

Beyond the Standard Model phenomenology with FEYNRULES

Fuks Benjamin

LPTHE - CNRS - UPMC

**Dartmouth-TRIUMF-University of Washington
High-Energy Physics/Cosmology Tools Bootcamp**

October 24, 2017

Outline

1. FEYNRULES in a nutshell
2. Implementing supersymmetric QCD in FEYNRULES
3. Using FEYNRULES with supersymmetric QCD model
4. Advanced model implementation techniques
5. Summary

The quest for new physics at the LHC

- ◆ The quest for physics beyond the Standard Model at the LHC is on-going
 - ♣ How to get hints of new physics?
 - ★ Confront data to the Standard Model expectation in search channels
 - ★ Observe unexplained deviations at a good confidence level
 - ♣ Ingredient 1: predictions for the Standard Model background
 - ♣ **Ingredient 2: predictions for the new physics signals**

Monte Carlo tools and new physics

◆ Path towards the characterization of (potentially observed) new physics

- ❖ Getting information on the nature of an observation
 - ★ Fitting (and interpreting) deviations by some new physics signals
 - ★ Leading order Monte Carlo tools and techniques can do a proper job
- ❖ Final words on the nature of any potential new physics
 - ★ Accurate measurements of the model parameters
 - ★ More precise predictions are mandatory

◆ Monte Carlo tools play a key role at each step

- ❖ Prospective collider studies (*a priori* preparation)
 - ★ Study of various new physics signal
- ❖ Reinterpretation of existing results (*a posteriori* reactions to announcements)
 - ★ LHC recasting in new theoretical context

**New physics simulations
for the LHC are crucial**

New physics simulations: the challenges

◆ But what are the challenges with respect to new physics simulations?

- ❖ Theoretically, we are still in the dark
 - ★ No sign of new physics
 - ★ All measurements are Standard-Model-like
- ❖ There is not any leading new physics candidate theory
 - ★ Plethora of models to implement in the tools

◆ New physics is a standard in many tools today

- ❖ Result of 20 years of developments
- ❖ NLO and loop-induced processes can now be simulated automatically
- ❖ BSM precision simulations are standard in MG5_aMC@NLO

New physics simulations: the 'how-to'

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

- ★ Model definition: particles, parameters & vertices (\equiv 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 tasks** (from Feynman 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

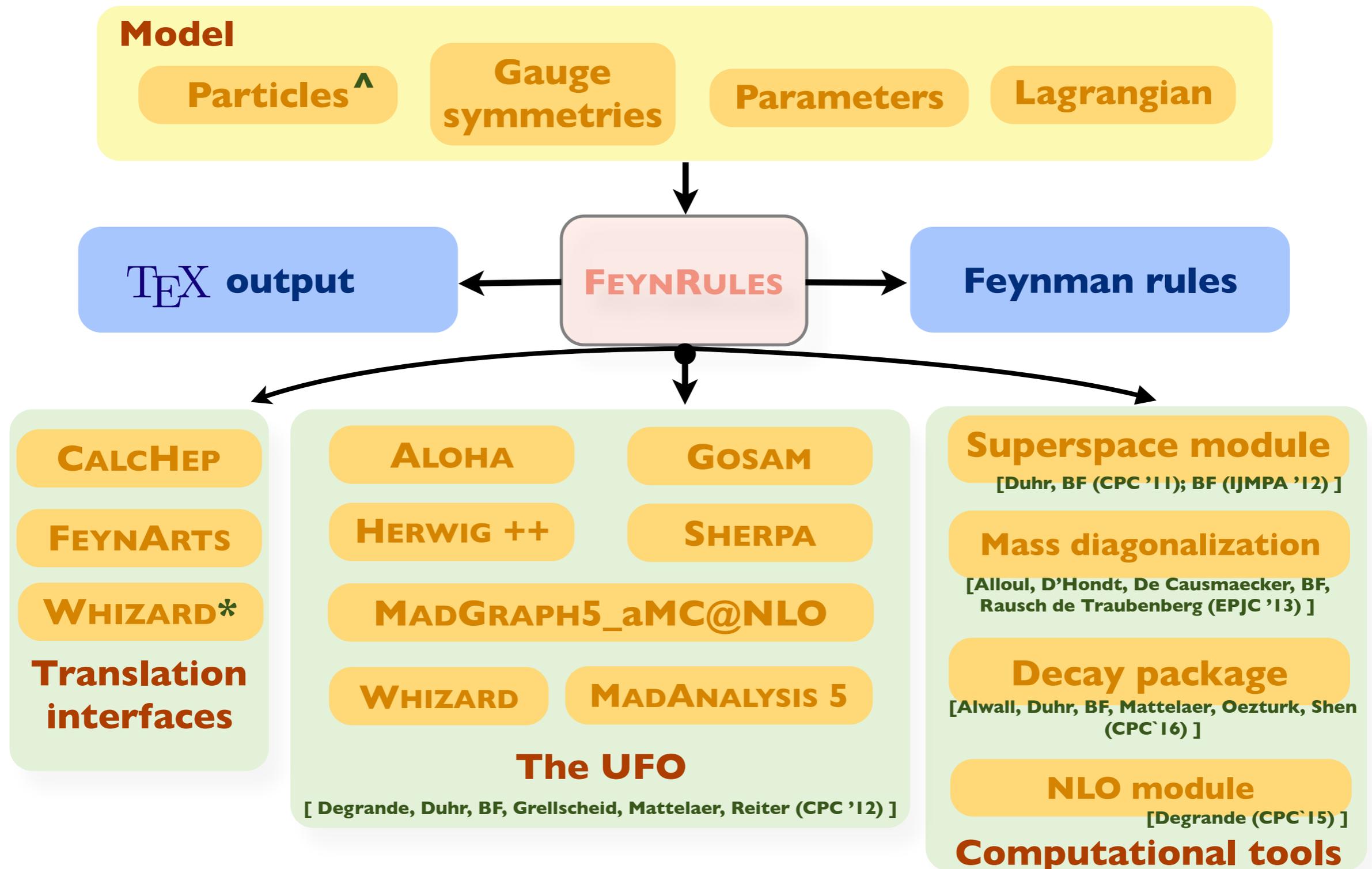
- ❖ **Validation** of an implementation using several Monte Carlo programs

◆ 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, BF (CPC'14)]



* Whizard interface: Christensen, Duhr, BF, Reuter, Speckner (EPJC '12)

[^] Support for spin 3/2: Christensen, de Aquino, Deutschmann, Duhr, BF, Garcia-Cely, Mattelaer, Mawatari, Oexl, Takaesu (EPJC '13)

Outline

1. FEYNRULES in a nutshell
- 2. Implementing supersymmetric QCD in FEYNRULES**
3. Using FEYNRULES with supersymmetric QCD model
4. Advanced model implementation techniques
5. Summary

Supersymmetric QCD: particle content

◆ Particle content

- ❖ **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**

(Broken) supersymmetric QCD: the model

◆ The dynamics of the model is embedded in the Lagrangian

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

- ♣ Kinetic terms for all fields (first line)
- ♣ Mass terms for the squarks, quarks and gluino (second line)
- ♣ Supersymmetric gauge interactions for all fields (last two lines)

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 authors, *etc.*
- ❖ Reference publications used can be added
- ❖ Webpage information can be added

```
M$ModelName = "SUSYQCD";

M$Information = {
  Authors      -> {"Benjamin Fuks"},
  Date         -> "23.10.17",
  Version      -> "1.0.0",
  Institutions -> {"LPTHE / UPMC"},
  Emails       -> {"fuks@lpthe.jussieu.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

```
IndexRange[Index[Gluon]] = NoUnfold[Range[8]];
IndexRange[Index[Colour]] = NoUnfold[Range[3]];
```

❖ QCD has a special role in MC event generators ➤ many special names

- ★ *Colour, Sextet and Gluon* for the color 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

- ❖ Fundamental indices starting with the letter *m*
- ❖ Adjoint indices starting with the letter *a*

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

The declaration of the gauge group (I)

◆ Each element the group is declared in the $M\$GaugeGroups$ list

❖ A declaration \equiv a set of MATHEMATICA replacement rules

❖ In our SUSY-QCD model:

★ We must only declare $SU(3)_C$: we choose the name $SU3C$

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

❖ Each rule represents one group property (QCD: special names exist)

★ *Abelian*: abelian or non-abelian group

★ *GaugeBoson*: associated gauge boson

★ *CouplingConstant*, *StructureConstant*: coupling and the structure constants

★ *Representations*: list of 2-tuples linking indices to representation matrices

See the manual for more details on gauge groups

The declaration of the gauge group (2)

◆ Advantages of a proper gauge group declaration

- ❖ Render the writing of the Lagrangian easier:
 - ★ **Covariant derivatives** ($DC[\text{field}, \text{Lorentz index}]$)
 - ★ **Field strength tensors** ($FS[\text{field}, \text{Lorentz index } 1, \text{Lorentz index } 2]$)
 - ★ Useful for Lagrangian building in superspace (briefly covered in the last slides)
 - [Duhr & BF CPC 182 (2011) 2404; BF IJMPA 27 (2012) 1230007]
- ❖ Example: gluon and gluino kinetic terms

$$-\frac{1}{4}g_{\mu\nu}g^{\mu\nu} + \frac{i}{2}\bar{g}\not{D}g$$

$$-1/4 \text{ FS}[G, \mu, \nu, a] \text{ FS}[G, \mu, \nu, a] + \\ 1/2 \text{ gobar.Ga}[\mu].\text{DC}[\text{go}, \mu]$$

Declaring the gluon field

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

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

- ♣ Each rule represents a property of the field
 - ★ Vector field \triangleright the label is $V[I]$ (with V , and not F, S, R, T , etc.)
 - ★ **Classname**: defines the symbol to use in the Lagrangian $\triangleright G$
 - ★ **Indices**: the gluon lies in the adjoint representation of $SU(3)_c$
 - \triangleright The gluon has been previously set as the gauge boson of $SU(3)_c$
 - \triangleright The index (*Gluon*) is internally linked to the adjoint representation
 - ★ Other: vanishing mass and width, PDG code set to 21, self-conjugate

See the manual for more options for field declarations

Declaring a Dirac fermion 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**: defines two symbols to use in the Lagrangian ➤ *go* and *gobar*
- ★ **Mass and width**: two symbols and their associated numerical values

See the manual for more details on field declarations

Declaring the gluino field with Weyl fermions

◆ Second option: two-component spinors (Lagrangians sometimes easier)

```
W[1] == {
  ClassName      -> go|w,
  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
},
```

❖ Extra options are available

- ★ Two-component and four-component fermionic fields
 - the labels are **W[I]** (with a W) and **F[I]**
- ★ Weyl fermions are **unphysical** and linked to four-component fermions
 - **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

More on quarks: generation indices

◆ How to implement three generations of up-type quarks?

```
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}
},
```

❖ Slight differences for a bunch of options (+ new options)

★ **ClassMembers**: specify all the members of the class

★ We introduce a **generation index**: *Gen*

➤ **FlavorIndex** defines which index is the flavor index

★ **Mass, Width, PDG**: one for each class member (plus a generic mass symbol)

Declaring squark fields

◆ Extra elements in the *M\$ClassesDescription* list

```

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
}

```

❖ Nothing special

- ★ Scalar field ➤ the label is **S[3]**
- ★ **Classname**: defines pairs of symbols
➤ *sq1*, *sq2*, *sq1bar* and *sq2bar*
- ★ **Mass, width, Indices, etc.**: standard

**Missing:
squark mixing**

Mixing squark fields

◆ Mixing implemented via physical and unphysical fields

```

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]}
},

```

❖ Squark fields mix as

$$\begin{pmatrix} \tilde{q}_1 \\ \tilde{q}_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} \tilde{q}_L \\ \tilde{q}_R \end{pmatrix}$$

- ★ Left and right-handed squarks **unphysical**
 - removed from the Lagrangian and replaced
- ★ **Definitions** include the replacement rules (inversion of the above relation)
 - The rotations will be performed **automatically** by FEYNRULES
 - The Lagrangian can be written in the gauge basis (easier with sqL/sqR)

Parameters to implement

◆ The model requires to implement three parameters

♣ Masses and widths are handled automatically

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

♣ The strong coupling appears explicitly and implicitly in the Lagrangian

➤ g_s is needed

➤ α_s is also needed (required by the Monte Carlo tools)

♣ The squark mixing angle must be implemented θ

$$\begin{pmatrix} \tilde{q}_1 \\ \tilde{q}_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \tilde{q}_L \\ \tilde{q}_R \end{pmatrix}$$

Parameter declaration : strong couplings

◆ Parameters are declared as elements of the list *M\$Parameters*

- ♣ A declaration \equiv a set of MATHEMATICA replacement rules
- ♣ The two strong coupling parameters α_s and g_s are not independent

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

★ Internal and External parameters

- **External** \equiv free parameter
 \Rightarrow numerical value
- **Internal** \equiv dependent parameter
 \Rightarrow formula

★ InteractionOrder: specific to some MC

- more efficient diagram generation

★ ParameterName: QCD is special (in MC tools)

See the manual for more details on parameter declarations

Parameter declaration : squark mixing

◆ The squark mixing angle

- ♣ Nothing special (standard external parameter)

```
theta == {  
  ParameterType  -> External,  
  Value          -> Pi/4.  
}
```

◆ More options (assuming matrix-form mixing)

- ♣ Tensor parameters can be implemented too (Indices, Unitary)
- ♣ Complex parameters can be implemented too (ComplexParameter)

See the manual for more details on parameter declarations

Parameter organization

◆ Les Houches blocks can be specific to organizer the external parameters

- ❖ Use of a Les Houches parameter card by the MC programs
- ❖ Implemented via BlockName and OrderBlock
- ❖ If unspecified: everything goes into the 'FRBlock' block

◆ Complete α_s implementation as an example

- ❖ New options are used (TeX, Description, BlockName, OrderBlock)

```
aS == {
  ParameterType    -> External,
  BlockName        -> SMINPUTS,
  OrderBlock       -> 3,
  Value            -> 0.1184,
  InteractionOrder -> {QCD,2},
  TeX              -> Subscript[\[Alpha],s],
  Description      -> "Strong coupling constant at the Z pole"
},
```

Outline

1. FEYNRULES in a nutshell
2. Implementing supersymmetric QCD in FEYNRULES
- 3. Using FEYNRULES with supersymmetric QCD model**
4. Advanced model implementation techniques
5. Summary

Getting started

Starting a FEYNRULES MATHEMATICA session

◆ Step I: 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: 2.3.28 (21 May 2017).
Authors: A. Alloul, N. Christensen, C. Degrande, C. Duhr, B. Fuks

Please cite:
- Comput.Phys.Commun.185:2250-2300,2014 (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.

Loading the model implementation

◆ Step2: loading the SUSY-QCD implementation into FEYNRULES

- ♣ Moving to the right directory
- ♣ Loading the model itself

```
SetDirectory[NotebookDirectory[]];  
LoadModel["susyqcd.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 encoded in the preamble of the model are printed to the screen
- ♣ More information can be obtained by typing **ModelInformation[]**

The vector Lagrangian

The SUSYQCD vector Lagrangian

◆ The gauge sector of the theory : one 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

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

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

**Implementation of the
SUSYQCD vector
Lagrangian in FEYNRULES**

Implementing the vector Lagrangian

◆ Textbook expression:

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

◆ Makes use of the strengths of a proper gauge-group implementation

- ❖ Existence of shortcut functions (DC, FS, etc.)
- ❖ Very compact implementation

```
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;
```

★ All non-Lorentz indices understood (FEYNRULES automatically handles them)

◆ Indices can be noted explicitly too (no differences)

```
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];
```

Check of the vector Lagrangian implementation

◆ Textbook expression

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

◆ FEYNRULES implementation

```
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;
```

◆ Printing the vector Lagrangian

In[6]:= **LVector1**

```
Out[6]= -1/2 Mgo go . go + 1/2 i go . gamma^mu . (partial_mu [go] - i gs FSU3C^a$658 . go G_mu,a$658) -
        1/4 (-partial_nu [G_mu,a] + partial_mu [G_nu,a] + gs f_a,bb$656,cc$656 G_mu,bb$656 G_nu,cc$656)
        (-partial_nu [G_mu,a] + partial_mu [G_nu,a] + gs f_a,bb$657,cc$657 G_mu,bb$657 G_nu,cc$657)
```

◆ The output

- ♣ Covariant derivatives and field strength tensors have been automatically evaluated
- ♣ FSU3C denotes the structure constants of the SU3C gauge group
 - to be replaced in a second step by the f_{abc} .

Expanding the Lagrangian

◆ All indices can be restored automatically

In[7]:= `ExpandIndices [LVector1]`

$$\begin{aligned}
 \text{Out[7]} = & -\frac{1}{4} \partial_{\nu} [G_{\mu, a}]^2 + \frac{1}{2} \partial_{\nu} [G_{\mu, a}] \partial_{\mu} [G_{\nu, a}] - \frac{1}{4} \partial_{\mu} [G_{\nu, a}]^2 - \frac{1}{2} M g_0 g_{0_{i1667, i2667}} \cdot g_{0_{i1667, i2667}} + \\
 & \frac{1}{4} g_s \partial_{\nu} [G_{\mu, a}] f_{a, bb662, cc662} G_{\mu, bb662} G_{\nu, cc662} - \frac{1}{4} g_s \partial_{\mu} [G_{\nu, a}] f_{a, bb662, cc662} G_{\mu, bb662} G_{\nu, cc662} + \\
 & \frac{1}{4} g_s \partial_{\nu} [G_{\mu, a}] f_{a, bb663, cc663} G_{\mu, bb663} G_{\nu, cc663} - \frac{1}{4} g_s \partial_{\mu} [G_{\nu, a}] f_{a, bb663, cc663} G_{\mu, bb663} G_{\nu, cc663} - \\
 & \frac{1}{4} g_s^2 f_{a, bb662, cc662} f_{a, bb663, cc663} G_{\mu, bb662} G_{\mu, bb663} G_{\nu, cc662} G_{\nu, cc663} + \\
 & \frac{1}{4} i g_{0_{i667, i1669}} \cdot \partial_{\mu} [g_{0_{j667, i1669}}] \gamma_{i667, j667}^{\mu} - \\
 & \frac{1}{4} i g_s g_{0_{i670, i669}} \cdot g_{0_{j670, j669}} f_{a664, i669, j669} G_{\mu, a664} \gamma_{i670, j670}^{\mu} - \\
 & \frac{1}{4} i \partial_{\mu} [g_{0_{j667, i1669}}] \cdot g_{0_{i667, i1669}} \gamma_{j667, i667}^{\mu} + \frac{1}{4} i g_s g_{0_{j670, j669}} \cdot g_{0_{i670, i669}} f_{a664, i669, j669} G_{\mu, a664} \gamma_{j670, i670}^{\mu}
 \end{aligned}$$

- ❖ Kinetic terms (gluon and gluino)
- ❖ Mass terms (gluino)
- ❖ QCD interactions (with the proper structure constants)

Check of the implementation: hermiticity

◆ Textbook expression

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

◆ FEYNRULES implementation

```
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;
```

◆ The Lagrangian must be Hermitian

- ♣ Calculation of the Feynman rules of the Lagrangian minus its Hermitian conjugate
 - $\mathcal{L} - \mathcal{L}^\dagger$

```
In[8]:= 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.

Check of the implementation: normalization

◆ Textbook expression

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

◆ FEYNRULES implementation

```
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;
```

◆ The kinetic and mass terms must be canonically normalized

- ❖ Normalization of the quadratic terms (kinetic and mass terms)
- ❖ Absence of non-diagonal quadratic terms

```
In[9]:= CheckKineticTermNormalisation[LVector1];
        Neglecting all terms with more than 2 particles.
        All kinetic terms are diagonal.
        All kinetic terms are correctly normalized.
```

```
In[11]:= CheckDiagonalMassTerms[LVector1];
        All mass terms are diagonal.
```

- ❖ Other methods: *CheckDiagonalQuadraticTerms*, *CheckDiagonalKineticTerms*

Check of the implementation: the mass spectrum

◆ FEYNRULES gluino implementation

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

◆ FEYNRULES Lagrangian implementation

```
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;
```

◆ Checks that the mass information is consistent (numerically)

- ♣ Masses can be extracted from the Lagrangian
- ♣ Masses are fixed when particles are declared

```
In[12]:= CheckMassSpectrum[LVector1]
```

```
Neglecting all terms with more than 2 particles.
```

```
All mass terms are diagonal.
```

```
Getting mass spectrum.
```

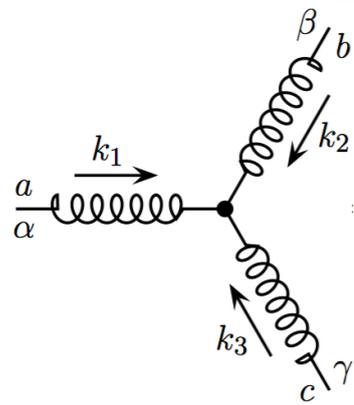
```
Checking for less than 0.1% agreement with model file values.
```

```
Out[12]//TableForm=
```

Particle	Analytic value	Numerical value	Model-file value
go	Mgo	500.	500.

The triple gluon vertex

◆ Literature



$$g_s f_{abc} \left[(k_1 - k_2)^\gamma \eta^{\alpha\beta} + (k_3 - k_1)^\beta \eta^{\gamma\alpha} + (k_2 - k_3)^\alpha \eta^{\beta\gamma} \right]$$

[Incoming momenta]

◆ FEYNRULES

Vertex 1

Particle 1 : Vector , G

Particle 2 : Vector , G

Particle 3 : Vector , G

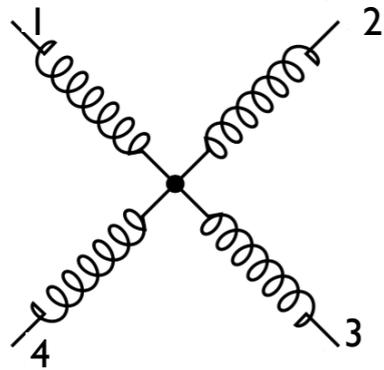
Vertex:

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

- ❖ Color: the index a_i is related to the i^{th} particle
(a is the index style for the adjoint $SU(3)_C$ indices)
- ❖ Spin: the index μ_i is the Lorentz index of the i^{th} (vector) particle

The quartic gluon vertex

◆ Literature



$$\begin{aligned}
 & ig_s^2 f^{a_1 a_2 b} f^{b a_3 a_4} (\eta^{\mu_1 \mu_4} \eta^{\mu_2 \mu_3} - \eta^{\mu_1 \mu_3} \eta^{\mu_2 \mu_4}) \\
 & + ig_s^2 f^{a_1 a_3 b} f^{b a_2 a_4} (\eta^{\mu_1 \mu_4} \eta^{\mu_2 \mu_3} - \eta^{\mu_1 \mu_2} \eta^{\mu_3 \mu_4}) \\
 & + ig_s^2 f^{a_1 a_4 b} f^{b a_2 a_3} (\eta^{\mu_1 \mu_3} \eta^{\mu_2 \mu_4} - \eta^{\mu_1 \mu_2} \eta^{\mu_3 \mu_4})
 \end{aligned}$$

◆ FEYNRULES

Vertex 2

Particle 1 : Vector , G

Particle 2 : Vector , G

Particle 3 : Vector , G

Particle 4 : Vector , G

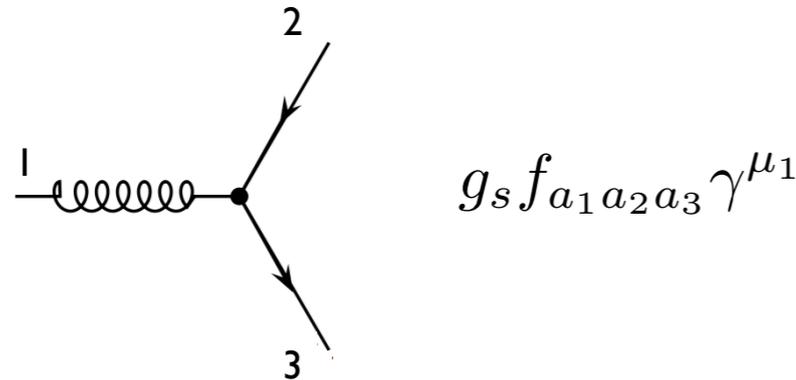
Vertex:

$$\begin{aligned}
 & i g_s^2 (f_{a_1, a_3, \text{Gluon}\$1} f_{a_2, a_4, \text{Gluon}\$1} \eta_{\mu_1, \mu_4} \eta_{\mu_2, \mu_3} + f_{a_1, a_2, \text{Gluon}\$1} f_{a_3, a_4, \text{Gluon}\$1} \eta_{\mu_1, \mu_4} \eta_{\mu_2, \mu_3} + \\
 & f_{a_1, a_4, \text{Gluon}\$1} f_{a_2, a_3, \text{Gluon}\$1} \eta_{\mu_1, \mu_3} \eta_{\mu_2, \mu_4} - f_{a_1, a_2, \text{Gluon}\$1} f_{a_3, a_4, \text{Gluon}\$1} \eta_{\mu_1, \mu_3} \eta_{\mu_2, \mu_4} - \\
 & f_{a_1, a_4, \text{Gluon}\$1} f_{a_2, a_3, \text{Gluon}\$1} \eta_{\mu_1, \mu_2} \eta_{\mu_3, \mu_4} - f_{a_1, a_3, \text{Gluon}\$1} f_{a_2, a_4, \text{Gluon}\$1} \eta_{\mu_1, \mu_2} \eta_{\mu_3, \mu_4})
 \end{aligned}$$

- ❖ Color: a_i is related to the i^{th} particle; the index *Gluon* $\$l$ is summed over
- ❖ Spin: μ_i is related to the i^{th} (vector) particle

Gluon and gluino interactions

◆ Literature



◆ FEYNRULES

Vertex 3

Particle 1 : Majorana , go

Particle 2 : Majorana , go

Particle 3 : Vector , G

Vertex:

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

- ♣ Color: a_i is related to the i^{th} particle
- ♣ Spin: μ_i is related to the i^{th} (vector) particle; s_i is related to the i^{th} fermion

The FeynmanRules function

◆ The function `FeynmanRules` has many options

- ❖ Restriction on the interactions to display
 - `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`

The matter Lagrangian

The SUSYQCD matter sector

◆ 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 coupling to 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 model dynamics is embedded in the Lagrangian

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

❖ **Mixed use of the mass / gauge bases** (makes things easier to implement)

❖ Kinetic terms (first three terms) and D-terms (second line) in the gauge basis

❖ Mass terms (end of first line) in the mass basis

❖ SUSY gauge quark-squark-gluino interactions (fourth line) in the gauge basis

Kinetic and gauge interactions

◆ Kinetic and gauge interactions, as well as mass terms

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

◆ The implementation in FEYNRULES follows the textbook expression

- ♣ Existence of shortcut functions (DC)
- ♣ Very compact implementation
- ♣ Repeated indices are summed

```
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];
```

Checking the Feynman rules

◆ We can compute the Feynman rules

```
In[18]:= 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[18]/MatrixForm=
```

$$\left(\begin{array}{l} \{G, 1\}, \{sq1, 2\}, \{sq1^\dagger, 3\} \\ \{G, 1\}, \{sq2, 2\}, \{sq2^\dagger, 3\} \\ \{G, 1\}, \{G, 2\}, \{sq1, 3\}, \{sq1^\dagger, 4\} \\ \{G, 1\}, \{G, 2\}, \{sq2, 3\}, \{sq2^\dagger, 4\} \\ \{\bar{q}, 1\}, \{q, 2\}, \{G, 3\} \end{array} \right) \begin{array}{l} i g s (p_2^{\mu_1} - p_3^{\mu_1}) T_{m_3, m_2}^{a_1} \\ i g s (p_2^{\mu_1} - p_3^{\mu_1}) T_{m_3, m_2}^{a_1} \\ i g s^2 \eta_{\mu_1, \mu_2} (T_{m_4, Colour\$1}^{a_1} T_{Colour\$1, m_3}^{a_2} + T_{Colour\$1, m_3}^{a_1} T_{m_4, Colour\$1}^{a_2}) \\ i g s^2 \eta_{\mu_1, \mu_2} (T_{m_4, Colour\$1}^{a_1} T_{Colour\$1, m_3}^{a_2} + T_{Colour\$1, m_3}^{a_1} T_{m_4, Colour\$1}^{a_2}) \\ i g s \gamma_{s_1, s_2}^{\mu_3} T_{m_1, m_2}^{a_3} \end{array}$$

- ❖ Field rotations have been performed automatically
 - ★ No more gauge eigenstates (sqL, sqR)
- ❖ All the QCD interactions of the squarks and quarks are derived
 - ★ The 3-point qqg interaction
 - ★ Two 3-point squark-gluon interactions (one for each squark species)
 - ★ Two 4-point squark-gluon interactions (one for each squark species)

D-terms

◆ D-terms (in the gauge eigenbasis)

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

◆ The FEYNRULES implementation (almost) follows the textbook expression

```
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**

Feynman rules from the D-terms

◆ D-terms (in the gauge eigenbasis)

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

◆ Feynman rules computation

```
Simplify[FeynmanRules[LD]] // MatrixForm
```

$\{\{sq1, 1\}, \{sq1, 2\}, \{sq1^\dagger, 3\}, \{sq1^\dagger, 4\}\}$	$-i g_s^2 \text{Cos}[2 \text{theta}]^2 \left(T_{m_3, m_2}^{\text{Gluon} \$1} T_{m_4, m_1}^{\text{Gluon} \$1} + T_{m_3, m_1}^{\text{Gluon} \$1} T_{m_4, m_2}^{\text{Gluon} \$1} \right)$
$\{\{sq1, 1\}, \{sq1^\dagger, 2\}, \{sq1^\dagger, 3\}, \{sq2, 4\}\}$	$\frac{1}{2} i g_s^2 \text{Sin}[4 \text{theta}] \left(T_{m_2, m_4}^{\text{Gluon} \$1} T_{m_3, m_1}^{\text{Gluon} \$1} + T_{m_2, m_1}^{\text{Gluon} \$1} T_{m_3, m_4}^{\text{Gluon} \$1} \right)$
$\{\{sq1^\dagger, 1\}, \{sq1^\dagger, 2\}, \{sq2, 3\}, \{sq2, 4\}\}$	$-i g_s^2 \text{Sin}[2 \text{theta}]^2 \left(T_{m_1, m_4}^{\text{Gluon} \$1} T_{m_2, m_3}^{\text{Gluon} \$1} + T_{m_1, m_3}^{\text{Gluon} \$1} T_{m_2, m_4}^{\text{Gluon} \$1} \right)$
$\{\{sq1, 1\}, \{sq1, 2\}, \{sq1^\dagger, 3\}, \{sq2^\dagger, 4\}\}$	$\frac{1}{2} i g_s^2 \text{Sin}[4 \text{theta}] \left(T_{m_3, m_2}^{\text{Gluon} \$1} T_{m_4, m_1}^{\text{Gluon} \$1} + T_{m_3, m_1}^{\text{Gluon} \$1} T_{m_4, m_2}^{\text{Gluon} \$1} \right)$
$\{\{sq1, 1\}, \{sq1^\dagger, 2\}, \{sq2, 3\}, \{sq2^\dagger, 4\}\}$	$-i g_s^2 \left(\text{Sin}[2 \text{theta}]^2 T_{m_2, m_3}^{\text{Gluon} \$1} T_{m_4, m_1}^{\text{Gluon} \$1} - \text{Cos}[2 \text{theta}]^2 T_{m_2, m_1}^{\text{Gluon} \$1} T_{m_4, m_3}^{\text{Gluon} \$1} \right)$
$\{\{sq1^\dagger, 1\}, \{sq2, 2\}, \{sq2, 3\}, \{sq2^\dagger, 4\}\}$	$-\frac{1}{2} i g_s^2 \text{Sin}[4 \text{theta}] \left(T_{m_1, m_3}^{\text{Gluon} \$1} T_{m_4, m_2}^{\text{Gluon} \$1} + T_{m_1, m_2}^{\text{Gluon} \$1} T_{m_4, m_3}^{\text{Gluon} \$1} \right)$
$\{\{sq1, 1\}, \{sq1, 2\}, \{sq2^\dagger, 3\}, \{sq2^\dagger, 4\}\}$	$-i g_s^2 \text{Sin}[2 \text{theta}]^2 \left(T_{m_3, m_2}^{\text{Gluon} \$1} T_{m_4, m_1}^{\text{Gluon} \$1} + T_{m_3, m_1}^{\text{Gluon} \$1} T_{m_4, m_2}^{\text{Gluon} \$1} \right)$
$\{\{sq1, 1\}, \{sq2, 2\}, \{sq2^\dagger, 3\}, \{sq2^\dagger, 4\}\}$	$-\frac{1}{2} i g_s^2 \text{Sin}[4 \text{theta}] \left(T_{m_3, m_2}^{\text{Gluon} \$1} T_{m_4, m_1}^{\text{Gluon} \$1} + T_{m_3, m_1}^{\text{Gluon} \$1} T_{m_4, m_2}^{\text{Gluon} \$1} \right)$
$\{\{sq2, 1\}, \{sq2, 2\}, \{sq2^\dagger, 3\}, \{sq2^\dagger, 4\}\}$	$-i g_s^2 \text{Cos}[2 \text{theta}]^2 \left(T_{m_3, m_2}^{\text{Gluon} \$1} T_{m_4, m_1}^{\text{Gluon} \$1} + T_{m_3, m_1}^{\text{Gluon} \$1} T_{m_4, m_2}^{\text{Gluon} \$1} \right)$

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

SUSY-gauge squark-quark-gluino couplings

◆ The SUSY gauge gluino-quark-squark interactions (in the gauge basis)

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

◆ The FEYNRULES is again following the textbook expression

```
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 be explicit** (scalars cannot be included in a fermion chain)
- ❖ *ProjM* is the left-handed chirality projector (*ProjP* is the right-handed one)
- ❖ The dot is used to connect the different elements of a fermion chain
 - ★ Rk: ProjM(s1,s2) is a scalar object

Feynman rules from the D-terms

◆ The SUSY gauge gluino-quark-squark interactions (in the gauge basis)

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

◆ Feynman rules computation

♣ Including the Hermitian conjugate pieces with the HC method

```
In[19]:= 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[19]//MatrixForm=
```

$$\left(\begin{array}{l} \left\{ \left\{ \bar{q}, 1 \right\}, \left\{ g_0, 2 \right\}, \left\{ sq1, 3 \right\} \right\} \quad -i \sqrt{2} g_s \left(\text{Cos}[\text{theta}] P_{+s_1, s_2} - P_{-s_1, s_2} \text{Sin}[\text{theta}] \right) T_{m_1, m_3}^{a_2} \\ \left\{ \left\{ g_0, 1 \right\}, \left\{ q, 2 \right\}, \left\{ sq1^\dagger, 3 \right\} \right\} \quad -i \sqrt{2} g_s \left(\text{Cos}[\text{theta}] P_{-s_1, s_2} - P_{+s_1, s_2} \text{Sin}[\text{theta}] \right) T_{m_3, m_2}^{a_1} \\ \left\{ \left\{ \bar{q}, 1 \right\}, \left\{ g_0, 2 \right\}, \left\{ sq2, 3 \right\} \right\} \quad i \sqrt{2} g_s \left(\text{Cos}[\text{theta}] P_{-s_1, s_2} + P_{+s_1, s_2} \text{Sin}[\text{theta}] \right) T_{m_1, m_3}^{a_2} \\ \left\{ \left\{ g_0, 1 \right\}, \left\{ q, 2 \right\}, \left\{ sq2^\dagger, 3 \right\} \right\} \quad i \sqrt{2} g_s \left(\text{Cos}[\text{theta}] P_{+s_1, s_2} + P_{-s_1, s_2} \text{Sin}[\text{theta}] \right) T_{m_3, m_2}^{a_1} \end{array} \right)$$

To phenomenology

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 MC tool(s)

- ♣ CALCHEP / COMPHEP
- ♣ FEYNARTS / FORMCALC
- ♣ UFO ➤ MADGRAPH5_AMC@NLO / SHERPA / HERWIG++ / WHIZARD

```
WriteCHOutput[{LVector1, LMatter}];  
WriteFeynArtsOutput[{LVector1, LMatter}];  
WriteUFO[{LVector1, LMatter}];
```

Limitations and fine prints (I)

◆ 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

Limitations and fine prints

◆ 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

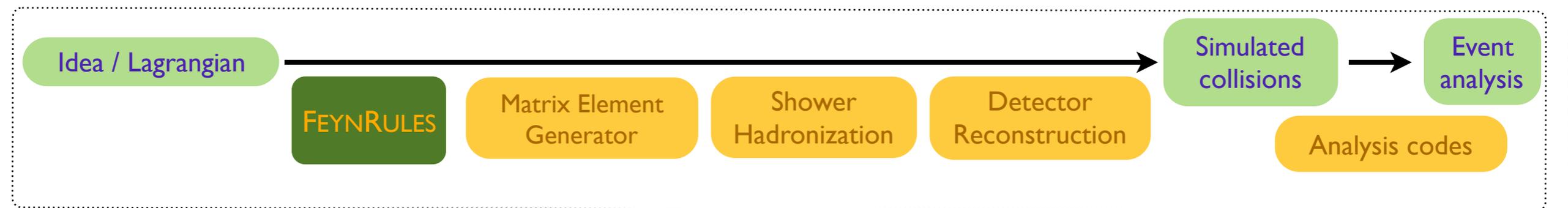
◆ Lorentz structures and spins: not supported in full generality

- ❖ 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: all Lorentz structures; spin 0, 1/2, 1
 - ★ WHIZARD: MSSM-like Lorentz structures; spin 0, 1/2, 1, 2

The UFO

New physics simulations in details

◆ A comprehensive approach to Monte Carlo simulations



◆ Implementation of any new physics theory in a MC tool is straightforward

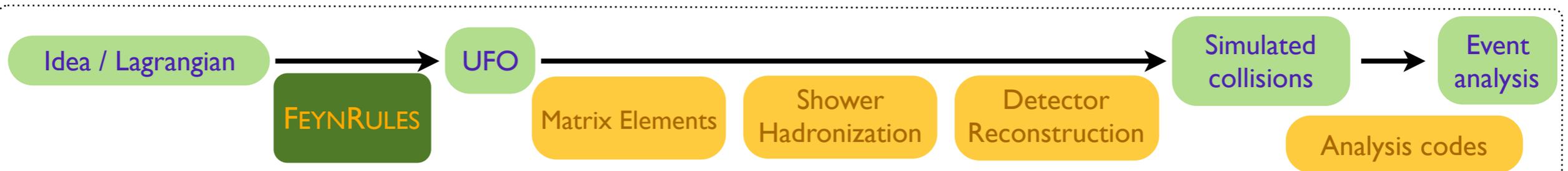
Many interfaces dedicated to specific tools

- ★ Removal of non compliant vertices
- ★ Translation to a specific format/language

! **Not efficient**

New physics simulations in details

◆ A comprehensive approach to Monte Carlo simulations



One format to rule them all

- ★ Easier to maintain
- ★ The MC generator decides what is needed

A step further: the Universal FEYNRULES Output

◆ The UFO in a nutshell

[Degrande, Duhr, BF, Grellscheid, Mattelaer, Reiter (CPC '12)]
[Degrande, Duhr, BF, Hirschi, Mattelaer, Shao et al. (in prep.)]

- ❖ UFO \equiv Universal FEYNRULES output
 - ★ **Universal** as not tied to any specific Monte Carlo program
- ❖ Consists of a **PYTHON module** to be linked to any code
- ❖ This module contains **all the model information**
 - ★ Allows the models to contain **generic color and Lorentz** structures
- ❖ Can be employed for next-to-leading order calculations

◆ The UFO is now a standard and used by many other programs

ALOHA

GOSAM

HERWIG ++

MADANALYSIS 5

SHERPA

MADGRAPH5_aMC@NLO

WHIZARD

LANHEP

SARAH

The UFO in practice

◆ The UFO is a set of PYTHON files

- ❖ Factorization of the information: particles, interactions, propagation, parameters, NLO, etc.

◆ Example

```
[fuchs@Benjamins-MacBook-Pro-3 ~/Work/tools/FeynRules/trunk/models/SUSYQCD_UFO$] ls
CT_couplings.py      SUSYQCD_UFO.log      couplings.py          object_library.py     propagators.py
CT_parameters.py     __init__.py          function_library.py  parameters.py          vertices.py
CT_vertices.py       coupling_orders.py   lorentz.py           particles.py           write_param_card.py
[fuchs@Benjamins-MacBook-Pro-3 ~/Work/tools/FeynRules/trunk/models/SUSYQCD_UFO$]
```

NLO

Interactions

Particles

Parameters

Propagators

Particles

◆ Particles are stored in the `particles.py` file

- ❖ Instances of the particle class
- ❖ Attributes: particle spin, color representation, mass, width, PDG code, etc.
- ❖ Antiparticles automatically derived

```
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()
```

Parameters

◆ Parameters are stored in the `parameters.py` file

- ❖ Instances of the parameter class
- ❖ External parameters are organized following a Les Houches-like structure (blocks and counters)
- ❖ PYTHON-compliant formula for the internal parameters

```
aS = Parameter(name = 'aS',
               nature = 'external',
               type = 'real',
               value = 0.1184,
               texname = '\\alpha_s',
               lhablock = 'SMINPUTS',
               lhacode = [ 3 ])

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

```
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 ])
```

Interactions: generalities

◆ Vertices decomposed in a **spin x color** basis (coupling strengths \equiv coordinates)

♣ Example: the quartic gluon vertex can be written as

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

- ★ 3 elements for the color basis
- ★ 3 elements for the spin (Lorentz structure) basis
- ★ 9 coordinates (6 are zero)

♣ Several files are used for the storage of the information

Example: the quartic gluon vertex

◆ General information in vertex.py

```
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** \equiv spin basis
(in lorentz.py; common to all vertices)
- ★ **color** \equiv color basis
- ★ **couplings** \equiv coordinates
(in couplings.py; common to all vertices)

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

◆ Lorentz structures: straightforward implementations in lorentz.py

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

◆ Couplings: straightforward implementations in couplings.py

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

Coupling orders: for selecting diagrams



Outline

1. FEYNRULES in a nutshell
2. Implementing supersymmetric QCD in FEYNRULES
3. Using FEYNRULES with supersymmetric QCD model
- 4. Advanced model implementation techniques**
5. Summary

Advanced techniques for FEYNRULES implementation

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

Extending models

Merging and extending models (I)

◆ 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

◆ Simplified models

♣ Special cases very relevant for LHC physics: Simplified Model Spectra

- ★ The Standard Model + one or two new particles
- ★ Often inspired by the MSSM or dark matter models
- ★ Ex.: the SM + lightest stop and neutralino + relevant subset of MSSM interactions

Merging and extending models (2)

◆ 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 the models available on the FEYNRULES database

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

The FEYNRULES model database

◆ $O(100)$ 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.*

Restricting model implementations

- ◆ Many BSM models of interest are subset of other models
 - ❖ Equivalent to the parent model, with some parameters set to 0 or 1
 - ★ Example 1: the massless version of a model (massless light quarks in the SM)
 - ★ Example 2: a mixing matrix set to the identity (no-CKM matrix in the SM)
 - ❖ 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 thus Monte Carlo model files)
 - ❖ Are free from the restricted parameters
 - ❖ Smaller files, more efficiency at the MC level
 - ★ Ex.: the MSSM has more than 10000 vertices; its flavor-conserving version ~1000

Example of restrictions

◆ 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"]
```

Superspace module

See the manual for more details on the superspace module

Superfield declaration

- ◆ A module dedicated to calculations in superspace
 - ♣ Superfield declaration and links to the component fields

A left-handed squark superfield

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

- ★ Component fields to be declared too
- ★ Vector superfields can be linked to gauge groups

Supersymmetric model implementation

◆ Supersymmetric model implementation

- ❖ Declaration of the model **gauge group**
- ❖ Declaration of all **fields and superfields**
- ❖ Declaration of all model **parameters**
- ❖ **Writing the Lagrangian (in a simplified way)**

◆ Supersymmetric Lagrangian in superspace are very compact

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

- ❖ First line: kinetic and gauge interaction terms for all fields
 - **Model independent** ⇒ can be automated
- ❖ Second line: superpotential and supersymmetry breaking Lagrangian
 - **Model dependent** ⇒ to be provided
- ❖ Series expansion in terms of component fields
- ❖ Automatic **derivation of supersymmetric Lagrangians**
- ❖ Solving the equations of motion of the unphysical fields

Implementing supersymmetric Lagrangians

◆ Supersymmetric Lagrangian in superspace are very compact

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

- ❖ First line: kinetic and gauge interaction terms for all fields
 - ★ Model independent \Rightarrow can be automated (**CSFKineticTerms** / **VSFKineticTerms**)
 - ★ Dedicated methods to access the components of a superfield (**Theta2Component**, etc.)
- ❖ Second line: superpotential and supersymmetry-breaking terms
 - ★ Model dependent \Rightarrow to be provided by the user (*SuperW* and *LSoft*)

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

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

Implementing supersymmetric Lagrangians

◆ The implemented Lagrangian above must be post-processed

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

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

- ♣ Solving the **equation of motion** for the auxiliary fields
- ♣ 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**)

Mass matrix diagonalization

See the manual for more details on the ASPERGE 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 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 module allowing one to extract the mass matrices from the Lagrangian
- ❖ A generator of C++ code ➤ numerical diagonalization of all mass matrices (the generated code can be used in a standalone way)
- ❖ See: Alloul, D'Hondt, De Causmaecker, BF, Rausch de Traubenberg [EPJC 73 (2013) 2325]

Example of the Z-boson/photon mixing

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

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

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

```
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];
```

Decays

See the manual for more details on the decay module

Two-body decays

◆ The problematics of the decay widths and branching ratios

- ❖ Some MC tools need decay tables (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

Two-body decays

◆ Two-body decays in general

- ✿ Two-body decays can be directly read **from three-point vertices** (\mathcal{V})

$$\Gamma_{1 \rightarrow 2} = \frac{1}{2|M|S} \int d\Phi_N |\mathcal{M}_{1 \rightarrow 2}|^2 = \frac{\sqrt{\lambda(M^2, m_1^2, m_2^2)}}{16 \pi S |M|^3} \mathcal{V}_{l_1 l_2 l_3}^{a_1 a_2 a_3} \mathcal{P}_1^{l_1 l'_1} \mathcal{P}_2^{l_2 l'_2} \mathcal{P}_3^{l_3 l'_3} (\mathcal{V}^*)_{l'_1 l'_2 l'_3}^{a_1 a_2 a_3}$$

- ★ 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

◆ The decay module of FEYNRULES

- ✿ 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** (used, e.g. by MADWIDTH)

Running 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, BF, Mattelaer, Öztürk, Shen [CPC (2015)]

Snippet of the UFO output

◆ The information is (by default) employed by the UFO interface

- ❖ Can be turned off: (**AddDecays** → **False**)
- ❖ The UFO contains an extra file *decays.py*
- ❖ This file can be used by MC codes
- ❖ 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)')})
```



NLO

Higher-order corrections (in QCD)

◆ NLO calculations matched to parton shower (for BSM) are automated

♣ Model-dependent parts of calculations (on top of the tree-level information)

★ Counterterms

★ Finite pieces of the loop-integrals

UFO @ NLO

[Degrande, Duhr, BF, Hirschi, Mattelaer, Shao et al. (in prep.)]

♣ Model independent contributions

★ Subtraction of the divergences

★ Matching to the parton showers

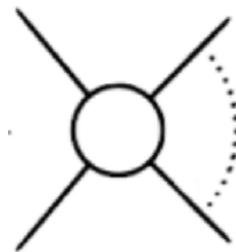
Recap' on NLO calculations

◆ Contributions to an NLO result in QCD

♣ Three ingredients: the Born, virtual loop and real emission contributions

$$\sigma_{NLO} = \int d^4\Phi_n \mathcal{B} + \int d^4\Phi_n \int_{\text{loop}} d^d\ell \mathcal{V} + \int d^4\Phi_{n+1} \mathcal{R}$$

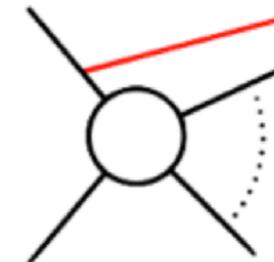
Born



Virtuals: one extra power of α_s and divergent



Reals: one extra power of α_s and divergent



Extra information is needed

Virtual contributions

◆ Loop diagram calculations

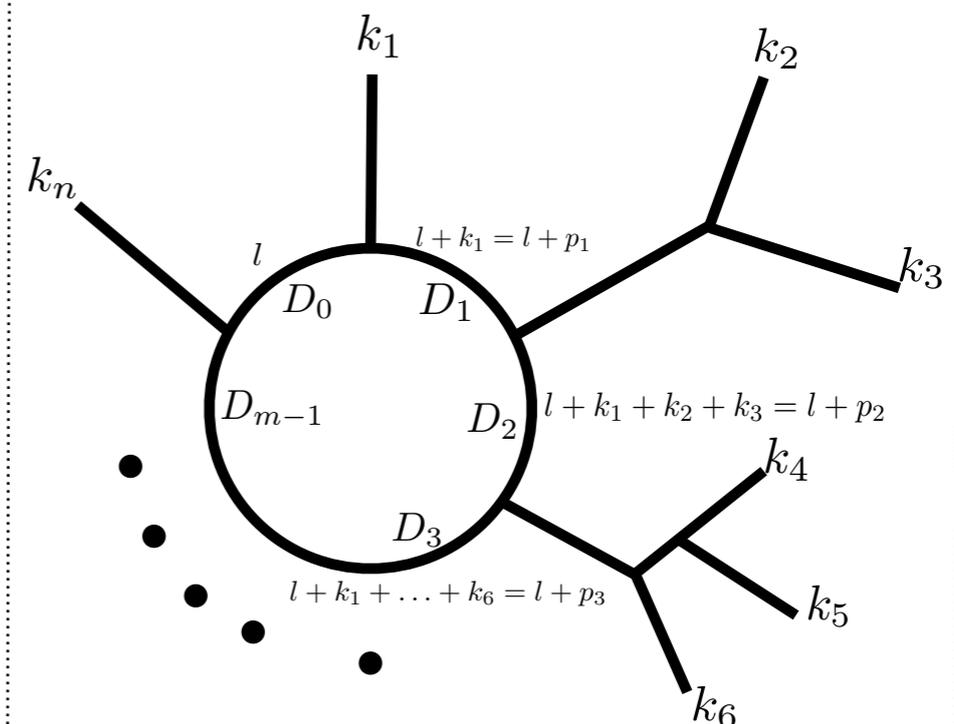
- ❖ Calculations to be done in $d=4-2\epsilon$ dimensions
 - ★ Divergences made explicit ($1/\epsilon^2, 1/\epsilon$)

- ❖ Rewriting loop integrals with **scalar integrals**

$$\int d^d \ell \frac{N(\ell)}{D_0 D_1 \cdots D_{m-1}} = \sum a_i \int d^d \ell \frac{1}{D_{i_0} D_{i_1} \cdots}$$

- ★ Involves integrals with **up to four denominators**
 - The decomposition basis is finite
 - **Can be computed once and for all**
- ★ The reduction is the process-dependent part

m -point diagram with n external momenta



The rational terms (R_1 and R_2)

◆ The loop momentum lives in a d -dimensional space

- ♣ Reduction to be done in d dimensions

$$\int d^d \ell \frac{N(\ell, \tilde{\ell})}{\bar{D}_0 \bar{D}_1 \cdots \bar{D}_{m-1}} \quad \text{with } \bar{\ell} = \ell + \tilde{\ell}$$

D-dim
4-dim
(-2ε)-dim

- ♣ Numerical methods works in 4 dimensions: need to be compensated!

◆ The R_1 terms originates from the denominators

- ♣ Connected to the internal propagators

◆ The R_2 terms originates from the numerator

- ♣ Can be seen as extra diagrams with special Feynman rules

R₁ terms

◆ The R₁ terms originates from the denominators

$$\frac{1}{\bar{D}} = \frac{1}{D} \left(1 - \frac{\tilde{\ell}^2}{\bar{D}} \right)$$

♣ These extra pieces can be calculated **generically** (3 integrals in total)

$$\int d^d \bar{\ell} \frac{\tilde{\ell}^2}{\bar{D}_i \bar{D}_j} = -\frac{i\pi^2}{2} \left[m_i^2 + m_j^2 - \frac{(p_i - p_j)^2}{2} \right] + \mathcal{O}(\varepsilon)$$

$$\int d^d \bar{\ell} \frac{\tilde{\ell}^2}{\bar{D}_i \bar{D}_j \bar{D}_k} = -\frac{i\pi^2}{2} + \mathcal{O}(\varepsilon)$$

$$\int d^d \bar{\ell} \frac{\tilde{\ell}^2}{\bar{D}_i \bar{D}_j \bar{D}_k \bar{D}_l} = -\frac{i\pi^2}{6} + \mathcal{O}(\varepsilon)$$

- ♣ The denominator structure is already known at the reduction time
- ♣ The R₁ coefficients are extracted during the reduction

R₂ terms

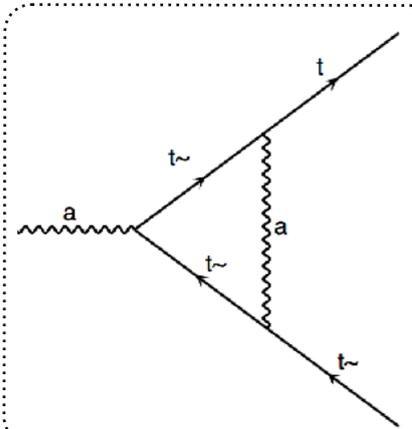
◆ The R₂ terms originates from the numerator

$$\bar{N}(\bar{\ell}) = \underset{\substack{\downarrow \\ \text{D-dim}}}{N}(\bar{\ell}) + \underset{\substack{\downarrow \\ \text{4-dim}}}{N}(\bar{\ell}) + \underset{\substack{\downarrow \\ \text{(-2}\varepsilon\text{)-dim}}}{\tilde{N}}(\tilde{\ell}, \ell, \varepsilon) \quad \Rightarrow \quad R_2 \equiv \lim_{\varepsilon \rightarrow 0} \frac{1}{(2\pi)^4} \int d^d \bar{\ell} \frac{\tilde{N}(\tilde{\ell}, \ell, \varepsilon)}{\bar{D}_0 \bar{D}_1 \cdots \bar{D}_{m-1}}$$

❖ Practically, we isolate the epsilon part

❖ There is only a finite set of loops for which it does not vanish

◆ They can be re-expressed in terms of R₂ Feynman rules



$$\propto \int d^d \bar{\ell} \frac{\tilde{\ell}^2}{\bar{D}_i \bar{D}_j \bar{D}_k} = -\frac{i\pi^2}{2} + \mathcal{O}(\varepsilon) \Rightarrow \text{Diagram} \quad -i \frac{\alpha}{2\pi} e\gamma^\mu$$

R₂ Feynman rules

◆ The R₂ are process dependent and model-dependent (like Feynman rules)

- ❖ In a renormalizable theory, there is a finite number of them
- ❖ They can be derived from the sole knowledge of the bare Lagrangian

[Ossala, Papadopoulos, Pittau (JHEP'08)]

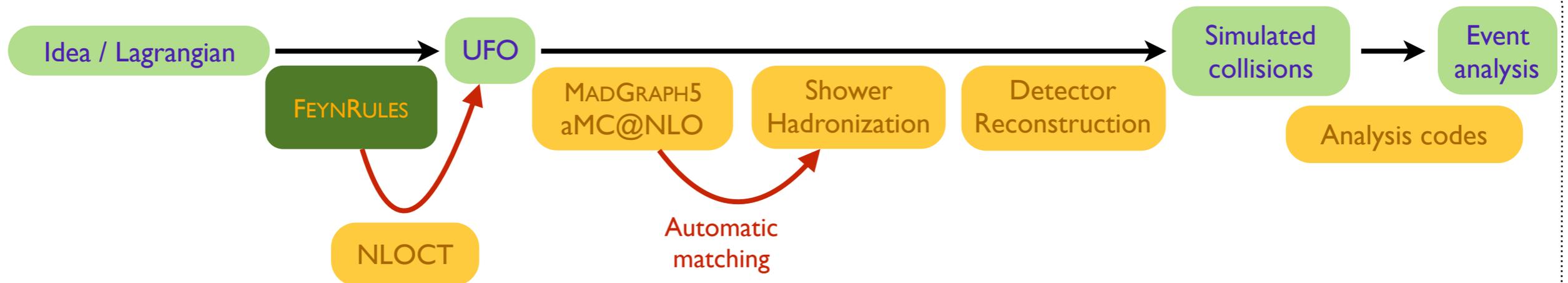
◆ The R₂ calculation can be automated and performed once and for all

- ❖ Development of the NLOCT package (extension of FEYNRULES)
- ❖ Computation, for any model, of all R₂ and UV counterterms
 - ★ In the on-shell and MSbar schemes
- ❖ Inclusion of the output in the UFO

[Degrande (CPC'15)]

Automated NLO simulations with MG5_AMC

◆ A comprehensive approach to Monte Carlo simulations at the NLO in QCD



Outline

1. FEYNRULES in a nutshell
2. Implementing supersymmetric QCD in FEYNRULES
3. Using FEYNRULES with supersymmetric QCD model
4. Advanced model implementation techniques
5. **Summary**

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!**