Beyond the Standard Model phenomenology with FEYNRULES

Fuks Benjamin

CERN - IPHC - U. Strasbourg

The Third NCTS school on FEYNRULES-MADGRAPH for LHC Physics
@ National Tsing Hua University, Taipei, Taiwan

June 16-20, 2014

Outline

Monte Carlo tools and discoveries at the LHC (I)

Assumption

There is some new physics to be discovered

A BSM LHC story

- *A priori* preparation
  - Viable model building (top-down & bottom-up)
  - Phenomenological studies
  - Prospective collider analyses
- *A posteriori* reactions to announcements
  - Model building (top-down & bottom-up)
  - Recasting experimental analyses
  - Measurements (precision predictions)

Predictions for the LHC

- Option 1: handmade calculations
  - Factorial growth of the number of diagrams
  - Tedious and error prone
- Option 2: Monte Carlo simulations
  - Easy to use
  - Can include the full collision environment

Monte Carlo tools and discoveries at the LHC (2)

Predictions for the LHC

~~✦ Option 1: handmade calculations~~
~~✤ Factorial growth of the number of diagrams~~
~~✤ Tedious and error prone~~

✦ Option 2: Monte Carlo simulations
✤ Easy to use
✤ Can include the full collision environment

✦ How to implement a new physics model in a Monte Carlo program?

★ Model definition: particles, parameters & vertices (≡ Lagrangian)

★ To be translated in a programming language, following some conventions, *etc*.

★ Tedious, time-consuming, error prone

★ Iterations for all considered tools and models

★ Beware of the restrictions of each tool (Lorentz structures, color structures)

★ **Highly redundant** (each tool, each model)
★ **No-brainer task** (from Feymman rules to code)

→ **FEYNRULES**: **Systematization & automation**

FEYNRULES in a nutshell

✦ What is FEYNRULES?

- ✤ A framework to develop new physics models
- ✤ Automatic export to several Monte Carlo event generators

➡ Facilitate phenomenological investigations of BSM models
➡ Facilitate the confrontation of BSM models to data

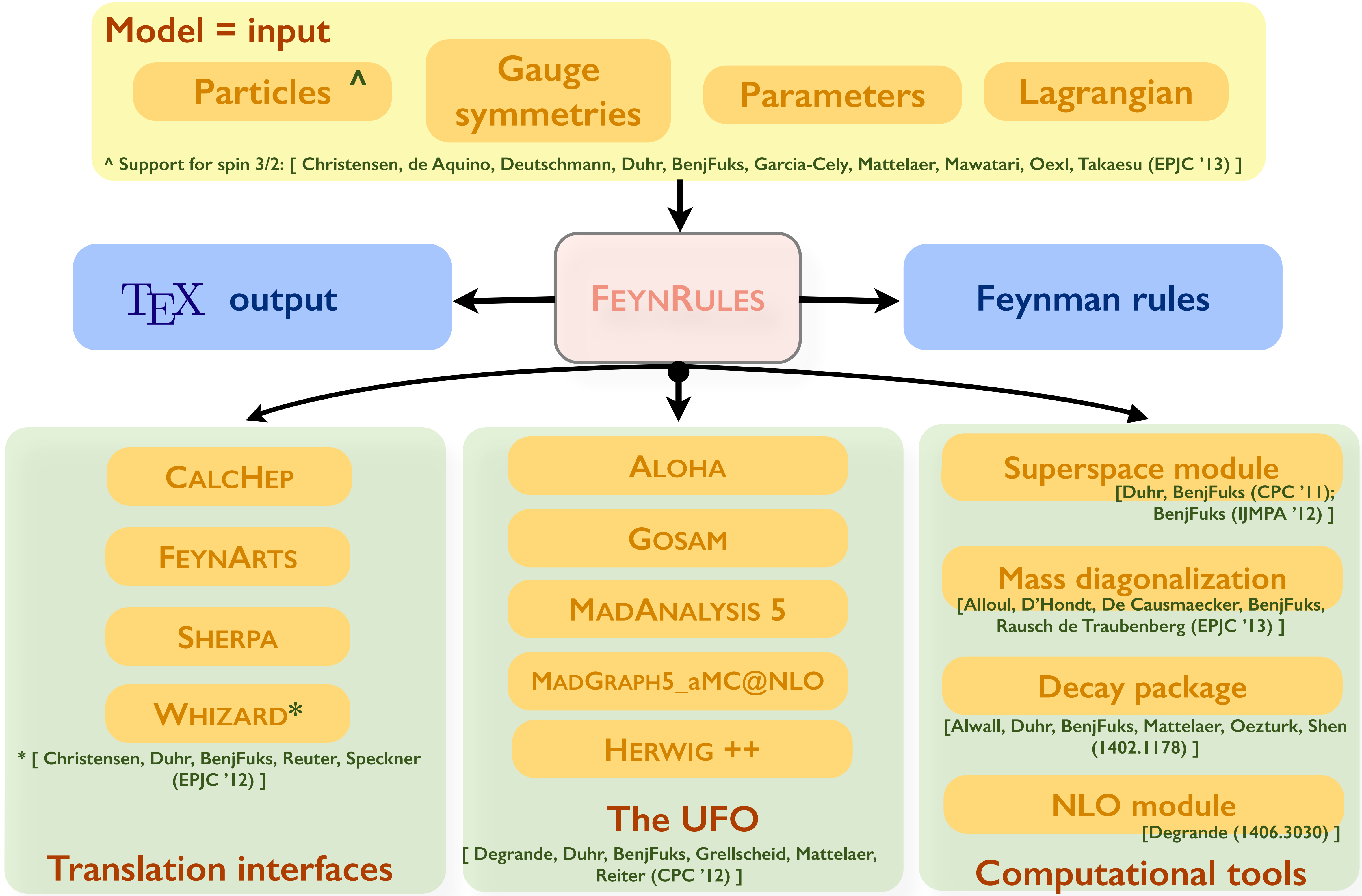
- ✤ Validation of an implementation using several of the linked Monte Carlo programs

✦ Main features

- ✤ MATHEMATICA package
- ✤ Core function: derives Feynman rules from a Lagrangian
- ✤ Requirements: locality, Lorentz and gauge invariance
- ✤ Supported fields: scalar, (two- and four-component) fermion, vector (and ghost), spin-3/2, tensor, superfield

From FeynRules to Monte Carlo tools

[Christensen, Duhr (CPC '09); Alloul, Christensen, Degrande, Duhr, BenjFuks (CPC '14)]



Outline

(Broken) supersymmetric QCD: the model

- Particle content (simplified for the scope of the lecture)
  - Two matter supermultiplets in the fundamental representation of $SU(3)_c$
    - ★ One massive Dirac fermion: a quark
    - ★ Two massive scalar fields: a left-handed and a right-handed squark
  - One ($SU(3)_c$) gauge supermultiplet
    - ★ One massive Majorana fermion: a gluino
    - ★ One massless gauge boson: the gluon

- The dynamics of the model is embedded in the Lagrangian

$$\begin{aligned}\mathcal{L} = & -\frac{1}{4} g_{\mu\nu} g^{\mu\nu} + \frac{i}{2} \bar{\tilde{g}} \not{D} \tilde{g} + D_\mu \tilde{q}_L^\dagger D^\mu \tilde{q}_L + D_\mu \tilde{q}_R^\dagger D^\mu \tilde{q}_R + i \bar{q} \not{D} q \\ & - m^2_{\tilde{q}_L} \tilde{q}_L^\dagger \tilde{q}_L - m^2_{\tilde{q}_R} \tilde{q}_R^\dagger \tilde{q}_R - m_q \bar{q} q - \frac{1}{2} m_{\tilde{g}} \bar{\tilde{g}} \tilde{g} \\ & - \frac{g_s^2}{2} \Big[- \tilde{q}_L^\dagger T^a \tilde{q}_L + \tilde{q}_R^\dagger T^a \tilde{q}_R \Big] \Big[- \tilde{q}_L^\dagger T^a \tilde{q}_L + \tilde{q}_R^\dagger T^a \tilde{q}_R \Big] \\ & + \sqrt{2} g_s \Big[- \tilde{q}_L^\dagger T^a \big(\bar{\tilde{g}}^a P_L q \big) + \big(\bar{q} P_L \tilde{g}^a \big) T^a \tilde{q}_R + \text{h.c.} \Big]\end{aligned}$$

  - Kinetic terms for all fields (first line)
  - Mass terms for the quark, squarks and gluino (second line)
  - (supersymmetric) gauge interactions for all fields (the last two lines of the Lagrangian)

How to write a FEYNRULES model file?

- A FEYNRULES model file is compliant with the MATHEMATICA syntax
- It is a .fr file containing:

- A preamble
  - Author information
  - Model information
  - Index definitions

- The declaration of the gauge group
  - Abelian or not
  - Representation matrices
  - Structure constants
  - Coupling constant
  - Gauge boson or vector superfield

- The declaration of the fields
  - Names, spins, PDG codes
  - Indices, quantum numbers
  - Masses, widths
  - Classes and class members

- The declaration of the parameters
  - External and internal
  - Scalar and tensor

- A Lagrangian

The preamble of the model file: general information

- An electronic signature for the model implementation
  - Important for traceability, documentation, contact with the model authors, *etc.*
  - Reference publications used can be added
  - Webpage information can be added

```
(* ****************************************************** *)
(* *****                                            ***** *)
(* *****  FeynRules model file: SUSY-QCD            ***** *)
(* *****  Author: B. Fuks                           ***** *)
(* *****                                            ***** *)
(* ****************************************************** *)


(* ************************ *)
(* *****  Information   ***** *)
(* ************************ *)
M$ModelName = "SUSYQCD";

M$Information = {
  Authors      -> {"Benjamin Fuks"},
  Date         ->  “16.06.14”,
  Version      ->  "1.0.0",
  Institutions -> {“CERN / IPHC Strasbourg / U. of Strasbourg"},
  Emails       -> {"benjamin.fuks@iphc.cnrs.fr"}
};
```

The preamble of the model file: indices

✦ The dimension of the indices must be declared

- In our SUSY-QCD model:
  - ★ Fundamental $SU(3)_C$ indices for the squarks and quark: *Colour*, dimension 3
  - ★ Adjoint $SU(3)_C$ indices for the gluon and gluino: *Gluon*, dimension 8
  - ★ Lorentz and spin indices are automatically handled

```
(* ************************ *)
(* *****    Indices    ***** *)
(* ************************ *)
IndexRange[Index[Gluon ]] = NoUnfold[Range[8]];
IndexRange[Index[Colour]] = NoUnfold[Range[3]];
```

- QCD has a special role for Monte Carlo event generators ➢ many special names
  - ★ *Colour* and *Gluon* for the indices
  - ★ *G* for the gluon field
  - ★ *T* for the fundamental representation matrices, *f* and *d* for the structure constants
  - ★ G (gs will be used) and aS for the coupling constants

✦ The style of the indices can be specified

- In our SUSY-QCD model:
  - ★ Fundamental $SU(3)_C$ indices: starts with the letter *m*
  - ★ Adjoint $SU(3)_C$ indices: starts with the letter *a*

```
IndexStyle[Colour, m];
IndexStyle[Gluon,  a];
```

The declaration of the gauge group

- Each direct factor of the group is declared as an element of the *M$GaugeGroups list*
  - A declaration ≡ a set of MATHEMATICA replacement rules
  - In our SUSY-QCD model:
    - ★ We must only declare $SU(3)_C$: we choose the name *SU3C*

```
(* ************************ *)
(* *****  Gauge groups  ***** *)
(* ************************ *)
M$GaugeGroups = {
  SU3C == {
    Abelian           -> False,
    GaugeBoson        -> G,
    CouplingConstant  -> gs,
    StructureConstant -> f,
    Representations   -> { {T,Colour} }
  }
};
```

  - Each rule represents one group property (reminder for QCD: special names exist)
    - ★ *Abelian*: abelian or non-abelian gauge group
    - ★ *GaugeBoson*: the associated gauge boson
    - ★ *CouplingConstant, StructureConstant*: coupling and the structure constants
    - ★ *Representation*: list of 2-tuples linking an index (*Colour*) to the symbol of a representation matrix (*T*)

- Advantages of a proper gauge group declaration
  - Render the writing of the Lagrangian easier:
    - ★ Covariant derivatives (*DC[field, Lorentz index]*)
    - ★ Field strength tensors (*FS[field, Lorentz index 1, Lorentz index 2]*)
    - ★ Useful for Lagrangian building in superspace (very briefly covered in the last part of this lecture)
      - ➢ Duhr, BenjFuks [CPC 182 (2011) 2404]; BenjFuks [IJMPA 27 (2012) 1230007]

See the manual for more details on gauge groups

Declaring the gluon field

- Each field is declared as an element of the *M$ClassesDescription* list
  - A declaration ≡ a set of MATHEMATICA replacement rules
  - In our SUSY-QCD model, we first declare the $SU(3)_C$ gauge boson: *G*

```
(* ************************* *)
(* *****     Fields     ***** *)
(* ************************* *)
M$ClassesDescription = {
  V[1] == {
    ClassName        -> G,
    SelfConjugate    -> True,
    Indices          -> {Index[Gluon]},
    Mass             -> 0,
    Width            -> 0,
    PDG              -> 21
  },
  …
```

- Each rule represents a property of the declared field
  - ★ Vector field ➢ the label is V[1] (with V, and not F, S, R, T, *etc.*)
  - ★ Classname: defines the symbol to use in the Lagrangian ➢ *G*
  - ★ SelfConjugate: the gluon is its own antiparticle ➢ *True*
  - ★ Indices: the gluon lies in the adjoint representation of SU(3)c
    - ➢ The gluon has been previously set as the gauge boson of $SU(3)_c$
    - ➢ Its index (*Gluon*) is internally linked to the adjoint representation of the group
  - ★ Other properties: vanishing mass and widths, PDG code set to 21
  - ★ Not used options: *Unphysical, Definitions, PropagatorLabel, PropagatorType, PropagatorArrow, ParticleName, AntiParticleName, QuantumNumbers*

See the manual for more details on field declarations

Declaring the gluino field

✦ A second element in the *M$ClassesDescription* list

```
F[1] == {
  ClassName        -> go,
  SelfConjugate    -> True,
  Indices          -> {Index[Gluon]},
  Mass             -> {Mgo,500},
  Width            -> {Wgo,10},
  PDG              -> 1000021
},
```

✤ Differences with the gluon field declaration

- ★ Four-component fermionic field ➢ the label is F[1] (with an F)
- ★ Classname in the fermion case: defines two symbols to use in the Lagrangian ➢ *go* and *gobar*
- ★ Mass and width: we define two symbols and their associated numerical values

✦ Second option: two-component spinors (Lagrangians are sometimes easier to write)

```
W[1] == {
  ClassName     -> gow,
  Unphysical    -> True,
  Chirality     -> Left,
  SelfConjugate -> False,
  Indices       -> {Index[Gluon]},
},
```

```
F[1] == {
  ClassName        -> go,
  WeylComponents   -> gow,
  SelfConjugate    -> True,
  Indices          -> {Index[Gluon]},
  Mass             -> {Mgo,500},
  Width            -> {Wgo,10},
  PDG              -> 1000021
},
```

- ★ Two-component and four-component fermionic fields ➢ the labels are W[1] (with a W) and F[1]
- ★ Weyl fermions are unphysical and linked to a four-component fermion (WeylComponents)
- ★ Several symbols are defined ➢ *go* and *gobar; gow* and *gowbar*
- ★ The chirality of the Weyl fermion can be specified

See the manual for more details on field declarations

Declaring the (top) quark field

✦ A third element in the *M$ClassesDescription* list

```
F[2] == {
  ClassName        -> q,
  SelfConjugate    -> False,
  Indices          -> {Index[Colour]},
  Mass             -> {Mq, 173},
  Width            -> {Wq, 1.50833649},
  PDG              -> 6
},
```

✤ Nothing special compared to the other fields

- ★ Fundamental QCD indices are specified

✦ Parenthesis: three generations of up-type quarks (*Gen* being the generation index)

```
F[3] == {
  ClassName        -> uq,
  ClassMembers     -> {u, c, t},
  SelfConjugate    -> False,
  Indices          -> {Index[Gen], Index[Colour]},
  FlavorIndex      -> Gen,
  Mass             -> {Mu, {MU,2.55*^-3}, {MC,1.42}, {MT,173}},
  Width            -> {0, 0, {WT, 1.50833649}},
  PDG              -> {2, 4, 6}
},
```

- ★ ClassMembers: specify all the members of the class
- ★ FlavorIndex: defines which index is the flavor index (efficiency of the code)
- ★ Mass, Width, PDG: one for each class member (plus a generic mass symbol)

See the manual for more details on field declarations

Declaring the (top) squark fields

✦ Extra elements in the *M$ClassesDescription* list

```
S[1] == {
  ClassName        -> sqL,
  SelfConjugate    -> False,
  Indices          -> {Index[Colour]},
  Unphysical       -> True,
  Definitions      -> {sqL[c_] -> Cos[theta] sq1[c] - Sin[theta] sq2[c]}
},
S[2] == {
  ClassName        -> sqR,
  SelfConjugate    -> False,
  Indices          -> {Index[Colour]},
  Unphysical       -> True,
  Definitions      -> {sqR[c_] -> Sin[theta] sq1[c] + Cos[theta] sq2[c]}
},
```

```
S[3] == {
  ClassName        -> sq1,
  SelfConjugate    -> False,
  Indices          -> {Index[Colour]},
  Mass             -> {Msq1,300},
  Width            -> {Wsq1,10},
  PDG              -> 1000006
},
S[4] == {
  ClassName        -> sq2,
  SelfConjugate    -> False,
  Indices          -> {Index[Colour]},
  Mass             -> {Msq2,800},
  Width            -> {Wsq2,2},
  PDG              -> 2000006
}
```

✤ Squark fields mix:

$$\begin{pmatrix}\tilde q_1\\ \tilde q_2\end{pmatrix} = \begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}\tilde q_L\\ \tilde q_R\end{pmatrix}$$

- ★ Left and right-handed squarks are declared as unphysical
- ★ Definitions linking gauge- and mass-eigenstates are provided (inversion of the relation above)
  - ➢ The rotations will be performed automatically by FEYNRULES
  - ➢ The Lagrangian can be written in the gauge basis (easier)

See the manual for more details on field declarations

Parameter declaration

✦ The model contains three parameters:

- ✤ The strong coupling constants: g_s, α_s are both needed (required by the Monte Carlo tools)
- ✤ The squark mixing angle θ
- ✤ Masses and widths are handled automatically

✦ Parameters are declared as elements of the list *M$Parameters*

- ✤ A declaration ≡ a set of MATHEMATICA replacement rules
- ✤ In our SUSY-QCD model, we have:

```
(* ************************** *)
(* *****   Parameters   ***** *)
(* ************************** *)
M$Parameters = {
  aS    == {
    ParameterType    -> External,
    Value            -> 0.1184,
    InteractionOrder -> {QCD, 2}
  },
  gs == {
    ParameterType    -> Internal,
    Value            -> Sqrt[4 Pi aS],
    InteractionOrder -> {QCD, 1},
    ParameterName    -> G
  },
  theta == {
    ParameterType    -> External,
    Value            -> Pi/4.
  }
};
```

- ★ We have Internal and External parameters
  - ➢ External ≡ free model parameter ⇒ numerical value
  - ➢ Internal ≡ dependent model parameter ⇒ formula
- ★ InteractionOrder: specific to MADGRAPH (more efficient diagram generation)
- ★ ParameterName: specific to Monte Carlo tools
- ★ Not used options: *TeX, Definitions, ComplexParameter, Description, BlockName, OrderBlock*
- ★ Specific to tensor parameters: *Indices, Unitary*

See the manual for more details on parameter declarations

Outline

Implementing the vector Lagrangian

- Temporary restriction to the gauge content of the theory
  - One ($SU(3)_c$) gauge supermultiplet
    - ★ One massive Majorana fermion: a gluino
    - ★ One massless gauge boson: the gluon

- The dynamics of the model is embedded in the vector Lagrangian

$$\mathcal{L}_{\rm vector} = -\frac{1}{4} g_{\mu\nu} g^{\mu\nu} + \frac{i}{2} \bar{\tilde{g}} \not{D} \tilde{g} - \frac{1}{2} m_{\tilde{g}} \bar{\tilde{g}} \tilde{g}$$

  - Kinetic terms for all the gluino and gluon fields
  - Mass terms for the gluino
  - Gauge interactions for both fields (embedded into gauge covariant objects)

- The implementation in FEYNRULES is easy (*cf.* predefined functions linked to the gauge group)
  - Option 1 (left): all non-Lorentz indices understood (FEYNRULES takes care of reintroducing them)
  - Option 2 (right): all indices explicit

```
LVector1 := -1/4 FS[G,mu,nu,a] FS[G,mu,nu,a] +
            I/2 gobar.Ga[mu].DC[go,mu] -
            1/2 Mgo gobar.go;
```

```
LVector2 := -1/4 FS[G,mu,nu,a] FS[G,mu,nu,a] +
            I/2 Ga[mu,s1,s2] gobar[s1,a].DC[go[s2,a],mu] -
            1/2 Mgo gobar[s1,a].go[s1,a];
```

Starting a MATHEMATICA session (1)

✦ Step1: loading FEYNRULES into MATHEMATICA

- ✤ Setting up the FEYNRULES path
- ✤ Loading FEYNRULES itself

```
In[2]:= $FeynRulesPath = SetDirectory["~/Work/tools/FeynRules/branch/feynrules-current"];
        << FeynRules`

         - FeynRules -
        Version: 2.0.25 ( 22 April 2014).
        Authors: A. Alloul, N. Christensen, C. Degrande, C. Duhr, B. Fuks

        Please cite:
            - arXiv:1310.1921;
            - Comput.Phys.Commun.180:1614-1641,2009 (arXiv:0806.4194).

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

        The FeynRules palette can be opened using the command FRPalette[].
```

✦ The output

- ✤ Information on FEYNRULES, the authors, the version number, references, *etc*.

Starting a MATHEMATICA session (2)

✦ Step2: loading the SUSY-QCD implementation into FEYNRULES
- Moving to the right directory
- Loading the model itself

```
In[4]:= SetDirectory[NotebookDirectory[]];
        LoadModel["susyqcd_nthu.fr"];
        This model implementation was created by
        Benjamin Fuks
        Model Version: 1.0.0
        For more information, type ModelInformation[].

          - Loading particle classes.
          - Loading gauge group classes.
          - Loading parameter classes.

        Model SUSYQCD loaded.
```

✦ The output
- Information of the model fil preamble are printed to the screen

Checking the implementation: comparing with books

✦ Printing the vector Lagrangian and comparing with the textbook expression

$$\mathcal{L}_{\rm vector} = -\frac{1}{4} g_{\mu\nu} g^{\mu\nu} + \frac{i}{2}\bar{\tilde{g}} \not{D} \tilde{g} - \frac{1}{2} m_{\tilde{g}} \bar{\tilde{g}} \tilde{g}$$

```
LVector1 := -1/4 FS[G,mu,nu,a] FS[G,mu,nu,a] +
            I/2 gobar.Ga[mu].DC[go,mu] -
            1/2 Mgo gobar.go;
```

In[8]:= **LVector1**

Out[8]= $-\frac{1}{2}\,\text{Mgo}\,\overline{\text{go}}.\text{go} + \frac{1}{2}\,i\,\overline{\text{go}}.\gamma^{mu}.\left(\partial_{mu}[\text{go}] - i\,\text{gs}\,\text{FSU3C}^{a\$471}.\text{go}\,G_{mu,a\$471}\right) - \frac{1}{4}\left(-\partial_{nu}[G_{mu,a}] + \partial_{mu}[G_{nu,a}] + \text{gs}\,f_{a,bb\$469,cc\$469}\,G_{mu,bb\$469}\,G_{nu,cc\$469}\right)\left(-\partial_{nu}[G_{mu,a}] + \partial_{mu}[G_{nu,a}] + \text{gs}\,f_{a,bb\$470,cc\$470}\,G_{mu,bb\$470}\,G_{nu,cc\$470}\right)$

✦ The output

✤ Covariant derivatives and field strength tensors have been automatically evaluated

✦ Restoring all indices automatically

In[9]:= **ExpandIndices[LVector1]**

Out[9]= $-\frac{1}{4}\partial_{nu}[G_{mu,a}]^2 + \frac{1}{2}\partial_{nu}[G_{mu,a}]\,\partial_{mu}[G_{nu,a}] - \frac{1}{4}\partial_{mu}[G_{nu,a}]^2 - \frac{1}{2}\,\text{Mgo}\,\text{go}_{i1\$478,i2\$478}.\text{go}_{i1\$478,i2\$478} + \frac{1}{4}\,\text{gs}\,\partial_{nu}[G_{mu,a}]\,f_{a,bb\$473,cc\$473}\,G_{mu,bb\$473}\,G_{nu,cc\$473} - \frac{1}{4}\,\text{gs}\,\partial_{mu}[G_{nu,a}]\,f_{a,bb\$473,cc\$473}\,G_{mu,bb\$473}\,G_{nu,cc\$473} + \frac{1}{4}\,\text{gs}\,\partial_{nu}[G_{mu,a}]\,f_{a,bb\$474,cc\$474}\,G_{mu,bb\$474}\,G_{nu,cc\$474} - \frac{1}{4}\,\text{gs}\,\partial_{mu}[G_{nu,a}]\,f_{a,bb\$474,cc\$474}\,G_{mu,bb\$474}\,G_{nu,cc\$474} - \frac{1}{4}\,\text{gs}^2\,f_{a,bb\$473,cc\$473}\,f_{a,bb\$474,cc\$474}\,G_{mu,bb\$473}\,G_{mu,bb\$474}\,G_{nu,cc\$473}\,G_{nu,cc\$474} + \frac{1}{4}\,i\,\text{go}_{i\$478,i1\$480}.\partial_{mu}[\text{go}_{j\$478,i1\$480}]\,\gamma^{mu}_{i\$478,j\$478} - \frac{1}{4}\,i\,\text{gs}\,\text{go}_{i\$481,i\$480}.\text{go}_{j\$481,j\$480}\,f_{a\$475,i\$480,j\$480}\,G_{mu,a\$475}\,\gamma^{mu}_{i\$481,j\$481} - \frac{1}{4}\,i\,\partial_{mu}[\text{go}_{j\$478,i1\$480}].\text{go}_{i\$478,i1\$480}\,\gamma^{mu}_{j\$478,i\$478} + \frac{1}{4}\,i\,\text{gs}\,\text{go}_{j\$481,j\$480}.\text{go}_{i\$481,i\$480}\,f_{a\$475,i\$480,j\$480}\,G_{mu,a\$475}\,\gamma^{mu}_{j\$481,i\$481}$

Checking the implementation: hermiticity, normalization

✦ The Lagrangian must be Hermitian

$$\mathcal{L}_{\rm vector} = -\frac{1}{4} g_{\mu\nu} g^{\mu\nu} + \frac{i}{2} \bar{\tilde{g}} \not{D} \tilde{g} - \frac{1}{2} m_{\tilde{g}} \bar{\tilde{g}} \tilde{g}$$

```
In[10]:= CheckHermiticity[LVector1];
Checking for hermiticity by calculating the Feynman rules contained in L-HC[L].
If the lagrangian is hermitian, then the number of vertices should be zero.
Starting Feynman rule calculation.
Expanding the Lagrangian...
No vertices found.
0 vertices obtained.
The lagrangian is hermitian.
```

✦ The Lagrangian must be canonically normalized

```
In[11]:= CheckKineticTermNormalisation[LVector1];
Neglecting all terms with more than 2 particles.
All kinetic terms are diagonal.
All kinetic terms are correctly normalized.
```

✤ Other methods: *CheckDiagonalQuadraticTerms, CheckDiagonalKineticTerms, CheckDiagonalMassTerms*

Checking the implementation: the mass spectrum

- Checks at the level of the masses can be performed
  - The masses can be extracted from the Lagrangian
  - The masses are given when particles are declared
  - They should match

```
In[12]:= CheckMassSpectrum[LVector1]
         Neglecting all terms with more than 2 particles.
         All mass terms are diagonal.
         Getting mass spectrum.
         Checking for less then 0.1% agreement with model file values.
Out[12]//TableForm=
         Particle    Analytic value    Numerical value    Model-file value
         go          Mgo               500.               500.
```

The Feynman rules (I)

✦ Extract all N-point interactions from the Lagrangian (with N>2)

```
In[14]:= FeynmanRules[LVector1, ScreenOutput → True];

Starting Feynman rule calculation.
Expanding the Lagrangian...
Collecting the different structures that enter the vertex.
3 possible non-zero vertices have been found -> starting the computation: 3 / 3.
3 vertices obtained.
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
Vertex 1
Particle 1 : Vector , G
Particle 2 : Vector , G
Particle 3 : Vector , G
Vertex:
```

$$gs\, f_{a_1,a_2,a_3}\, p_1^{\mu_3}\, \eta_{\mu_1,\mu_2} - gs\, f_{a_1,a_2,a_3}\, p_2^{\mu_3}\, \eta_{\mu_1,\mu_2} - gs\, f_{a_1,a_2,a_3}\, p_1^{\mu_2}\, \eta_{\mu_1,\mu_3} + gs\, f_{a_1,a_2,a_3}\, p_3^{\mu_2}\, \eta_{\mu_1,\mu_3} + gs\, f_{a_1,a_2,a_3}\, p_2^{\mu_1}\, \eta_{\mu_2,\mu_3} - gs\, f_{a_1,a_2,a_3}\, p_3^{\mu_1}\, \eta_{\mu_2,\mu_3}$$

- ✤ Three vertices are found
- ✤ The expression above consists of the triple gluon vertex
- ✤ The index a_i is related to the i^{th} particle
  (reminder: *a* is the index style for the adjoint $SU(3)_C$ indices)
- ✤ The index μ_i is the Lorentz index of the i^{th} (vector) particle

The Feynman rules (2)

✦ The second vertex: the quartic gluon interaction

```
Vertex 2
Particle 1 : Vector , G
Particle 2 : Vector , G
Particle 3 : Vector , G
Particle 4 : Vector , G
Vertex:
```

$$i\, gs^2\, f_{a_1,a_3,\text{Gluon\$1}}\, f_{a_2,a_4,\text{Gluon\$1}}\, \eta_{\mu_1,\mu_4}\, \eta_{\mu_2,\mu_3} + i\, gs^2\, f_{a_1,a_2,\text{Gluon\$1}}\, f_{a_3,a_4,\text{Gluon\$1}}\, \eta_{\mu_1,\mu_4}\, \eta_{\mu_2,\mu_3} + i\, gs^2\, f_{a_1,a_4,\text{Gluon\$1}}\, f_{a_2,a_3,\text{Gluon\$1}}\, \eta_{\mu_1,\mu_3}\, \eta_{\mu_2,\mu_4} - i\, gs^2\, f_{a_1,a_2,\text{Gluon\$1}}\, f_{a_3,a_4,\text{Gluon\$1}}\, \eta_{\mu_1,\mu_3}\, \eta_{\mu_2,\mu_4} - i\, gs^2\, f_{a_1,a_4,\text{Gluon\$1}}\, f_{a_2,a_3,\text{Gluon\$1}}\, \eta_{\mu_1,\mu_2}\, \eta_{\mu_3,\mu_4} - i\, gs^2\, f_{a_1,a_3,\text{Gluon\$1}}\, f_{a_2,a_4,\text{Gluon\$1}}\, \eta_{\mu_1,\mu_2}\, \eta_{\mu_3,\mu_4}$$

- ✤ The index a_i is related to the i^{th} particle
  (reminder: *a* is the index style for the adjoint $SU(3)_C$ indices)
- ✤ The index *Gluon$1* is repeated ⇒ summed index
- ✤ The index μ_i is the Lorentz index of the i^{th} (vector) particle

The Feynman rules (3)

✦ The third vertex: the gluon-gluino interaction

```
Vertex 3
Particle 1 : Majorana , go
Particle 2 : Majorana , go
Particle 3 : Vector , G
Vertex:
```

$$gs\, f_{a_1,a_2,a_3}\, \gamma_{s_1,s_2}^{\mu_3}$$

- The index a_i is related to the i^{th} particle (reminder: *a* is the index style for the adjoint $SU(3)_C$ indices)
- The index μ_i is the Lorentz index of the i^{th} (vector) particle
- The index s_i is the spin index of the i^{th} (fermionic) particle

✦ The function FeynmanRules has many options

- Restriction on the interactions (MaxParticles, MaxCanonicalDimension, *etc.*)
- Selection of specific particles (Free, Contains, *etc.*)
- ScreenOutput: displaying the vertices to the screen or not
- FlavorExpand: perform a flavor expansion (otherwise, classes are used)

See the manual for more details on the function *FeynmanRules*

Implementing the matter Lagrangian (I)

✦ Matter fields

✤ Two matter supermultiplets in the fundamental representation of $SU(3)_c$

★ One massive Dirac fermion: a quark

★ Two mixing massive scalar fields: two squark

★ Gauge interactions with the $SU(3)_c$ gauge supermultiplet

$$\begin{pmatrix} \tilde q_1 \\ \tilde q_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \tilde q_L \\ \tilde q_R \end{pmatrix}$$

✦ The dynamics of the model is embedded in the matter Lagrangian

$$\begin{aligned}\mathcal{L}_{\rm matter} = & D_\mu \tilde q_L^\dagger D^\mu \tilde q_L + D_\mu \tilde q_R^\dagger D^\mu \tilde q_R + i\bar q \not{D} q - m^2_{\tilde q_i} \tilde q_i^\dagger \tilde q_i - m_q \bar q q \\ & - \frac{g_s^2}{2}\Big[- \tilde q_L^\dagger T^a \tilde q_L + \tilde q_R^\dagger T^a \tilde q_R \Big] \Big[- \tilde q_L^\dagger T^a \tilde q_L + \tilde q_R^\dagger T^a \tilde q_R \Big] \\ & + \sqrt{2} g_s \Big[- \tilde q_L^\dagger T^a (\tilde g^a P_L q) + (\bar q P_L \tilde g^a) T^a \tilde q_R \Big] + \text{h.c.}\end{aligned}$$

✤ Kinetic and gauge interaction terms (half of the first line) in the gauge basis

✤ Mass terms (second half of the first line) in the mass basis (because easier to implement)

✤ D-terms (second line)

✤ Supersymmetric gauge quark-squark-gluino interactions (fourth line)

Implementing the matter Lagrangian (2)

✦ Kinetic and gauge interactions

$$\mathcal{L}_{\rm matter} = D_\mu \tilde{q}_L^\dagger D^\mu \tilde{q}_L + D_\mu \tilde{q}_R^\dagger D^\mu \tilde{q}_R + i\bar{q}\not{D}q - m_{\tilde{q}_i}^2 \tilde{q}_i^\dagger \tilde{q}_i - m_q \bar{q}q$$

✦ The implementation in FeynRules is easy (*cf.* predefined functions linked to the gauge group)

```
Lkin := DC[sqLbar[cc],mu] DC[sqL[cc],mu] +
        DC[sqRbar[cc],mu] DC[sqR[cc],mu] +
        I qbar.Ga[mu].DC[q,mu] -
        Mq qbar.q -
        Msq1^2 sq1bar[cc] sq1[cc] -
        Msq2^2 sq2bar[cc] sq2[cc];
```

✤ Repeated indices are summed

✦ We can compute the Feynman rules

```
In[9]:= Simplify[FeynmanRules[Lkin]] // MatrixForm
Starting Feynman rule calculation.
Expanding the Lagrangian...
Collecting the different structures that enter the vertex.
9 possible non-zero vertices have been found -> starting the computation: 9 / 9.
5 vertices obtained.
Out[9]//MatrixForm=
```

{{G, 1}, {sq1, 2}, {sq1†, 3}}	$i\, gs\, (p_2^{\mu_1} - p_3^{\mu_1})\, T^{a_1}_{m_3,m_2}$
{{G, 1}, {sq2, 2}, {sq2†, 3}}	$i\, gs\, (p_2^{\mu_1} - p_3^{\mu_1})\, T^{a_1}_{m_3,m_2}$
{{G, 1}, {G, 2}, {sq1, 3}, {sq1†, 4}}	$i\, gs^2\, \eta_{\mu_1,\mu_2} (T^{a_1}_{m_4,\mathrm{Colour\$1}} T^{a_2}_{\mathrm{Colour\$1},m_3} + T^{a_1}_{\mathrm{Colour\$1},m_3} T^{a_2}_{m_4,\mathrm{Colour\$1}})$
{{G, 1}, {G, 2}, {sq2, 3}, {sq2†, 4}}	$i\, gs^2\, \eta_{\mu_1,\mu_2} (T^{a_1}_{m_4,\mathrm{Colour\$1}} T^{a_2}_{\mathrm{Colour\$1},m_3} + T^{a_1}_{\mathrm{Colour\$1},m_3} T^{a_2}_{m_4,\mathrm{Colour\$1}})$
{{q̄, 1}, {q, 2}, {G, 3}}	$i\, gs\, \gamma^{\mu_3}_{s_1,s_2}\, T^{a_3}_{m_1,m_2}$

✤ Field rotations have been performed automatically

Implementing the matter Lagrangian (3)

✦ The D-terms

$$\mathcal{L}_{\rm matter} = -\frac{g_s^2}{2}\Big[-\tilde{q}_L^\dagger T^a \tilde{q}_L + \tilde{q}_R^\dagger T^a \tilde{q}_R\Big]\Big[-\tilde{q}_L^\dagger T^a \tilde{q}_L + \tilde{q}_R^\dagger T^a \tilde{q}_R\Big]$$

✦ The implementation in FEYNRULES is easy

```
LD := -1/2 gs^2 *
        (sqRbar[cc1] T[a,cc1,cc2] sqR[cc2] - sqLbar[cc1] T[a,cc1,cc2] sqL[cc2]) *
        (sqRbar[cc3] T[a,cc3,cc4] sqR[cc4] - sqLbar[cc3] T[a,cc3,cc4] sqL[cc4]);
```

✤ Repeated indices (*cc1, cc2, cc3, cc4*) are summed

✤ A single index can only be used twice

✦ We can compute the Feynman rules

{{sq1, 1}, {sq1, 2}, {sq1†, 3}, {sq1†, 4}}	$-i\, gs^2 \, \mathrm{Cos}[2\,\mathrm{theta}]^2 \left(T^{Gluon\$1}_{m_3,m_2} T^{Gluon\$1}_{m_4,m_1} + T^{Gluon\$1}_{m_3,m_1} T^{Gluon\$1}_{m_4,m_2}\right)$
{{sq1, 1}, {sq1†, 2}, {sq1†, 3}, {sq2, 4}}	$\frac{1}{2}\, i\, gs^2 \, \mathrm{Sin}[4\,\mathrm{theta}] \left(T^{Gluon\$1}_{m_2,m_4} T^{Gluon\$1}_{m_3,m_1} + T^{Gluon\$1}_{m_2,m_1} T^{Gluon\$1}_{m_3,m_4}\right)$
{{sq1†, 1}, {sq1†, 2}, {sq2, 3}, {sq2, 4}}	$-i\, gs^2 \, \mathrm{Sin}[2\,\mathrm{theta}]^2 \left(T^{Gluon\$1}_{m_1,m_4} T^{Gluon\$1}_{m_2,m_3} + T^{Gluon\$1}_{m_1,m_3} T^{Gluon\$1}_{m_2,m_4}\right)$
{{sq1, 1}, {sq1, 2}, {sq1†, 3}, {sq2†, 4}}	$\frac{1}{2}\, i\, gs^2 \, \mathrm{Sin}[4\,\mathrm{theta}] \left(T^{Gluon\$1}_{m_3,m_2} T^{Gluon\$1}_{m_4,m_1} + T^{Gluon\$1}_{m_3,m_1} T^{Gluon\$1}_{m_4,m_2}\right)$
{{sq1, 1}, {sq1†, 2}, {sq2, 3}, {sq2†, 4}}	$-i\, gs^2 \left(\mathrm{Sin}[2\,\mathrm{theta}]^2 \, T^{Gluon\$1}_{m_2,m_3} T^{Gluon\$1}_{m_4,m_1} - \mathrm{Cos}[2\,\mathrm{theta}]^2 \, T^{Gluon\$1}_{m_2,m_1} T^{Gluon\$1}_{m_4,m_3}\right)$
{{sq1†, 1}, {sq2, 2}, {sq2, 3}, {sq2†, 4}}	$-\frac{1}{2}\, i\, gs^2 \, \mathrm{Sin}[4\,\mathrm{theta}] \left(T^{Gluon\$1}_{m_1,m_3} T^{Gluon\$1}_{m_4,m_2} + T^{Gluon\$1}_{m_1,m_2} T^{Gluon\$1}_{m_4,m_3}\right)$
{{sq1, 1}, {sq1, 2}, {sq2†, 3}, {sq2†, 4}}	$-i\, gs^2 \, \mathrm{Sin}[2\,\mathrm{theta}]^2 \left(T^{Gluon\$1}_{m_3,m_2} T^{Gluon\$1}_{m_4,m_1} + T^{Gluon\$1}_{m_3,m_1} T^{Gluon\$1}_{m_4,m_2}\right)$
{{sq1, 1}, {sq2, 2}, {sq2†, 3}, {sq2†, 4}}	$-\frac{1}{2}\, i\, gs^2 \, \mathrm{Sin}[4\,\mathrm{theta}] \left(T^{Gluon\$1}_{m_3,m_2} T^{Gluon\$1}_{m_4,m_1} + T^{Gluon\$1}_{m_3,m_1} T^{Gluon\$1}_{m_4,m_2}\right)$
{{sq2, 1}, {sq2, 2}, {sq2†, 3}, {sq2†, 4}}	$-i\, gs^2 \, \mathrm{Cos}[2\,\mathrm{theta}]^2 \left(T^{Gluon\$1}_{m_3,m_2} T^{Gluon\$1}_{m_4,m_1} + T^{Gluon\$1}_{m_3,m_1} T^{Gluon\$1}_{m_4,m_2}\right)$

✤ All nine vertices automatically derived from the (very compact) Lagrangian

Implementing the matter Lagrangian (4)

- The squark-quark-gluino interactions

$$\mathcal{L}_{\rm matter} = \sqrt{2} g_s \Big[- \tilde{q}_L^\dagger T^a \big(\tilde{g}^a P_L q\big) + \big(\bar{q} P_L \tilde{g}^a\big) T^a \tilde{q}_R \Big] + {\rm h.c.}$$

- The implementation in FEYNRULES is again easy

```
Lgosqq :=  Sqrt[2] gs ProjM[s1,s2] * (
   - sqLbar[cc1] T[a,cc1,cc2] gobar[s1,a].q[s2,cc2] +
     qbar[s1,cc1].go[s2,a] T[a,cc1,cc2] sqR[cc2]);
```

  - All indices must this time be explicit (the scalars cannot be included in a fermion chain)
  - *ProjM* denotes the left-handed chirality projector (*ProjP* is the right-handed one)

- We can compute the Feynman rules (including the Hermitian conjugate pieces, with HC)

```
In[37]:= Simplify[FeynmanRules[Lgosqq + HC[Lgosqq]]] // MatrixForm
Starting Feynman rule calculation.
Expanding the Lagrangian...
Collecting the different structures that enter the vertex.
4 possible non-zero vertices have been found -> starting the computation: 4 / 4.
4 vertices obtained.
Out[37]//MatrixForm=
```

$$\begin{pmatrix} \{\{\bar{q},1\},\{go,2\},\{sq1,3\}\} & -i\sqrt{2}\,gs\,(Cos[theta]\,P_{+s_1,s_2} - P_{-s_1,s_2}\,Sin[theta])\,T^{a2}_{m_1,m_3} \\ \{\{go,1\},\{q,2\},\{sq1^\dagger,3\}\} & -i\sqrt{2}\,gs\,(Cos[theta]\,P_{-s_1,s_2} - P_{+s_1,s_2}\,Sin[theta])\,T^{a1}_{m_3,m_2} \\ \{\{\bar{q},1\},\{go,2\},\{sq2,3\}\} & i\sqrt{2}\,gs\,(Cos[theta]\,P_{-s_1,s_2} + P_{+s_1,s_2}\,Sin[theta])\,T^{a2}_{m_1,m_3} \\ \{\{go,1\},\{q,2\},\{sq2^\dagger,3\}\} & i\sqrt{2}\,gs\,(Cos[theta]\,P_{+s_1,s_2} + P_{-s_1,s_2}\,Sin[theta])\,T^{a1}_{m_3,m_2} \end{pmatrix}$$

From FeynRules to phenomenology

- The SUSY-QCD model has been implemented into FeynRules

```
LMatter := Lkin + LD + Lgosqq + HC[Lgosqq];

LSUSYQCD := LVector1 + LMatter;
```

- The Feynman rules have been extracted (and checked)
- We are now ready to export the model to one or several Monte Carlo tools
  - CalcHep / CompHep
  - FeynArts / FormCalc
  - Sherpa
  - UFO ➢ MadGraph5_aMC@NLO
  - Whizard / Omega

```
WriteCHOutput[LSUSYQCD]
WriteFeynArtsOutput[LSUSYQCD]
WriteSHOutput[LSUSYQCD]
WriteUFO[LSUSYQCD]
WriteWOOutput[LSUSYQCD]
```

Limitations and fine prints

- **Particle / parameter names**
  - The strong interaction has a special role
    - ★ Name for the gluon field, the coupling constant, *etc*.
    - ★ Which parameter is internal/external
    - ★ The numerical value of α_s at the Z-pole
  - For some generators, the electroweak interaction has also a special role
    - ★ Name for the Fermi coupling, the Z-boson mass
    - ★ Which parameter is external/internal
    - ★ At which scale must the numerical values be given

- **Color structures: not supported in full generality**
  - The interfaces discard the non-supported vertices
  - Representations handled by the FEYNRULES interfaces:
    - ★ CALCHEP: 1, 3, 8 (but limited)
    - ★ FEYNARTS: all
    - ★ MADGRAPH5_AMC@NLO: 1, 3, 6, 8
    - ★ SHERPA: 1, 3, 8
    - ★ WHIZARD: 1, 3, 8

- **Lorentz structures and spins: not supported in full generality**
  - The interfaces discard the non-supported vertices
  - Representations handled by the FEYNRULES interfaces:
    - ★ CALCHEP: MSSM-like Lorentz structures; spin 0,1/2,1,3/2, 2
    - ★ FEYNARTS: all Lorentz structures; spin 0,1/2,1
    - ★ MADGRAPH5_AMC@NLO: all Lorentz structures; spin 0,1/2,1,3/2,2
    - ★ SHERPA: SM-like Lorentz structures; spin 0,1/2,1
    - ★ WHIZARD: MSSM-like Lorentz structures; spin 0,1/2,1,2

From FEYNRULES to MADGRAPH 5: the UFO (I)

✦ The UFO in a nutshell

- ✤ UFO ≡ Universal FEYNRULES output (not tied to any Monte Carlo tool)
- ✤ Allows the model to contain generic color and Lorentz structures
- ✤ The FEYNRULES interface creates a PYTHON module to be linked to any code
- ✤ This module contains all the model information
- ✤ More information ➢ Degrande, Duhr, BenjFuks, Grellscheid, Mattelaer, Reiter [CPC 183 (2012) 1201]

✦ The UFO version of the model can be used by several programs

- ✤ ALOHA
- ✤ GOSAM
- ✤ HERWIG++
- ✤ MADANALYSIS 5
- ✤ MADGRAPH5_AMC@NLO

✦ Used with a copy-paste

```
In[6]:= WriteUFO[LSUSYQCD]
 --- Universal FeynRules Output (UFO) v 1.1 ---
Warning: no electric charge defined. Putting all electric charges to zero.
Starting Feynman rule calculation.
Expanding the Lagrangian...
Collecting the different structures that enter the vertex.
25 possible non-zero vertices have been found -> starting the computation: 25 / 25.
21 vertices obtained.
Flavor expansion of the vertices: 21 / 21
   - Saved vertices in InterfaceRun[ 1 ].
Computing the squared matrix elements relevant for the 1->2 decays:
4 / 4
Squared matrix elent compute in 0.112652 seconds.
Decay widths computed in 0.002791 seconds.
Preparing Python output.
    - Splitting vertices into building blocks.
    - Optimizing: 21/21 .
    - Writing files.
Done!
```

From FEYNRULES to MADGRAPH 5: the UFO (2)

✦ Particles

- Similar to the FEYNRULES model format, but with different names for the attributes (masses, widths, PDG codes, *etc.*)
- Antiparticles are automatically derived from the knowledge of the corresponding particle
- Special keyword for the (numerical value) zero
- Note: the embedding of the color and spin representation

```
G = Particle(pdg_code = 21,
             name = 'G',
             antiname = 'G',
             spin = 3,
             color = 8,
             mass = Param.ZERO,
             width = Param.ZERO,
             texname = 'G',
             antitexname = 'G',
             charge = 0)

go = Particle(pdg_code = 1000021,
              name = 'go',
              antiname = 'go',
              spin = 2,
              color = 8,
              mass = Param.Mgo,
              width = Param.Wgo,
              texname = 'go',
              antitexname = 'go',
              charge = 0)
```

```
sq1 = Particle(pdg_code = 1000006,
               name = 'sq1',
               antiname = 'sq1~',
               spin = 1,
               color = 3,
               mass = Param.Msq1,
               width = Param.Wsq1,
               texname = 'sq1',
               antitexname = 'sq1~',
               charge = 0)

sq1__tilde__ = sq1.anti()

sq2 = Particle(pdg_code = 2000006,
               name = 'sq2',
               antiname = 'sq2~',
               spin = 1,
               color = 3,
               mass = Param.Msq2,
               width = Param.Wsq2,
               texname = 'sq2',
               antitexname = 'sq2~',
               charge = 0)

sq2__tilde__ = sq2.anti()
```

```
q = Particle(pdg_code = 6,
             name = 'q',
             antiname = 'q~',
             spin = 2,
             color = 3,
             mass = Param.Mq,
             width = Param.Wq,
             texname = 'q',
             antitexname = 'q~',
             charge = 0)

q__tilde__ = q.anti()
```

From FEYNRULES to MADGRAPH 5: the UFO (3)

✦ Parameters

- Similar to the FEYNRULES model format, but with different names
- Note: the automated generation of a LesHouches structure for the external parameters (in particular for the masses and widths)
- PYTHON-compliant formula for the internal parameters

```
aS = Parameter(name = 'aS',
               nature = 'external',
               type = 'real',
               value = 0.1184,
               texname = '\\text{aS}',
               lhablock = 'FRBlock',
               lhacode = [ 1 ])

theta = Parameter(name = 'theta',
                  nature = 'external',
                  type = 'real',
                  value = 0.7853981633974483,
                  texname = '\\theta',
                  lhablock = 'FRBlock',
                  lhacode = [ 2 ])
```

```
Mgo = Parameter(name = 'Mgo',
                nature = 'external',
                type = 'real',
                value = 500,
                texname = '\\text{Mgo}',
                lhablock = 'MASS',
                lhacode = [ 1000021 ])

Wq = Parameter(name = 'Wq',
               nature = 'external',
               type = 'real',
               value = 1.50833649,
               texname = '\\text{Wq}',
               lhablock = 'DECAY',
               lhacode = [ 6 ])
```

```
G = Parameter(name = 'G',
              nature = 'internal',
              type = 'real',
              value = '2*cmath.sqrt(aS)*cmath.sqrt(cmath.pi)',
              texname = 'G')
```

From FEYNRULES to MADGRAPH 5: the UFO (4)

✦ Vertices

- Decomposed in a spin x color basis
  - ★ For instance, the quartic gluon vertex can be written as

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

- Each element of the decomposition is stored separately

★ vertices.py: contains the decomposition, the color basis and generic elements for the couplings and the spin basis

```
V_2 = Vertex(name = 'V_2',
             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.VVVV1, L.VVVV2, L.VVVV3 ],
             couplings = {(1,1):C.GC_4,(0,0):C.GC_4,(2,2):C.GC_4})
```

★ lorentz.py: contains each necessary element of the spin basis (reused across vertices as much as possible)

```
VVVV1 = Lorentz(name = 'VVVV1',
                spins = [ 3, 3, 3, 3 ],
                structure = 'Metric(1,4)*Metric(2,3) - Metric(1,3)*Metric(2,4)')

VVVV2 = Lorentz(name = 'VVVV2',
                spins = [ 3, 3, 3, 3 ],
                structure = 'Metric(1,4)*Metric(2,3) - Metric(1,2)*Metric(3,4)')

VVVV3 = Lorentz(name = 'VVVV3',
                spins = [ 3, 3, 3, 3 ],
                structure = 'Metric(1,3)*Metric(2,4) - Metric(1,2)*Metric(3,4)')
```

★ couplings.py: the coupling strengths (reused across vertices as much as possible)

```
GC_4 = Coupling(name = 'GC_4',
                value = 'complex(0,1)*G**2',
                order = {'QCD':2})
```

Outline

Advanced techniques for implementing models in FeynRules

- Extension / restriction of existing models
- The superspace module of FeynRules
- Mass diagonalization
- Two-body decays
- Next-to-leading order module

Merging and extending model implementations

- ✦ Many BSM models of interest are simple extensions of another model
  - ✤ FEYNRULES allows one to start from a given model
    - ★ Add new particles, parameters, Lagrangian terms
    - ★ Modify existing particles, parameters, Lagrangian terms
    - ★ Remove some particles, parameters, Lagrangian terms
  - ✤ Special cases very relevant for LHC physics: Simplified Model Spectra
    - ★ The Standard Model + one or two new particles
    - ★ Often inspired by the MSSM
    - ★ Example: the SM + lightest stop and neutralino + the relevant subset of MSSM interactions

- ✦ The merged FEYNRULES model contains two .fr files
  - ✤ The parent model implementation
  - ✤ One extra file with the modifications
  - ✤ They must be loaded together (the parent model first)

```
LoadModel["SM.fr", "stops.fr"];
```

    - ★ No need to re-implement what is common (gauge groups, *etc.*)

- ✦ One can start from any of the models available on the FEYNRULES model database
  http://feynrules.irmp.ucl.ac.be

The FEYNRULES model database

- About 40 models are available online
  - Simple extensions of the Standard Model
    - ★ Simplified model spectra
    - ★ Four generation models
    - ★ Vector-like quarks
    - ★ Two-Higgs-Doublet Models, Hidden Abelian Higgs
    - ★ *etc.*
  - Supersymmetric models
    - ★ MSSM with and without R-parity
    - ★ The NMSSM
    - ★ R-symmetric supersymmetric models
    - ★ Left-right supersymmetric models
  - Extra-dimensional models
    - ★ Universal extra-dimensions
    - ★ Large extra-dimensions
    - ★ Heidi, Minimal Higgsless models
    - ★ Randall-Sundrum
  - Strongly coupled and effective field theories
    - ★ Technicolor
    - ★ Models with dimension-six and dimension-eight operators
    - ★ *etc.*

Available models

- Standard Model
- Simple extensions of the SM (18)
- Supersymmetric Models (5)
- Extra-dimensional Models (4)
- Strongly coupled and effective field theories (8)
- Miscellaneous (0)

Restricting model implementations

- Many BSM models of interest are subset of other
  - Equivalent to the parent model, but with some parameters set to specific values (0 or 1)
    - ★ Example 1: the massless version of a model (massless light quarks in the Standard Model)
    - ★ Example 2: a mixing matrix set to the identity (no-CKM matrix in the Standard Model)
  - FeynRules allows one to start from a given model
    - ★ Write the restrictions under the form of a list of Mathematica replacement rules (M$Restrictions)
    - ★ Read them into FeynRules
    - ★ Apply them before the computation of any Feynman rule

- The output Feynman rules (and this Monte Carlo model files)
  - Are free from the restricted parameters
  - Smaller files, more efficiency at the Monte Carlo level
    - ★ Example: the general MSSM has more than 10000 vertices; its flavor-conserving version only ~1000

- One practical example: the Standard Model without CKM-mixing

```
M$Restrictions = {
            CKM[i_,i_] -> 1,
            CKM[i_?NumericQ, j_?NumericQ] :> 0 /; (i =!= j),
            cabi -> 0
}
```

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

The supersymmetry module

✦ A module dedicated to calculations in superspace
- Superfield declaration and links to the component fields
- Series expansion in terms of component fields
- Automatic derivation of supersymmetric Lagrangians
- Solving the equations of motion of the unphysical fields
- Many extra built-in functions
- See: Duhr, BenjFuks [CPC 182 (2011) 2404]; BenjFuks [IJMPA 27 (2012) 1230007]

A left-handed squark superfield

```
CSF[1] == {
  ClassName      -> QL,
  Chirality      -> Left,
  Weyl           -> qLw,
  Scalar         -> sqL,
  Indices        -> {Index[Colour]}
}
```

✦ Supersymmetric model implementation
- Declaration of the model gauge group
- Declaration of all fields and superfields
- Declaration of all model parameters
- Can the writing of the Lagrangian be simplified?

✦ Supersymmetric Lagrangian in superspace are very compact

$$\mathcal{L} = \Phi^\dagger e^{-2gV}\Phi_{|_{\theta^2\bar\theta^2}} + \frac{1}{16g^2\tau_{\mathcal{R}}}\mathrm{Tr}(W^\alpha W_\alpha)_{|_{\theta^2}} + \frac{1}{16g^2\tau_{\mathcal{R}}}\mathrm{Tr}(\bar W_{\dot\alpha}\bar W^{\dot\alpha})_{|_{\bar\theta^2}}$$

$$+ W(\Phi)_{|_{\theta^2}} + W^\star(\Phi^\dagger)_{|_{\bar\theta^2}} + \mathcal{L}_{\rm soft}$$

➢ Model independent ⇒ can be automated

➢ Model dependent ⇒ to be provided

- First line: kinetic and gauge interaction terms for all fields
- Second line: model dependent superpotential and supersymmetry breaking Lagrangian

See the manual for more details on the superspace module

Implementing supersymmetric Lagrangians

✦ Supersymmetric Lagrangian in superspace are very compact

$$\mathcal{L} = \Phi^\dagger e^{-2gV}\Phi_{|_{\theta^2\bar\theta^2}} + \frac{1}{16g^2\tau_\mathcal{R}}\text{Tr}(W^\alpha W_\alpha)_{|_{\theta^2}} + \frac{1}{16g^2\tau_\mathcal{R}}\text{Tr}(\bar W_{\dot\alpha}\bar W^{\dot\alpha})_{|_{\bar\theta^2}}$$
$$+ W(\Phi)_{|_{\theta^2}} + W^\star(\Phi^\dagger)_{|_{\bar\theta^2}} + \mathcal{L}_\text{soft}$$

- First line: kinetic and gauge interaction terms for all fields
  - ★ Model independent ⇒ can be automated (CSFKineticTerms and VSFKineticTerms functions)
  - ★ Dedicated methods to access the components of a superfield (Theta2Component, *etc*.)

```
Lag = Theta2Thetabar2Component[ CSFKineticTerms[] ] +
   Theta2Component[ VSFKineticTerms[] ] +
   Thetabar2Component[ VSFKineticTerms[] ];
```

- Second line: superpotential and supersymmetry-breaking terms
  - ★ Model dependent ⇒ to be provided by the user (*SuperW* and *LSoft*)

```
Lag2 = LSoft + Theta2Component[ SuperW ] +  Thetabar2Component[ SuperW ];
```

✦ The Lagrangian above must be post-processed

- Solving the equation of motion for the auxiliary fields and inserting the solution back into the Lagrangian
  - ★ Automated (SolveEqMotionF and SolveEqMotionD)
- Replacement of Weyl spinors in terms of Majorana and Dirac spinors
  - ★ Automated (WeylToDirac)
- Rotation to the mass basis
  - ★ Standard FEYNRULES function (ExpandIndices)

See the manual for more details on the superspace module

Mass matrices and their diagonalization

✦ The problematics of the mass matrices

- Lagrangians are usually easily written in the gauge basis
- The included mass matrices are thus in general non-diagonal ➢ diagonalization required
- The gauge basis of fields must be rotated to the mass basis where the mass matrices are diagonal
- This diagonalization cannot in general be achieved analytically

✦ The ASPERGE package of FEYNRULES

- A MATHEMATICA module allowing one to extract the mass matrices from the Lagrangian
- A generator of C++ code ➢ numerical diagonalization of all the model mass matrices (the generated code can be used in a standalone way)
- See: Alloul, D'Hondt, De Causmaecker, BenjFuks, Rausch de Traubenberg [EPJC 73 (2013) 2325]

✦ Example: the Z-boson and photon in the Standard Model

- Each mixing is declared as a set of replacement rules (in *M$MixingsDescription*)
- Each rule represent a property of the mixing relation

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

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

- ASPERGE can compute the mass matrices:
- ASPERGE can generate its standalone C++ version:

```
ComputeMassMatrix[Lag];
WriteASperGe[Lag];
```

See the manual for more details on the ASPERGE module

Two-body decays

✦ The problematics of the decay widths and branching ratios

- ✤ Some Monte Carlo tools need the decay table (widths and branching ratios) to decay particles
- ✤ Widths and branching ratios are not independent quantities ➢ need to be calculated
- ✤ Some Monte Carlo tools compute these quantities on the fly
  - ➢ the procedure is repeated each time it is needed
- ✤ FeynRules offers a way to include analytical information on the two-body decay

✦ The decay module of FEYNRULES

- ✤ Two body decays can be directly read from the three-point vertices ($\mathcal{V}$) of the model

$$\Gamma_{1\to 2} = \frac{1}{2|M|S} \int \mathrm{d}\Phi_N \, |\mathcal{M}_{1\to 2}|^2 = \frac{\sqrt{\lambda(M^2, m_1^2, m_2^2)}}{16\,\pi\, S\, |M|^3} \mathcal{V}^{a_1 a_2 a_3}_{\ell_1 \ell_2 \ell_3} \, \mathcal{P}_1^{\ell_1 \ell_1'} \, \mathcal{P}_2^{\ell_2 \ell_2'} \, \mathcal{P}_3^{\ell_3 \ell_3'} \, (\mathcal{V}^*)^{a_1 a_2 a_3}_{\ell_1' \ell_2' \ell_3'}$$

  - ★ Partial width for the decay of a particle of mass M to two particles of masses m_1 and m_2
  - ★ Includes a symmetry factor S and $\mathcal{P}$ denotes the polarization tensor of each particle
- ✤ FEYNRULES makes use of MATHEMATICA to compute all partial widths of the model
  - ★ Ignores open and closed channels ➢ benchmark independent
  - ★ The information is exported to the UFO (already used by MADWIDTH, the decay module of MADGRAPH5)

The decay module of FEYNRULES

✦Automatic decay width computations

- ✤ All two-body decay widths can be easily computed from the Lagrangian

```
verts      = FeynmanRules[Lag];
vertsexp   = FlavorExpansion[verts];
results    = ComputeWidths[vertsexp];
```

- ✤ Many functions available for analytical calculations (PartialWidth, TotWidth, BranchingRatio)
- ✤ The numerical value provided for the particle widths can be updated accordingly (UpdateWidths)
- ✤ See: Alwall, Duhr, BenjFuks, Mattelaer, Öztürk, Shen [arXiv:1402.1178]

✦The information is (by default) employed at the UFO interface level

- ✤ Can be turned of: (AddDecays → False)
- ✤ The UFO contains an extra file *decays.py*
- ✤ Example of the Standard Model UFO: the top quark

```
Decay_t = Decay(name = 'Decay_t',
                particle = P.t,
                partial_widths = {(P.W__plus__,P.d):'((MT**2 - MW**2)*((3*CKM3x1*ee**2*MT**2*complexconjugate(CKM3x1))/(2.*sw**2) +
(3*CKM3x1*ee**2*MT**4*complexconjugate(CKM3x1))/(2.*MW**2*sw**2) - (3*CKM3x1*ee**2*MW**2*complexconjugate(CKM3x1))/sw**2))/
(96.*cmath.pi*abs(MT)**3)',
                                  (P.W__plus__,P.s):'((MT**2 - MW**2)*((3*CKM3x2*ee**2*MT**2*complexconjugate(CKM3x2))/(2.*sw**2) +
(3*CKM3x2*ee**2*MT**4*complexconjugate(CKM3x2))/(2.*MW**2*sw**2) - (3*CKM3x2*ee**2*MW**2*complexconjugate(CKM3x2))/sw**2))/
(96.*cmath.pi*abs(MT)**3)',
                                  (P.W__plus__,P.b):'(((3*CKM3x3*ee**2*MB**2*complexconjugate(CKM3x3))/(2.*sw**2) +
(3*CKM3x3*ee**2*MT**2*complexconjugate(CKM3x3))/(2.*sw**2) + (3*CKM3x3*ee**2*MB**4*complexconjugate(CKM3x3))/(2.*MW**2*sw**2) -
(3*CKM3x3*ee**2*MB**2*MT**2*complexconjugate(CKM3x3))/(MW**2*sw**2) + (3*CKM3x3*ee**2*MT**4*complexconjugate(CKM3x3))/(2.*MW**2*sw**2) -
(3*CKM3x3*ee**2*MW**2*complexconjugate(CKM3x3))/sw**2)*cmath.sqrt(MB**4 - 2*MB**2*MT**2 + MT**4 - 2*MB**2*MW**2 - 2*MT**2*MW**2 +
MW**4))/(96.*cmath.pi*abs(MT)**3)'})
```

See the manual for more details on the decay module

FEYNRULES @ NLO

✦ Ingredients for a next-to-leading order calculation

- Tree-level vertices
- UV counterterms
- R_2 counterterms (depending on the NLO event generator)

✦ The R_2 counterterms

- One-loop modules are based on unitarity approaches
- This requires the computation of the rational parts of the loops (R_1 and R_2)
- The R_2 counterterm can be calculated once and for all for each model
  - ★ From the knowledge of the tree-level Lagrangian
  - ★ Using special Feynman rules (numerators in ε dimensions; denominators in D dimensions)

✦ Technical details for the NLO module of FEYNRULES

- Automatic renormalization of the Lagrangian
- Use of the FEYNARTS interface of FEYNRULES
- Generation of a script for NLO diagram generation
  - ☑ R_2 and UV counterterms
- Export to the UFO
- See: Degrande [arXiv:1406.3030]

Outline

Summary

✦ The quest for new physics at the LHC has started
- ✤ Rely on Monte Carlo event generators for background and signal modeling
- ✤ FEYNRULES facilitates the implementation of new physics models in those tools

✦ FEYNRULES: http://feynrules.irmp.ucl.ac.be
- ✤ Straightforward implementation of new physics model in Monte Carlo tools
  - ★ Interfaces to many programs
- ✤ FEYNRULES is shipped with its own computational modules
  - ★ A superspace module
  - ★ A decay package
  - ★ A mass diagonalization module (ASPERGE)
  - ★ A brand new NLO module

Try it on with your favorite model!

From FEYNRULES to event analysis

0. Implementation of the model in FEYNRULES and generation of the Monte Carlo model files

1. Event generation with any Monte Carlo event generator

Next talk

- Both signal and backgrounds
- Precision in the normalization: (N)NLO inclusive results
- Generator choice: beware of restrictions (supported Lorentz and color structures)
  → e.g., MADGRAPH5_AMC@NLO

2. Parton showering and hadronization

Next-to-next talk

- Precision in the shapes: multiparton matrix-element merging techniques
  → e.g., MADGRAPH 5 + PYTHIA (automated merging)

3. Detector simulation

later this week

4. Event analysis with MADANALYSIS 5

later this week

- Parton-level, hadron-level and reconstructed-level analyses (and more)