)

      V3(5)= -F1(5)+F2(5)
      V3(6)= -F1(6)+F2(6)
      P3(0) = - DBLE(V3(5))
      P3(1) = - DBLE(V3(6))
      P3(2) = - DIMAG(V3(6))
      P3(3) = - DIMAG(V3(5))
      OM3 = 0D0
      IF (M3 .NE. 0D0) OM3=1D0/M3**2

      DENOM =1D0/(( (M3*( -M3+(0, 1)*W3))+( (P3(0)**2)-(P3(1)**2)
     $ -(P3(2)**2)-(P3(3)**2))))
      V3(1)= COUP*DENOM*( (OM3*( (F1(1)*( (F2(4)*( (0, 1)*P3(1)
     $ -P3(2)))+(F2(3)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))+( (F1(2)
     $ *( (F2(3)*( (0, 1)*P3(1)+P3(2)))+(F2(4)*( (0, 1)*P3(0)
     $ +(0, -1)*P3(3)))))+( (F1(3)*( (F2(2)*( (0, -1)*P3(1)+P3(2)))
     $ +(F2(1)*( (0, 1)*P3(0)+(0, -1)*P3(3)))))+(F1(4)*( (F2(1)*( (0,
     $ -1)*P3(1)-P3(2)))+(F2(2)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))))))
     $ *P3(0))+( (0, -1)*(F2(3)*F1(1))+(0, -1)*(F2(4)*F1(2))+(0,
     $ -1)*(F2(1)*F1(3))+(0, -1)*(F2(2)*F1(4))))
      V3(2)= COUP*DENOM*( (OM3*( (F1(1)*( (F2(4)*( (0, 1)*P3(1)
     $ -P3(2)))+(F2(3)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))+( (F1(2)
     $ *( (F2(3)*( (0, 1)*P3(1)+P3(2)))+(F2(4)*( (0, 1)*P3(0)
     $ +(0, -1)*P3(3)))))+( (F1(3)*( (F2(2)*( (0, -1)*P3(1)+P3(2)))
     $ +(F2(1)*( (0, 1)*P3(0)+(0, -1)*P3(3)))))+(F1(4)*( (F2(1)*( (0,
     $ -1)*P3(1)-P3(2)))+(F2(2)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))))))
     $ *P3(1))+( (0, 1)*(F2(4)*F1(1))+(0, 1)*(F2(3)*F1(2))+(0,
     $ -1)*(F2(2)*F1(3))+(0, -1)*(F2(1)*F1(4))))
      V3(3)= COUP*DENOM*( (OM3*( (F1(1)*( (F2(4)*( (0, 1)*P3(1)
     $ -P3(2)))+(F2(3)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))+( (F1(2)
     $ *( (F2(3)*( (0, 1)*P3(1)+P3(2)))+(F2(4)*( (0, 1)*P3(0)
     $ +(0, -1)*P3(3)))))+( (F1(3)*( (F2(2)*( (0, -1)*P3(1)+P3(2)))
     $ +(F2(1)*( (0, 1)*P3(0)+(0, -1)*P3(3)))))+(F1(4)*( (F2(1)*( (0,
     $ -1)*P3(1)-P3(2)))+(F2(2)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))))))
     $ *P3(2))+( -(F2(4)*F1(1))+(F2(3)*F1(2))+(F2(2)*F1(3))-(F2(1)
     $ *F1(4))))
      V3(4)= COUP*DENOM*( (OM3*( (F1(1)*( (F2(4)*( (0, 1)*P3(1)
     $ -P3(2)))+(F2(3)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))+( (F1(2)
     $ *( (F2(3)*( (0, 1)*P3(1)+P3(2)))+(F2(4)*( (0, 1)*P3(0)
     $ +(0, -1)*P3(3)))))+( (F1(3)*( (F2(2)*( (0, -1)*P3(1)+P3(2)))
     $ +(F2(1)*( (0, 1)*P3(0)+(0, -1)*P3(3)))))+(F1(4)*( (F2(1)*( (0,
     $ -1)*P3(1)-P3(2)))+(F2(2)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))))))
     $ *P3(3))+( (0, 1)*(F2(3)*F1(1))+(0, -1)*(F2(4)*F1(2))+(0,
     $ -1)*(F2(1)*F1(3))+(0, 1)*(F2(2)*F1(4))))
      END
```

[Slide from O. Mattelaer]

OLD ALOHA OUTPUT

```fortran
C     This File is Automatically generated by ALOHA
C     The process calculated in this file is:
C     Gamma(3,2,1)
C
      SUBROUTINE FFV1_3(F1, F2, COUP, M3, W3, V3)
      IMPLICIT NONE
      DOUBLE COMPLEX F1(*)
      DOUBLE COMPLEX F2(*)
      DOUBLE COMPLEX V3(*)
      DOUBLE COMPLEX COUP
      DOUBLE COMPLEX DENOM
      DOUBLE PRECISION M3, W3
      DOUBLE COMPLEX OM3
      DOUBLE PRECISION P3(0:3)

      V3(5)= +F1(5)+F2(5)
      V3(6)= +F1(6)+F2(6)
      P3(0) = - DBLE(V3(5))
      P3(1) = - DBLE(V3(6))
      P3(2) = - DIMAG(V3(6))
      P3(3) = - DIMAG(V3(5))
      OM3 = 0D0
      IF (M3 .NE. 0D0) OM3=1D0/M3**2

      DENOM =1D0/(( (M3*( -M3+(0, 1)*W3))+( (P3(0)**2)-(P3(1)**2)
     $ -(P3(2)**2)-(P3(3)**2))))
      V3(1)= COUP*DENOM*( (OM3*( (F1(1)*( (F2(4)*( (0, 1)*P3(1)
     $ -P3(2)))+(F2(3)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))+( (F1(2)
     $ *( (F2(3)*( (0, 1)*P3(1)+P3(2)))+(F2(4)*( (0, 1)*P3(0)
     $ +(0, -1)*P3(3)))))+( (F1(3)*( (F2(2)*( (0, -1)*P3(1)+P3(2)))
     $ +(F2(1)*( (0, 1)*P3(0)+(0, -1)*P3(3)))))+(F1(4)*( (F2(1)*( (0,
     $ -1)*P3(1)-P3(2)))+(F2(2)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))))))
     $ *P3(0))+( (0, -1)*(F2(3)*F1(1))+(0, -1)*(F2(4)*F1(2))+(0,
     $ -1)*(F2(1)*F1(3))+(0, -1)*(F2(2)*F1(4))))
      V3(2)= COUP*DENOM*( (OM3*( (F1(1)*( (F2(4)*( (0, 1)*P3(1)
     $ -P3(2)))+(F2(3)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))+( (F1(2)
     $ *( (F2(3)*( (0, 1)*P3(1)+P3(2)))+(F2(4)*( (0, 1)*P3(0)
     $ +(0, -1)*P3(3)))))+( (F1(3)*( (F2(2)*( (0, -1)*P3(1)+P3(2)))
     $ +(F2(1)*( (0, 1)*P3(0)+(0, -1)*P3(3)))))+(F1(4)*( (F2(1)*( (0,
     $ -1)*P3(1)-P3(2)))+(F2(2)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))))))
     $ *P3(1))+( (0, 1)*(F2(4)*F1(1))+(0, 1)*(F2(3)*F1(2))+(0,
     $ -1)*(F2(2)*F1(3))+(0, -1)*(F2(1)*F1(4))))
      V3(3)= COUP*DENOM*( (OM3*( (F1(1)*( (F2(4)*( (0, 1)*P3(1)
     $ -P3(2)))+(F2(3)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))+( (F1(2)
     $ *( (F2(3)*( (0, 1)*P3(1)+P3(2)))+(F2(4)*( (0, 1)*P3(0)
     $ +(0, -1)*P3(3)))))+( (F1(3)*( (F2(2)*( (0, -1)*P3(1)+P3(2)))
     $ +(F2(1)*( (0, 1)*P3(0)+(0, -1)*P3(3)))))+(F1(4)*( (F2(1)*( (0,
     $ -1)*P3(1)-P3(2)))+(F2(2)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))))))
     $ *P3(2))+( -(F2(4)*F1(1))+(F2(3)*F1(2))+(F2(2)*F1(3))-(F2(1)
     $ *F1(4))))
      V3(4)= COUP*DENOM*( (OM3*( (F1(1)*( (F2(4)*( (0, 1)*P3(1)
     $ -P3(2)))+(F2(3)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))+( (F1(2)
     $ *( (F2(3)*( (0, 1)*P3(1)+P3(2)))+(F2(4)*( (0, 1)*P3(0)
     $ +(0, -1)*P3(3)))))+( (F1(3)*( (F2(2)*( (0, -1)*P3(1)+P3(2)))
     $ +(F2(1)*( (0, 1)*P3(0)+(0, -1)*P3(3)))))+(F1(4)*( (F2(1)*( (0,
     $ -1)*P3(1)-P3(2)))+(F2(2)*( (0, 1)*P3(0)+(0, 1)*P3(3)))))))))
     $ *P3(3))+( (0, 1)*(F2(3)*F1(1))+(0, -1)*(F2(4)*F1(2))+(0,
     $ -1)*(F2(1)*F1(3))+(0, 1)*(F2(2)*F1(4))))
      END
```

OLD ALOHA OUTPUT

```fortran
C     This File is Automatically generated by ALOHA
C     The process calculated in this file is:
C     Gamma(3,2,1)
C
      SUBROUTINE FFV1_3(F1, F2, COUP, M3, W3,V3)
      IMPLICIT NONE
      COMPLEX*16 DENOM
      COMPLEX*16 V3(6)
      COMPLEX*16 TMP1
      REAL*8 W3
      REAL*8 P3(0:3)
      REAL*8 M3
      COMPLEX*16 F1(*)
      COMPLEX*16 F2(*)
      REAL*8 OM3
      COMPLEX*16 COUP
      OM3 = 0D0
      IF (M3.NE.0D0) OM3=1D0/M3**2
      V3(5) = -F1(5)+F2(5)
      V3(6) = -F1(6)+F2(6)
      P3(0) = -DBLE(V3(5))
      P3(1) = -DBLE(V3(6))
      P3(2) = -DIMAG(V3(6))
      P3(3) = -DIMAG(V3(5))
      TMP1 = (F1(1)*(F2(3)*(P3(0)+P3(3))+F2(4)*(P3(1)+(0D0, 1D0)
     $ *(P3(2))))+(F1(2)*(F2(3)*(P3(1)+(0D0, -1D0)*(P3(2)))+F2(4)
     $ *(P3(0)-P3(3)))+(F1(3)*(F2(1)*(P3(0)-P3(3))+F2(2)*-1D0*(P3(1)
     $ +(0D0, 1D0)*(P3(2))))+F1(4)*(F2(1)*(+(0D0, 1D0)*(P3(2))-P3(1))
     $ +F2(2)*(P3(0)+P3(3)))))
      DENOM = 1D0/(P3(0)**2-P3(1)**2-P3(2)**2-P3(3)**2 - M3 * (M3
     $ -(0,1)* W3))
      V3(1)= COUP*DENOM*(0D0, -1D0)*(F1(1)*F2(3)+F1(2)*F2(4)+F1(3)
     $ *F2(1)+F1(4)*F2(2)-P3(0)*OM3*TMP1)
      V3(2)= COUP*DENOM*(0D0, -1D0)*(F1(3)*F2(2)+F1(4)*F2(1)-F1(1)
     $ *F2(4)-F1(2)*F2(3)-P3(1)*OM3*TMP1)
      V3(3)= COUP*DENOM*(0D0, -1D0)*(+(0D0, -1D0)*(F1(1)*F2(4)
     $ +F1(4)*F2(1))+(0D0, 1D0)*(F1(2)*F2(3)+F1(3)*F2(2))-P3(2)*OM3
     $ *TMP1)
      V3(4)= COUP*DENOM*(0D0, -1D0)*(F1(2)*F2(4)+F1(3)*F2(1)-F1(1)
     $ *F2(3)-F1(4)*F2(2)-P3(3)*OM3*TMP1)
      END
```

[Slide from O. Mattelaer]

The UFO & ALOHA

- The development of the UFO goes hand in hand with the development of ALOHA.
- ALOHA = Automatic Language-independent Output of Helicity Amplitudes.
- Idea: ALOHA uses the information contained in the UFO to create the library of Lorentz structures for MadGraph on the fly.
- ALOHA is shipped with MadGraph, and runs unnoticed behind the scences... but it is one of the secret stars!
- It allows to output the Helas routines in Fortran, C++ (Pythia) and Python.
- UFO combined with ALOHA allows to implement virtually **any** model in MadGraph!

The UFO & ALOHA

- A neat application...

The UFO & ALOHA

- A neat application... in supergravity!

The UFO & ALOHA

- A neat application... in supergravity!

$$\mathcal{L} = -\frac{1}{4} F^a_{\mu\nu} F_a^{\mu\nu} + \lambda f^{abc} \mathrm{Tr}(F^a_{\mu\nu} F_b^{\nu\rho} F^c_{\rho\mu})$$

- Broedel and Dixon had derived a CSW construction for the color-ordered helicity amplitudes, but had no way to check the validity of the construction.

The UFO & ALOHA

- A neat application... in supergravity!

$$\mathcal{L} = -\frac{1}{4} F^a_{\mu\nu} F_a^{\mu\nu} + \lambda f^{abc} \mathrm{Tr}(F^a_{\mu\nu} F_b^{\nu\rho} F^c_{\rho\mu})$$

- Broedel and Dixon had derived a CSW construction for the color-ordered helicity amplitudes, but had no way to check the validity of the construction.
- Solution:
  - ➡ Put it into FeynRules, and let it run for a long time...
  - ➡ Get the UFO, and put it into MadGraph 5.
  - ➡ Hack matrix.f to read out the color-ordered helicity amplitudes for individual phase space points.

The UFO & ALOHA

```
V_4 = Vertex(name = 'V_4',
             particles = [ P.G, P.G, P.G, P.G, P.G, P.G ],
             color = [ 'f(-2,-3,-1)*f(-1,1,2)*f(3,4,-2)*f(5,6,-3)', 'f(-2,-3,-1)*f(-1,1,2)*f(3,4,-3)*f(5,6,-2)', 'f(-2,-3,-1)*f(-1,1,2)*f(3,5,-2)*f(4,6,-3)', 'f
(-2,-3,-1)*f(-1,1,2)*f(3,5,-3)*f(4,6,-2)', 'f(-2,-3,-1)*f(-1,1,2)*f(3,6,-2)*f(4,5,-3)', 'f(-2,-3,-1)*f(-1,1,2)*f(3,6,-3)*f(4,5,-2)', 'f(-2,-3,-1)*f(-1,1,3)*f(2,4,-2)*f
(5,6,-3)', 'f(-2,-3,-1)*f(-1,1,3)*f(2,4,-3)*f(5,6,-2)', 'f(-2,-3,-1)*f(-1,1,3)*f(2,5,-2)*f(4,6,-3)', 'f(-2,-3,-1)*f(-1,1,3)*f(2,5,-3)*f(4,6,-2)', 'f(-2,-3,-1)*f(-1,1,3)
*f(2,6,-2)*f(4,5,-3)', 'f(-2,-3,-1)*f(-1,1,3)*f(2,6,-3)*f(4,5,-2)', 'f(-2,-3,-1)*f(-1,1,4)*f(2,3,-2)*f(5,6,-3)', 'f(-2,-3,-1)*f(-1,1,4)*f(2,3,-3)*f(5,6,-2)', 'f
(-2,-3,-1)*f(-1,1,4)*f(2,5,-2)*f(3,6,-3)', 'f(-2,-3,-1)*f(-1,1,4)*f(2,5,-3)*f(3,6,-2)', 'f(-2,-3,-1)*f(-1,1,4)*f(2,6,-2)*f(3,5,-3)', 'f(-2,-3,-1)*f(-1,1,4)*f(2,6,-3)*f
(3,5,-2)', 'f(-2,-3,-1)*f(-1,1,5)*f(2,3,-2)*f(4,6,-3)', 'f(-2,-3,-1)*f(-1,1,5)*f(2,3,-3)*f(4,6,-2)', 'f(-2,-3,-1)*f(-1,1,5)*f(2,4,-2)*f(3,6,-3)', 'f(-2,-3,-1)*f(-1,1,5)
*f(2,4,-3)*f(3,6,-2)', 'f(-2,-3,-1)*f(-1,1,5)*f(2,6,-2)*f(3,4,-3)', 'f(-2,-3,-1)*f(-1,1,5)*f(2,6,-3)*f(3,4,-2)', 'f(-2,-3,-1)*f(-1,1,6)*f(2,3,-2)*f(4,5,-3)', 'f
(-2,-3,-1)*f(-1,1,6)*f(2,3,-3)*f(4,5,-2)', 'f(-2,-3,-1)*f(-1,1,6)*f(2,4,-2)*f(3,5,-3)', 'f(-2,-3,-1)*f(-1,1,6)*f(2,4,-3)*f(3,5,-2)', 'f(-2,-3,-1)*f(-1,1,6)*f(2,5,-2)*f
(3,4,-3)', 'f(-2,-3,-1)*f(-1,1,6)*f(2,5,-3)*f(3,4,-2)', 'f(-2,-3,-1)*f(-1,2,3)*f(1,4,-2)*f(5,6,-3)', 'f(-2,-3,-1)*f(-1,2,3)*f(1,4,-3)*f(5,6,-2)', 'f(-2,-3,-1)*f(-1,2,3)
*f(1,5,-2)*f(4,6,-3)', 'f(-2,-3,-1)*f(-1,2,3)*f(1,5,-3)*f(4,6,-2)', 'f(-2,-3,-1)*f(-1,2,3)*f(1,6,-2)*f(4,5,-3)', 'f(-2,-3,-1)*f(-1,2,3)*f(1,6,-3)*f(4,5,-2)', 'f
(-2,-3,-1)*f(-1,2,4)*f(1,3,-2)*f(5,6,-3)', 'f(-2,-3,-1)*f(-1,2,4)*f(1,3,-3)*f(5,6,-2)', 'f(-2,-3,-1)*f(-1,2,4)*f(1,5,-2)*f(3,6,-3)', 'f(-2,-3,-1)*f(-1,2,4)*f(1,5,-3)*f
(3,6,-2)', 'f(-2,-3,-1)*f(-1,2,4)*f(1,6,-2)*f(3,5,-3)', 'f(-2,-3,-1)*f(-1,2,4)*f(1,6,-3)*f(3,5,-2)', 'f(-2,-3,-1)*f(-1,2,5)*f(1,3,-2)*f(4,6,-3)', 'f(-2,-3,-1)*f(-1,2,5)
*f(1,3,-3)*f(4,6,-2)', 'f(-2,-3,-1)*f(-1,2,5)*f(1,4,-2)*f(3,6,-3)', 'f(-2,-3,-1)*f(-1,2,5)*f(1,4,-3)*f(3,6,-2)', 'f(-2,-3,-1)*f(-1,2,5)*f(1,6,-2)*f(3,4,-3)', 'f
(-2,-3,-1)*f(-1,2,5)*f(1,6,-3)*f(3,4,-2)', 'f(-2,-3,-1)*f(-1,2,6)*f(1,3,-2)*f(4,5,-3)', 'f(-2,-3,-1)*f(-1,2,6)*f(1,3,-3)*f(4,5,-2)', 'f(-2,-3,-1)*f(-1,2,6)*f(1,4,-2)*f
(3,5,-3)', 'f(-2,-3,-1)*f(-1,2,6)*f(1,4,-3)*f(3,5,-2)', 'f(-2,-3,-1)*f(-1,2,6)*f(1,5,-2)*f(3,4,-3)', 'f(-2,-3,-1)*f(-1,2,6)*f(1,5,-3)*f(3,4,-2)', 'f(-2,-3,-1)*f(-1,3,4)
*f(1,2,-2)*f(5,6,-3)', 'f(-2,-3,-1)*f(-1,3,4)*f(1,2,-3)*f(5,6,-2)', 'f(-2,-3,-1)*f(-1,3,4)*f(1,5,-2)*f(2,6,-3)', 'f(-2,-3,-1)*f(-1,3,4)*f(1,5,-3)*f(2,6,-2)', 'f
(-2,-3,-1)*f(-1,3,4)*f(1,6,-2)*f(2,5,-3)', 'f(-2,-3,-1)*f(-1,3,4)*f(1,6,-3)*f(2,5,-2)', 'f(-2,-3,-1)*f(-1,3,5)*f(1,2,-2)*f(4,6,-3)', 'f(-2,-3,-1)*f(-1,3,5)*f(1,2,-3)*f
(4,6,-2)', 'f(-2,-3,-1)*f(-1,3,5)*f(1,4,-2)*f(2,6,-3)', 'f(-2,-3,-1)*f(-1,3,5)*f(1,4,-3)*f(2,6,-2)', 'f(-2,-3,-1)*f(-1,3,5)*f(1,6,-2)*f(2,4,-3)', 'f(-2,-3,-1)*f(-1,3,5)
*f(1,6,-3)*f(2,4,-2)', 'f(-2,-3,-1)*f(-1,3,6)*f(1,2,-2)*f(4,5,-3)', 'f(-2,-3,-1)*f(-1,3,6)*f(1,2,-3)*f(4,5,-2)', 'f(-2,-3,-1)*f(-1,3,6)*f(1,4,-2)*f(2,5,-3)', 'f
(-2,-3,-1)*f(-1,3,6)*f(1,4,-3)*f(2,5,-2)', 'f(-2,-3,-1)*f(-1,3,6)*f(1,5,-2)*f(2,4,-3)', 'f(-2,-3,-1)*f(-1,3,6)*f(1,5,-3)*f(2,4,-2)', 'f(-2,-3,-1)*f(-1,4,5)*f(1,2,-2)*f
(3,6,-3)', 'f(-2,-3,-1)*f(-1,4,5)*f(1,2,-3)*f(3,6,-2)', 'f(-2,-3,-1)*f(-1,4,5)*f(1,3,-2)*f(2,6,-3)', 'f(-2,-3,-1)*f(-1,4,5)*f(1,3,-3)*f(2,6,-2)', 'f(-2,-3,-1)*f(-1,4,5)
*f(1,6,-2)*f(2,3,-3)', 'f(-2,-3,-1)*f(-1,4,5)*f(1,6,-3)*f(2,3,-2)', 'f(-2,-3,-1)*f(-1,4,6)*f(1,2,-2)*f(3,5,-3)', 'f(-2,-3,-1)*f(-1,4,6)*f(1,2,-3)*f(3,5,-2)', 'f
(-2,-3,-1)*f(-1,4,6)*f(1,3,-2)*f(2,5,-3)', 'f(-2,-3,-1)*f(-1,4,6)*f(1,3,-3)*f(2,5,-2)', 'f(-2,-3,-1)*f(-1,4,6)*f(1,5,-2)*f(2,3,-3)', 'f(-2,-3,-1)*f(-1,4,6)*f(1,5,-3)*f
(2,3,-2)', 'f(-2,-3,-1)*f(-1,5,6)*f(1,2,-2)*f(3,4,-3)', 'f(-2,-3,-1)*f(-1,5,6)*f(1,2,-3)*f(3,4,-2)', 'f(-2,-3,-1)*f(-1,5,6)*f(1,3,-2)*f(2,4,-3)', 'f(-2,-3,-1)*f(-1,5,6)
*f(1,3,-3)*f(2,4,-2)', 'f(-2,-3,-1)*f(-1,5,6)*f(1,4,-2)*f(2,3,-3)', 'f(-2,-3,-1)*f(-1,5,6)*f(1,4,-3)*f(2,3,-2)' ],
             lorentz = [ L.VVVVVV16, L.VVVVVV17, L.VVVVVV18, L.VVVVVV19, L.VVVVVV20, L.VVVVVV21, L.VVVVVV22, L.VVVVVV23, L.VVVVVV24, L.VVVVVV25, L.VVVVVV26,
L.VVVVVV27, L.VVVVVV28, L.VVVVVV29, L.VVVVVV30 ],
             couplings = {(5,5):C.GC_7,(4,5):C.GC_8,(3,3):C.GC_8,(2,3):C.GC_7,(11,9):C.GC_8,(10,9):C.GC_7,(7,1):C.GC_8,(6,1):C.GC_7,(17,12):C.GC_8,(16,12):C.GC_7,
(13,2):C.GC_8,(12,2):C.GC_7,(21,10):C.GC_7,(20,10):C.GC_8,(19,11):C.GC_7,(18,11):C.GC_8,(33,11):C.GC_8,(32,11):C.GC_7,(31,2):C.GC_7,(30,2):C.GC_8,(39,10):C.GC_8,
(38,10):C.GC_7,(37,1):C.GC_7,(36,1):C.GC_8,(51,12):C.GC_7,(50,12):C.GC_8,(49,9):C.GC_7,(48,9):C.GC_8,(63,12):C.GC_8,(62,12):C.GC_7,(61,3):C.GC_7,(60,3):C.GC_8,
(71,10):C.GC_7,(70,10):C.GC_8,(67,5):C.GC_8,(66,5):C.GC_7,(75,9):C.GC_8,(74,9):C.GC_7,(73,5):C.GC_7,(72,5):C.GC_8,(83,11):C.GC_7,(82,11):C.GC_8,(79,3):C.GC_8,
(78,3):C.GC_7,(89,2):C.GC_8,(88,2):C.GC_7,(87,1):C.GC_8,(86,1):C.GC_7,(9,7):C.GC_8,(8,7):C.GC_7,(15,13):C.GC_8,(14,13):C.GC_7,(27,8):C.GC_7,(26,8):C.GC_8,
(25,14):C.GC_7,(24,14):C.GC_8,(35,14):C.GC_8,(34,14):C.GC_7,(41,8):C.GC_8,(40,8):C.GC_7,(45,13):C.GC_7,(44,13):C.GC_8,(43,7):C.GC_7,(42,7):C.GC_8,(65,8):C.GC_7,
(64,8):C.GC_8,(69,13):C.GC_8,(68,13):C.GC_7,(77,14):C.GC_7,(76,14):C.GC_8,(81,7):C.GC_8,(80,7):C.GC_7,(1,0):C.GC_8,(0,0):C.GC_7,(23,4):C.GC_8,(22,4):C.GC_7,
(53,4):C.GC_7,(52,4):C.GC_8,(57,4):C.GC_8,(56,4):C.GC_7,(55,0):C.GC_7,(54,0):C.GC_8,(85,0):C.GC_8,(84,0):C.GC_7,(29,6):C.GC_7,(28,6):C.GC_8,(47,6):C.GC_8,(46,6):C.GC_7,
(59,6):C.GC_7,(58,6):C.GC_8})
```

The UFO & ALOHA

```
VVVV42 = Lorentz(name = 'VVVV42',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(4,5)*Metric(1,3)*Metric(2,5) - P(1,5)*Metric(2,5)*Metric(3,4) - P(4,5)*Metric(1,2)*Metric(3,5) + P(1,5)*Metric(2,4)*Metric(3,5)')

VVVV43 = Lorentz(name = 'VVVV43',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(5,1)*Metric(1,4)*Metric(2,3) - P(3,1)*Metric(1,4)*Metric(2,5) - P(5,1)*Metric(1,2)*Metric(3,4) + P(3,1)*Metric(1,2)*Metric(4,5)')

VVVV44 = Lorentz(name = 'VVVV44',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(4,1)*Metric(1,5)*Metric(2,3) - P(3,1)*Metric(1,5)*Metric(2,4) - P(4,1)*Metric(1,2)*Metric(3,5) + P(3,1)*Metric(1,2)*Metric(4,5)')

VVVV45 = Lorentz(name = 'VVVV45',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(5,2)*Metric(1,3)*Metric(2,4) - P(3,2)*Metric(1,5)*Metric(2,4) - P(5,2)*Metric(1,2)*Metric(3,4) + P(3,2)*Metric(1,2)*Metric(4,5)')

VVVV46 = Lorentz(name = 'VVVV46',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(4,2)*Metric(1,3)*Metric(2,5) - P(3,2)*Metric(1,4)*Metric(2,5) - P(4,2)*Metric(1,2)*Metric(3,5) + P(3,2)*Metric(1,2)*Metric(4,5)')

VVVV47 = Lorentz(name = 'VVVV47',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(4,1)*Metric(1,5)*Metric(2,3) - P(4,1)*Metric(1,3)*Metric(2,5) - P(2,1)*Metric(1,5)*Metric(3,4) + P(2,1)*Metric(1,3)*Metric(4,5)')

VVVV48 = Lorentz(name = 'VVVV48',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(5,1)*Metric(1,4)*Metric(2,3) - P(5,1)*Metric(1,3)*Metric(2,4) - P(2,1)*Metric(1,4)*Metric(3,5) + P(2,1)*Metric(1,3)*Metric(4,5)')

VVVV49 = Lorentz(name = 'VVVV49',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(5,3)*Metric(1,3)*Metric(2,4) - P(5,3)*Metric(1,2)*Metric(3,4) + P(2,3)*Metric(1,5)*Metric(3,4) - P(2,3)*Metric(1,3)*Metric(4,5)')

VVVV50 = Lorentz(name = 'VVVV50',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(4,3)*Metric(1,3)*Metric(2,5) - P(4,3)*Metric(1,2)*Metric(3,5) + P(2,3)*Metric(1,4)*Metric(3,5) - P(2,3)*Metric(1,3)*Metric(4,5)')

VVVV51 = Lorentz(name = 'VVVV51',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(3,4)*Metric(1,4)*Metric(2,5) - P(2,4)*Metric(1,4)*Metric(3,5) - P(3,4)*Metric(1,2)*Metric(4,5) + P(2,4)*Metric(1,3)*Metric(4,5)')
```

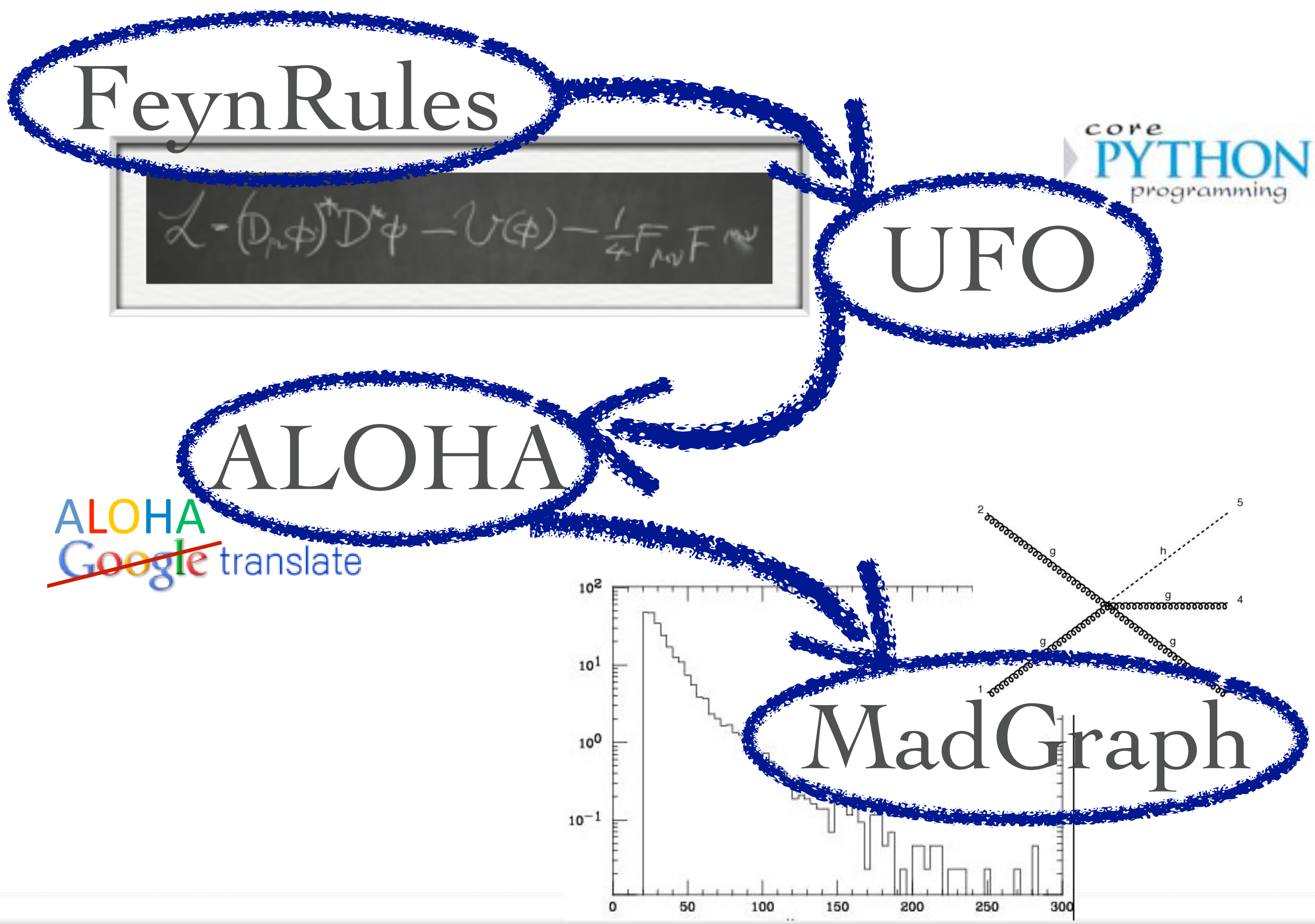
The UFO & ALOHA

```
VVVV42 = Lorentz(name = 'VVVV42',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(4,5)*Metric(1,3)*Metric(2,5) - P(1,5)*Metric(2,5)*Metric(3,4) - P(4,5)*Metric(1,2)*Metric(3,5) + P(1,5)*Metric(2,4)*Metric(3,5)')

VVVV43 = Lorentz(name = 'VVVV43',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(5,1)*Metric(1,4)*Metric(2,3) - P(3,1)*Metric(1,4)*Metric(2,5) - P(5,1)*Metric(1,2)*Metric(3,4) + P(3,1)*Metric(1,2)*Metric(4,5)')

VVVV44 = Lorentz(name = 'VVVV44',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(4,1)*Metric(1,5)*Metric(2,3) - P(3,1)*Metric(1,5)*Metric(2,4) - P(4,1)*Metric(1,2)*Metric(3,5) + P(3,1)*Metric(1,2)*Metric(4,5)')

VVVV45 = Lorentz(name = 'VVVV45',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(5,2)*Metric(1,3)*Metric(2,4) - P(3,2)*Metric(1,5)*Metric(2,4) - P(5,2)*Metric(1,2)*Metric(3,4) + P(3,2)*Metric(1,2)*Metric(4,5)')

VVVV46 = Lorentz(name = 'VVVV46',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(4,[illegible])*Metric(1,3)*Metric(2,5) - P(3,2)*Metric(1,4)*Metric([illegible],5) - P(4,2)*Metric(1,2)*Metr[illegible]c(3,5) + P(3,2)*Metric(1,2)*Metric(4,5)')

VVVV47 = Lorentz(name = 'VVVV47',
                 spins = [ 3, 3, [illegible] ],
                 structure = 'P(4,[illegible])*Metric(1,5)*Metric(2,3) - P(4,1)*Metric(1,3)*Metric(2,5) - P(2,1)*Metric(1,5)*Metr[illegible](3,4) + P(2,1)*Metric(1,3)*Metric(4,5)')

VVVV48 = Lorentz(name = 'VVVV48',
                 spins = [ 3, 3, [illegible] 3, 3 ],
                 structure = 'P(5,1)*Metric(1,4)*Metric(2,3) [illegible] *Metric(1,5)*Metric(2,4) - P(2,1)*Metric(1,4)*Metric(3,5) + P(2,1)*Metric(1,3)*Metric(4,5)')

VVVV49 = Lorentz(name = 'VVVV49',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(5,3)*Metric(1,3)*Metric(2,4) - P(5,3)*Metric(1,2)*Metric(3,4) + P(2,3)*Metric(1,5)*Metric(3,4) - P(2,3)*Metric(1,3)*Metric(4,5)')

VVVV50 = Lorentz(name = 'VVVV50',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(4,3)*Metric(1,3)*Metric(2,5) - P(4,3)*Metric(1,2)*Metric(3,5) + P(2,3)*Metric(1,4)*Metric(3,5) - P(2,3)*Metric(1,3)*Metric(4,5)')

VVVV51 = Lorentz(name = 'VVVV51',
                 spins = [ 3, 3, 3, 3, 3 ],
                 structure = 'P(3,4)*Metric(1,4)*Metric(2,5) - P(2,4)*Metric(1,4)*Metric(3,5) - P(3,4)*Metric(1,2)*Metric(4,5) + P(2,4)*Metric(1,3)*Metric(4,5)')
```

All Helicity amplitudes agreed out of the box with the CSW construction!

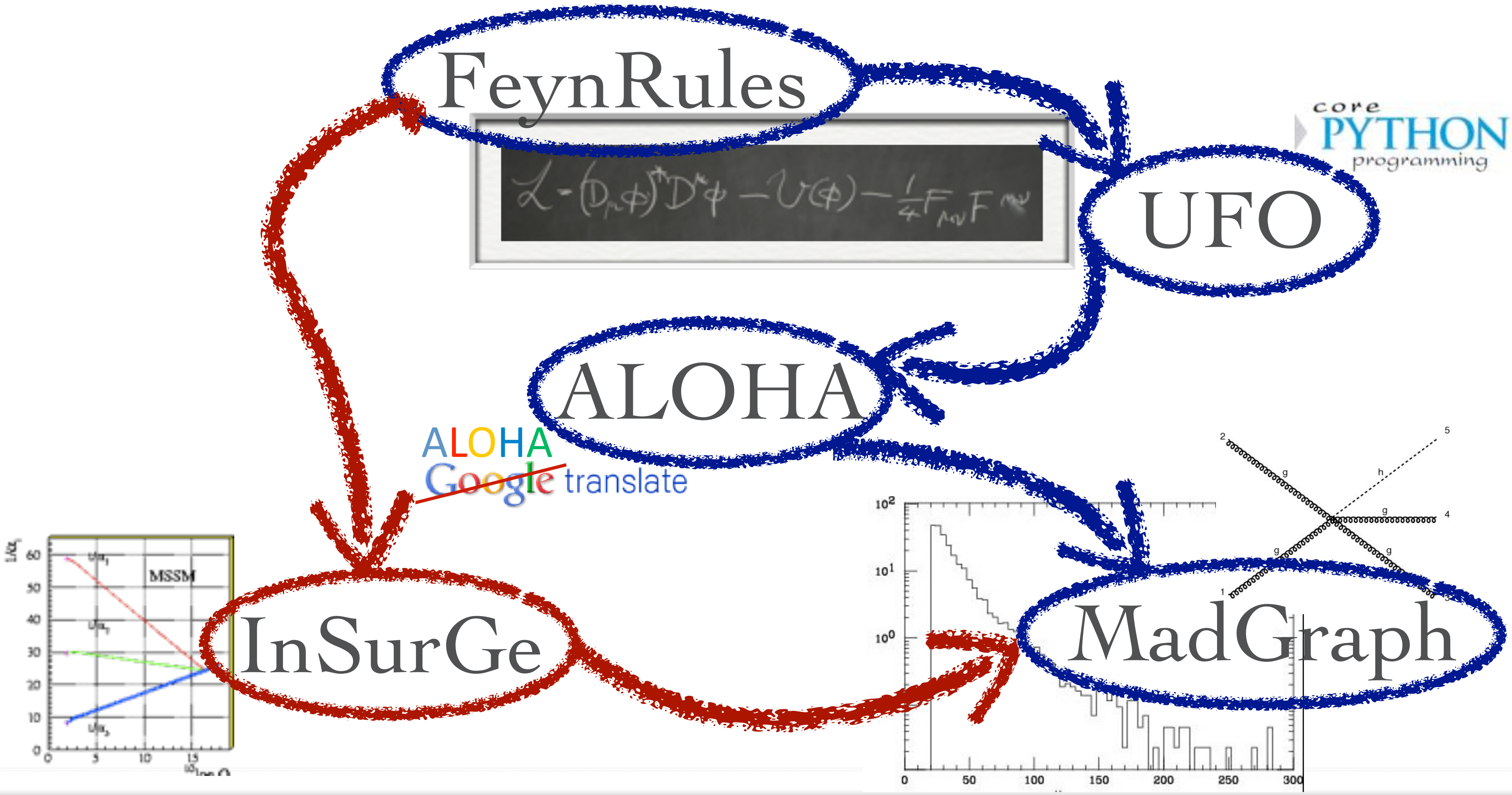
Summary

- UFO combined with ALOHA allows to implement virtually **any** model in MadGraph!
- We thus have a simulation chain that is complete and goes all the way from the Lagrangian to the events!



Summary

- UFO combined with ALOHA allows to implement virtually **any** model in MadGraph!
- We thus have a simulation chain that is complete and goes all the way from the Lagrangian to the events!



Summary

- UFO combined with ALOHA allows to implement virtually **any** model in MadGraph!
- We thus have a simulation chain that is complete and goes all the way from the Lagrangian to the events!