Computer tools in particle physics

- Lecture 5 : odds and ends -

Avelino Vicente
IFIC – CSIC / U. Valencia

Curso de doctorado de la U. València

IFIC
May 22-26 2017

Outline of the lecture

- Propaganda
  - FlavorKit
  - DsixTools
- Other popular tools
- Questions, comments, ...

Chuck Norris fact of the day

When Chuck Norris does a pushup, he isn't lifting himself up, he's pushing the Earth down

FlavorKit

W. Porod, F. Staub, A. Vicente

Manual: arXiv:1405.1434
Website: http://sarah.hepforge.org/FlavorKit.html

Flavor observables in a nutshell

Step 1: Consider a lagrangian that includes all the operators relevant for the flavor observable

$$\mathcal{L}_{eff} = \sum_i {\color{red} C_i} {\color{blue} \mathcal{O}_i}$$

Flavor observables in a nutshell

Step 1: Consider a lagrangian that includes all the operators relevant for the flavor observable

$$\mathcal{L}_{eff} = \sum_i C_i \mathcal{O}_i$$

Step 2: Compute the Wilson coefficients at a given loop order

Flavor observables in a nutshell

Step 1: Consider a lagrangian that includes all the operators relevant for the flavor observable

$$\mathcal{L}_{eff} = \sum_i C_i \mathcal{O}_i$$

Step 2: Compute the Wilson coefficients at a given loop order

Step 3: Plug the results for the Wilson coefficients into a general expression for the flavor observable

Example: $\mathrm{BR}(\mu \to e\gamma)$

[In the SM extended with Dirac neutrino masses]

Step 1: Consider a lagrangian that includes all the operators relevant for the flavor observable

$$\mathcal{L}_{\mu e\gamma} = i e \, m_\mu \, \bar{e} \, \sigma^{\mu\nu} q_\nu \left(K_2^L P_L + K_2^R P_R \right) \mu A_\mu + \mathrm{h.c.}$$

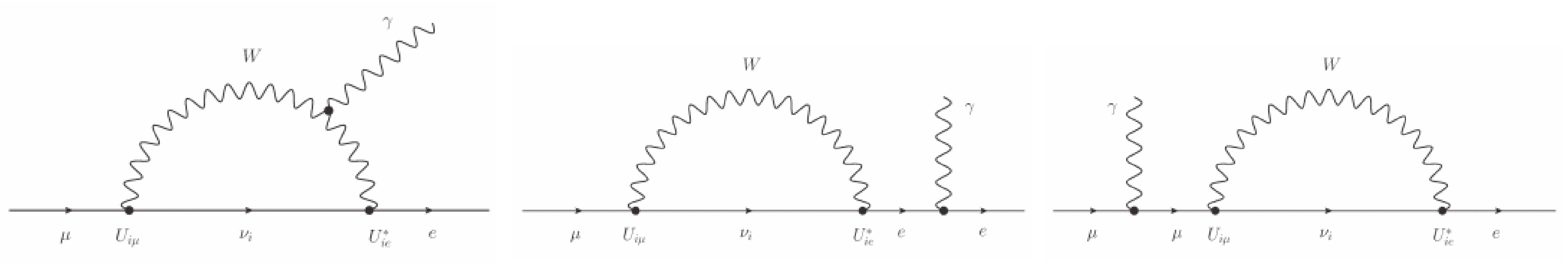
Dipole interaction lagrangian

K_2^L, K_2^R : Wilson coefficients

Example: $\mathrm{BR}(\mu \to e\gamma)$

[In the SM extended with Dirac neutrino masses]

Step 2: Compute the Wilson coefficients at a given loop order



$$K_2^L = \frac{G_F}{2\sqrt{2}\pi^2} m_\mu \sum_i U_{i\mu} U_{ie}^* (F_1 + F_2)$$

$$K_2^R = \frac{G_F}{2\sqrt{2}\pi^2} m_e \sum_i U_{i\mu} U_{ie}^* (F_1 - F_2)$$

[Ma, Pramudita, '81]

Example: $\mathrm{BR}(\mu \to e\gamma)$

[In the SM extended with Dirac neutrino masses]

Step 3: Plug the results for the Wilson coefficients into a general expression for the flavor observable

$$\mathrm{BR}\,(\mu \to e\gamma) = \frac{\alpha\, m_\mu^5}{4\,\Gamma_\mu} \left(|K_2^L|^2 + |K_2^R|^2\right)$$

Flavor observables in a nutshell

Step 1: Consider a lagrangian that includes all the operators relevant for the flavor observable

Some freedom. Requires a good understanding of the observable but technically easy

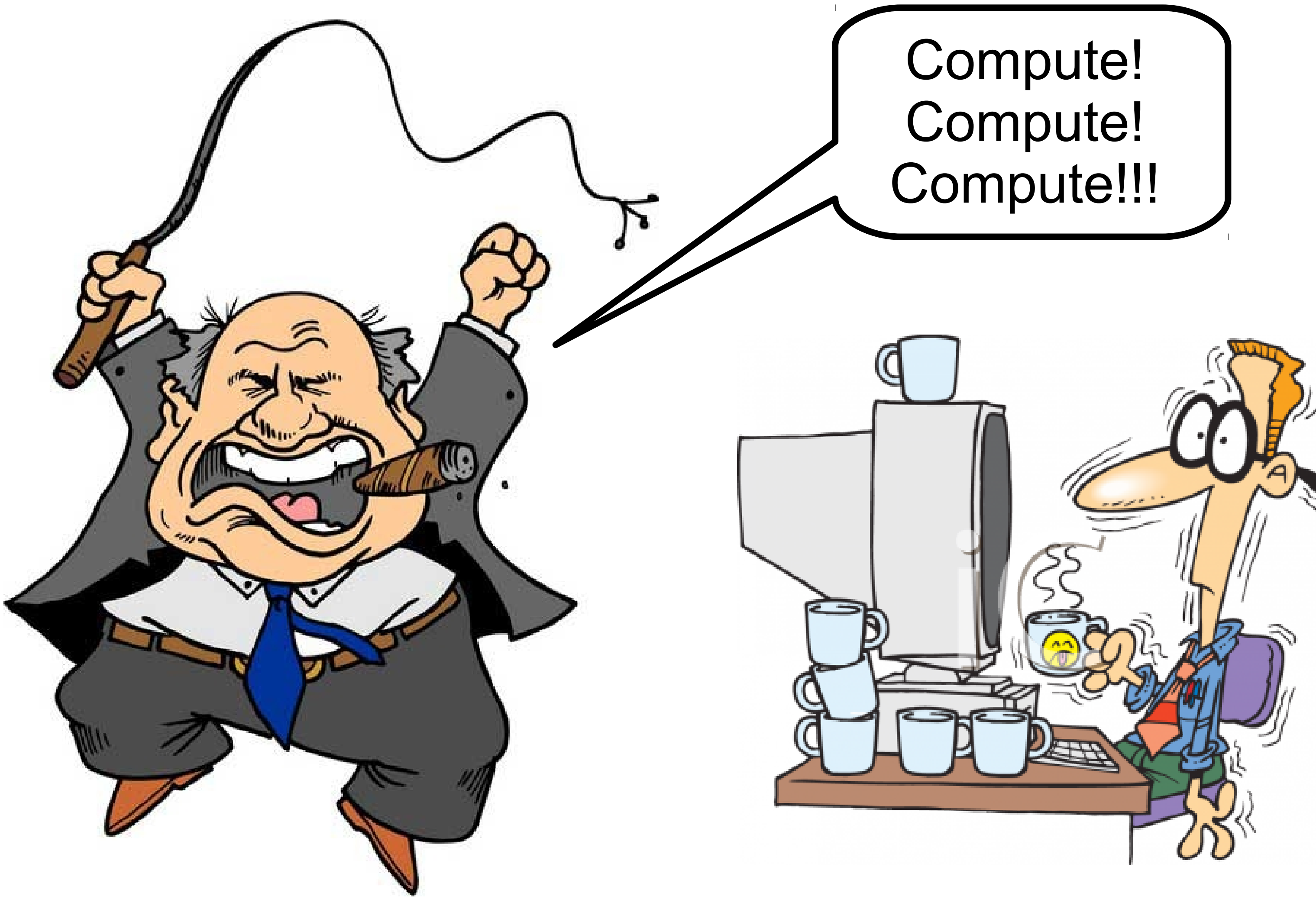
Step 2: Compute the Wilson coefficients at a given loop order

Complicated and model dependent part of the computation

Step 3: Plug the results for the Wilson coefficients into a general expression for the flavor observable

Model independent. Can make use of results in the literature

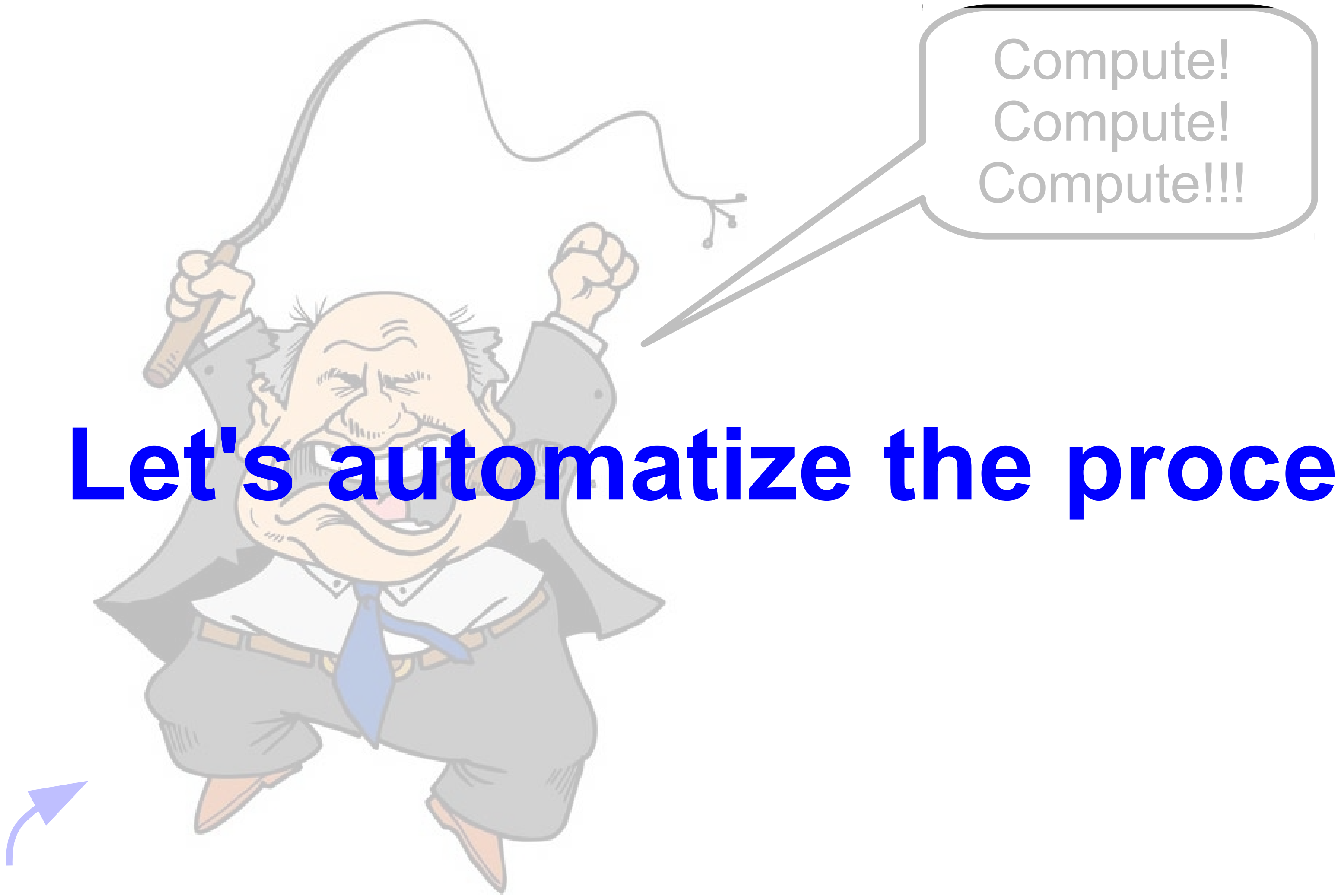
Usual approach



Professor

Poor student

Usual approach



Let's automatize the process!

Professor

Poor student

FlavorKit

To compute flavor observables one needs:

1) Expressions for all vertices and masses → SARAH

2) Expressions for the Wilson coefficients → FeynArts/FormCalc

3) Expressions for the observables → Literature

4) Numerical evaluation → SPheno

FlavorKit is the combination of these tools

How to use FlavorKit

- **Basic usage**

  For those who do not need any operator nor observable beyond what is already implemented in FlavorKit. In this case, FlavorKit reduces to the standard SARAH package.

Observables already in FlavorKit

Lepton flavor	Quark flavor
$\ell_\alpha \to \ell_\beta \gamma$	$B^0_{s,d} \to \ell^+ \ell^-$
$\ell_\alpha \to 3\,\ell_\beta$	$\bar{B} \to X_s \gamma$
$\mu - e$ conversion in nuclei	$\bar{B} \to X_s \ell^+ \ell^-$
$\tau \to P\,\ell$	$\bar{B} \to X_{d,s} \nu \bar{\nu}$
$h \to \ell_\alpha \ell_\beta$	$B \to K \ell^+ \ell^-$
$Z \to \ell_\alpha \ell_\beta$	$K \to \pi \nu \bar{\nu}$
	$\Delta M_{B_{s,d}}$
	ΔM_K and ε_K
	$P \to \ell \nu$

Ready to be computed in your favourite model!

How to use FlavorKit

Basic usage

For those who do not need any operator nor observable beyond what is already implemented in FlavorKit. In this case, FlavorKit reduces to the standard SARAH package.

Advanced usage

For those with further requirements:

- New observables
- New operators

2nd Chuck Norris fact of the day

Chuck Norris can run collider simulations with MadGraph on an abacus

DsixTools

A. Celis, J. Fuentes-Martín, A. Vicente, J. Virto

Manual: arXiv:1704.04504
Website: https://dsixtools.github.io/

The SMEFT

$$\boxed{\mathcal{L} = \mathcal{L}_{\text{SM}}^{(4)} + \frac{1}{\Lambda}\sum_k C_k^{(5)} Q_k^{(5)} + \frac{1}{\Lambda^2}\sum_k C_k^{(6)} Q_k^{(6)} + \mathcal{O}\left(\frac{1}{\Lambda^3}\right)}$$

Gauge invariant operators

Focus on dimension-6 operators

Warsaw basis

[Grzadkowski et al, 2010]

2499 real parameters (3045 with B-violation)

Full 1-loop RGEs computed

[Alonso, Chang, Jenkins, Manohar, Shotwell, Trott, 2013-2014]

The SMEFT

$$\mathcal{L} = \mathcal{L}_{\rm SM}^{(4)} + \frac{1}{\Lambda} \sum_k C_k^{(5)} Q_k^{(5)} + \frac{1}{\Lambda^2} \sum_k C_k^{(6)} Q_k^{(6)} + \mathcal{O}\left(\frac{1}{\Lambda^3}\right)$$

Gauge invariant operators

Focus on dimension-6 operators

Warsaw basis

[Grzadkowski et al, 2010]

2499 real parameters (3045 with B-violation)

Full 1-loop RGEs computed

[Alonso, Chang, Jenkins, Manohar, Shotwell, Trott, 2013-2014]

Non-trivial coupled system

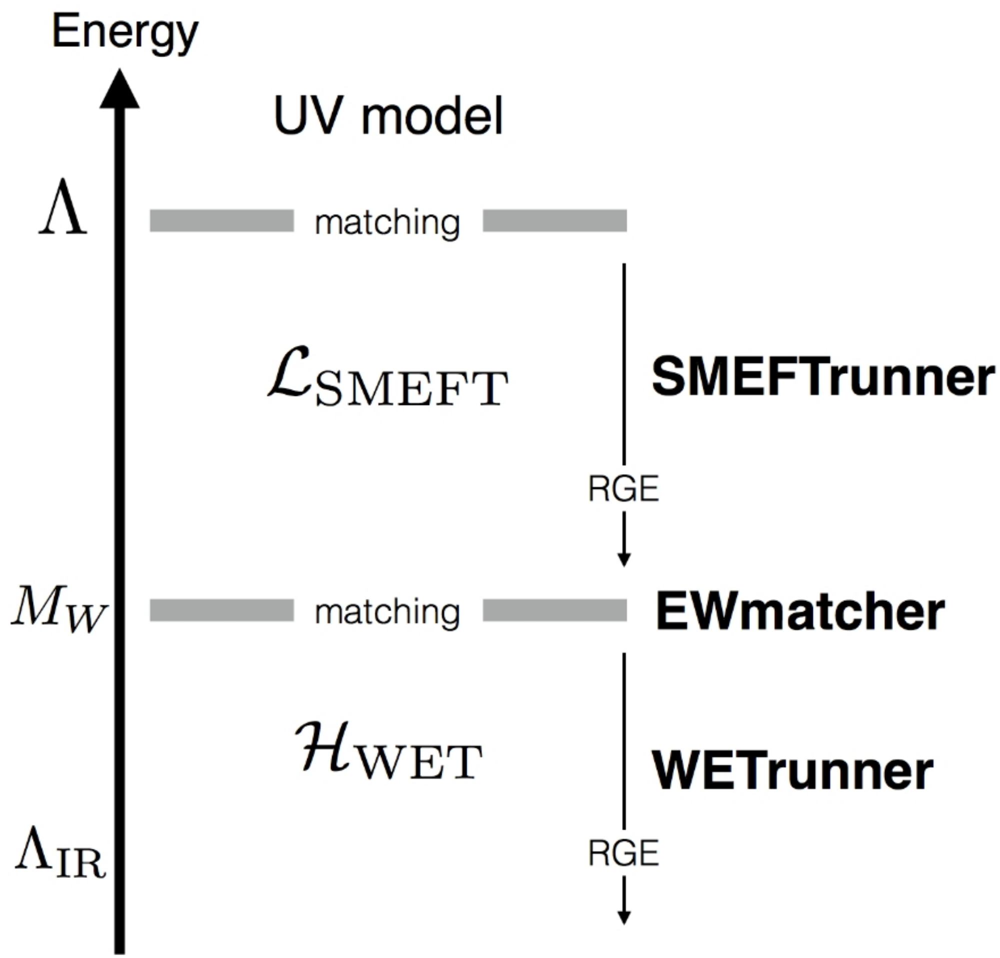
Computers are known to be good at complex games...

DsixTools

Mathematica package

Modular structure

Each module can be used independently



Stuff DsixTools can do for you

- Complete 1-loop SMEFT RGE running
- Easy analytical treatment
- Direct input on the notebook
- Easy loops with varying WCs and/or energy scales
- Input and Output with SLHA inspired text files
- Transformation to fermion mass basis at the EW scale
- EW matching to many WET operators for B-physics
- QED and QCD running from the EW scale down to the b-quark mass scale

And much more to come!

Other popular tools

FeynRules

A. Alloul, N. D. Christensen, C. Degrande, C. Duhr, B. Fuks

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

Mathematica package that allows to derive Feynman rules from a Lagrangian

- Similar in purpose to SARAH
- Input required from the user: particle content and Lagrangian
- Many models already implemented
- Interface to other popular codes (matrix element generators)

FeynCalc

R. Mertig, F. Orellana, V. Shtabovenko

Website: https://feyncalc.github.io/

Mathematica package for symbolic evaluation of Feynman diagrams and algebraic calculations

- “Abandoned” for many years but recently revived
- Lorentz index contraction, color factor calculation, Dirac matrix manipulation and traces, SU(N) algebra...
- Many tools to deal with loop integrals: PV reduction, tables of integrals...
- Generation of Feynman rules

Sym2Int

R. Fonseca

Website: http://renatofonseca.net/sym2int.php

Mathematica package that lists all valid interactions given the model symmetries and fields

- Uses the theory group of Susyno (http://renatofonseca.net/susyno.php)
- A large variety of symmetry groups and representations
- Not restricted to dim-4 interaction terms
- Can also be used to calculate gauge and Lorentz contractions

HEPfit

The HEPfit collaboration

Website: http://hepfit.roma1.infn.it/index.html

Code for the Combination of Indirect and Direct Constraints on HEP Models

- Global fits to Higgs, precision and flavor observables
- Bayesian statistics
- Predictions to new observables based on fit results
- New BSM models can be added

Concluding remarks

Concluding remarks

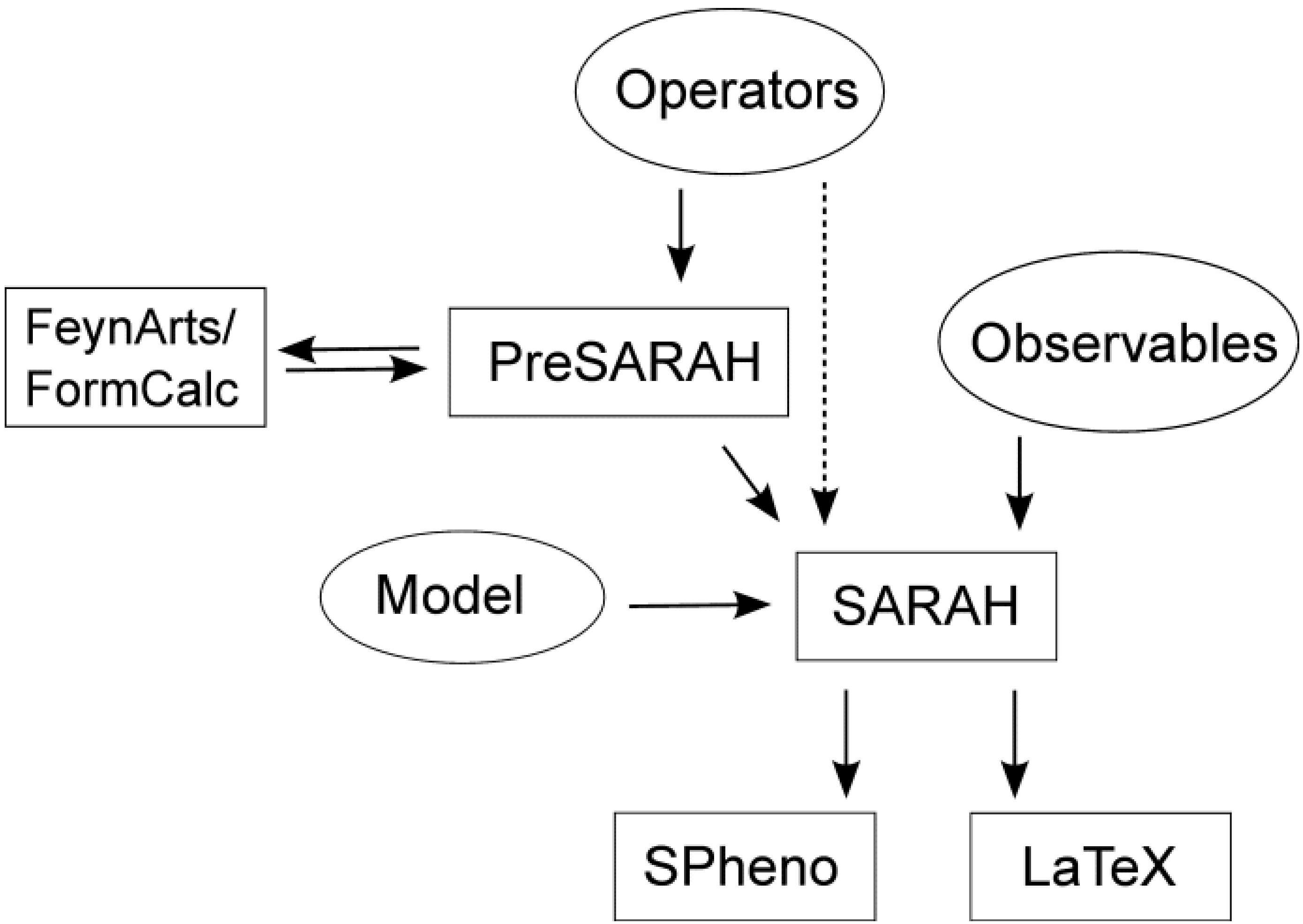
Many “routine tasks” can nowadays be performed with the help of (easy to use) computer tools

Keep in mind:

- Do not be afraid to use them
- Always understand what you are doing

Backup slides

FlavorKit



Operators
FeynArts/
FormCalc
PreSARAH
Observables
Model
SARAH
SPheno
LaTeX

New observables

Implementing a new observable

Two files: steering file “observable.m” + Fortran code “observable.f90”

```
NameProcess = "LLpGamma";
NameObservables = {{muEgamma, 701, "BR(mu->e gamma)"},
                   {tauEgamma, 702, "BR(tau->e gamma)"},
                   {tauMuGamma, 703, "BR(tau->mu gamma)"}};

NeededOperators = {K2L, K2R};

Body = "LLpGamma.f90";
```

Steering file
LLpGamma.m

Reminder:

$$\mathcal{L}_{\mu e\gamma} = ie\, m_\mu\, \bar{e}\, \sigma^{\mu\nu} q_\nu \left(K_2^L P_L + K_2^R P_R \right) \mu A_\mu + \text{h.c.}$$

New observables

```
Real(dp) :: width
Integer :: i1, gt1, gt2

Do i1=1,3
If (i1.eq.1) Then        ! mu -> e gamma
      gt1 = 2
      gt2 = 1
Elseif (i1.eq.2) Then
      ...
End if

width = 0.25_dp*mf_l(gt1)**5*(Abs(K2L(gt1,gt2))**2 + Abs(K2R(gt1,gt2))**2)*Alpha

If (i1.eq.1) Then
      muEgamma = width/(width+GammaMu)
Elseif (i1.eq.2) Then
      ...
End if
End do
```

Fortran code

LLpGamma.f90

New operators

Implementing a new operator

One file: PreSARAH input file “operator.m”

Generic expressions for the Wilson coefficients of new operators can be computed with the help of an additional package (PreSARAH):

- User friendly definition of new operators
- Uses FeynArts/FormCalc [by T. Hahn] to obtain the generic expressions
- Writes all necessary files for SARAH

Example:

$$\mathcal{L}_{2d2\ell} = \sum_{\substack{I=S,V,T \\ X,Y=L,R}} E^{I}_{XY} \, \bar{d}_\beta \Gamma_I P_X d_\alpha \, \bar{\ell}_\gamma \, \Gamma_I P_Y \ell_\gamma + \text{h.c.}$$

$$(\Gamma_{S,V,T} = 1, \gamma_\mu, \sigma_{\mu\nu})$$

New operators

```
NamePrócess="2d2L";

ConsideredProcess = "4Fermion";
FermionOrderExternal={2,1,4,3};
NeglectMasses={1,2,3,4};


ExternalFields= {DownQuark,bar[DownQuark],ChargedLepton,bar[ChargedLepton]};

CombinationGenerations = {{3,1,1,1}, {3,1,2,2}, {3,1,3,3},{3,2,1,1}, {3,2,2,2}, {3,2,3,3}};


AllOperators={{OddllSLL,Op[7].Op[7]},
              {OddllSRL,Op[6].Op[7]},
              ...,
              {OddllVRR,Op[7,Lor[1]].Op[7,Lor[1]]},
              ...,
              {OddllTLL,Op[-7,Lor[1],Lor[2]].Op[-7,Lor[1],Lor[2]]},
              ...};
```

PreSARAH input file
2d2L.m

Note:

$$\mathrm{Op}[7]\ ,\ \mathrm{Op}[6]\ = P_{L,R}$$

$$\mathrm{Lor}[1]\ = \gamma_\mu$$

Other flavor codes

- **MicrOmegas** [Belanger, Boudjema, Pukhov, Semenov]
- **NMSSM-Tools** [Ellwanger, Hugonie]
- **SPheno** [Porod, Staub]
- **SuperIso** [Mahmoudi]
- **SuSeFLAV** [Chowdhury, Garani, Vempati]
- **SUSY_FLAVOR** [Rosiek, Chankowski, Dedes, Jäger, Tanedo] [Crivellin, Rosiek]
- **...**

Restrictions: Only specific models + hard to extend

FlavorKit limitations

Disclaimer 

FlavorKit is a tool intended to be as general as possible. For this reason, there are some limitations compared to codes which perform specific calculations in a specific model:

- Chiral resummation is not included because of its large model dependence
- Higher order corrections cannot be computed (although they can be included in a parametric way)

A simple program: numerics

A DsixTools Program

This notebook loads DsixTools and shows how to use the SMEFTrunner module.

```
SetDirectory[NotebookDirectory[]];
```

Start DsixTools

```
Needs["DsixTools`"]
```

Read input files

```
ReadInputFiles["Options.dat", "WCsInput.dat", "SMInput.dat"];
```

Load SMEFTrunner module

```
LoadModule["SMEFTrunner"]
```

Use SMEFTrunner module

```
LoadBetaFunctions;

RunRGEsSMEFT;
```

SMEFT WCs input file

```
Block WC4
6 1.0          # phiBtilde
Block IMWCDPHI
1 1 0.1        # dphi(1,1)
1 2 0.2        # dphi(1,2)
1 3 0.3        # dphi(1,3)
2 1 0.1        # dphi(2,1)
2 2 0.2        # dphi(2,2)
2 3 0.3        # dphi(2,3)
3 1 0.4        # dphi(3,1)
3 2 0.5        # dphi(3,2)
3 3 0.6        # dphi(3,3)
Block WCDD
2 3 2 3 1.0    # dd(2,3,2,3)
Block WCPHIQ3
1 3 1.0        # phiq3(1,3)
```

WCsInput.dat

Simple text file

Inspired by the SLHA

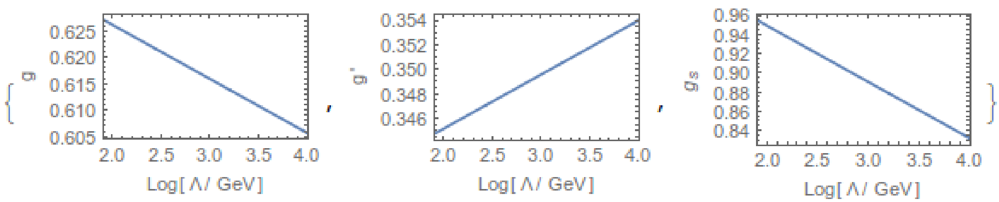
Similar format for the output file

Also possible to give input directly on the notebook

Results after SMEFTrunner

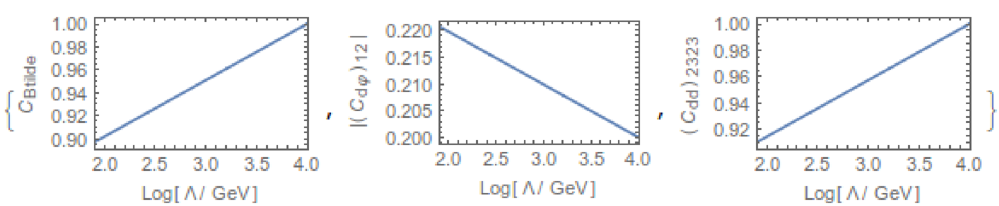
```
In[7]:= (* The results can also be plotted as a function of the energy scale *)

In[8]:= (* Gauge couplings *)
   plotGauge1 = Plot[outSMEFTrunner[[1]], {t, tLOW, tHIGH}, Frame → True, Axes → False, PlotRange → {{tLOW, tHIGH}, Automatic},
      FrameLabel → {"Log[Λ/GeV]", "g", None, None}];
   plotGauge2 = Plot[outSMEFTrunner[[2]], {t, tLOW, tHIGH}, Frame → True, Axes → False, PlotRange → {{tLOW, tHIGH}, Automatic},
      FrameLabel → {"Log[Λ/GeV]", "g'", None, None}];
   plotGauge3 = Plot[outSMEFTrunner[[3]], {t, tLOW, tHIGH}, Frame → True, Axes → False, PlotRange → {{tLOW, tHIGH}, Automatic},
      FrameLabel → {"Log[Λ/GeV]", "gs", None, None}];
   plotGauge = {plotGauge1, plotGauge2, plotGauge3}
```

Out[11]=




```
In[12]:= (* Wilson coefficients *)
   plotWC1 = Plot[outSMEFTrunner[[48]], {t, tLOW, tHIGH}, Frame → True, Axes → False, PlotRange → {{tLOW, tHIGH}, Automatic},
      FrameLabel → {"Log[Λ/GeV]", "CBtilde", None, None}];
   plotWC2 = Plot[Abs[outSMEFTrunner[[61]]], {t, tLOW, tHIGH}, Frame → True, Axes → False, PlotRange → {{tLOW, tHIGH}, Automatic},
      FrameLabel → {"Log[Λ/GeV]", "|(Cdφ)12|", None, None}];
   plotWC3 = Plot[outSMEFTrunner[[443]], {t, tLOW, tHIGH}, Frame → True, Axes → False, PlotRange → {{tLOW, tHIGH}, Automatic},
      FrameLabel → {"Log[Λ/GeV]", "(Cdd)2323", None, None}];
   plotWC = {plotWC1, plotWC2, plotWC3}
```

Out[15]=



Another simple program: analytics

A DsixTools Program

This notebook shows how to use the SMEFTrunner module to study SMEFT β functions analytically.

```
SetDirectory[NotebookDirectory[]];
```

Start DsixTools

```
Needs["DsixTools`"]
```

Set CP conservation

```
CPV = 0;
```

Load SMEFTrunner module

```
LoadModule["SMEFTrunner"]
```

Compute β functions

```
GetBeta;
```

Results

```
In[6]:= (* Let us compute β1q(1) and β1q(3) assuming top dominance and no NP effects in the 1st fermion family *)

In[7]:= (* Top dominance approximation *)
top = {GD[i_, j_] :> 0, GE[i_, j_] :> 0, GU[i_, j_] :> If[i == j == 3, Vtb yt, If[i == 2 && j == 3, Vts yt, 0]]};

In[8]:= (* No NP in 1st family *)
WCs2F = {φL1, φL3, φQ1, φQ3};
WCs4F = {LQ1, LQ3, LU, QE, QU1, QU8, QD1, QD8, QQ1, QQ3};
nofirst2F = Table[Part[WCs2F, i][a_, b_] → If[AnyTrue[{a, b}, # == 1 &], 0, 1] Part[WCs2F, i][a, b], {i, 1, Length[WCs2F]}];
nofirst4F = Table[Part[WCs4F, i][a_, b_, c_, d_] → If[AnyTrue[{a, b, c, d}, # == 1 &], 0, 1] Part[WCs4F, i][a, b, c, d],
   {i, 1, Length[WCs4F]}];
nofirst = Join[nofirst2F, nofirst4F];

In[13]:= βlq1 = β[lq1][[2, 2, 2, 3]] /. top /. nofirst // Expand
```

Out[13]= $\frac{1}{2}$ Vtb Vts yt^2 LQ1[2, 2, 2, 2] − $\frac{1}{3}$ gp^2 LQ1[2, 2, 2, 3] + $\frac{1}{2}$ Vtb^2 yt^2 LQ1[2, 2, 2, 3] + $\frac{1}{2}$ Vts^2 yt^2 LQ1[2, 2, 2, 3] + $\frac{1}{2}$ Vtb Vts yt^2 LQ1[2, 2, 3, 3] + $\frac{2}{3}$ gp^2 LQ1[3, 3, 2, 3] + 9 g^2 LQ3[2, 2, 2, 3] − Vtb Vts yt^2 LU[2, 2, 3, 3] + $\frac{2}{3}$ gp^2 QD1[2, 3, 2, 2] + $\frac{2}{3}$ gp^2 QD1[2, 3, 3, 3] + $\frac{2}{3}$ gp^2 QE[2, 3, 2, 2] + $\frac{2}{3}$ gp^2 QE[2, 3, 3, 3] − $\frac{2}{9}$ gp^2 QQ1[2, 2, 2, 3] − $\frac{4}{3}$ gp^2 QQ1[2, 3, 2, 2] − $\frac{14}{9}$ gp^2 QQ1[2, 3, 3, 3] − $\frac{2}{3}$ gp^2 QQ3[2, 2, 2, 3] − $\frac{2}{3}$ gp^2 QQ3[2, 3, 3, 3] − $\frac{4}{3}$ gp^2 QU1[2, 3, 2, 2] − $\frac{4}{3}$ gp^2 QU1[2, 3, 3, 3] + Vtb Vts yt^2 φL1[2, 2] − $\frac{1}{3}$ gp^2 φQ1[2, 3]

```
In[14]:= βlq3 = β[lq3][[2, 2, 2, 3]] /. top /. nofirst // Expand
```

Out[14]= 3 g^2 LQ1[2, 2, 2, 3] + $\frac{1}{2}$ Vtb Vts yt^2 LQ3[2, 2, 2, 2] − $\frac{16}{3}$ g^2 LQ3[2, 2, 2, 3] − gp^2 LQ3[2, 2, 2, 3] + $\frac{1}{2}$ Vtb^2 yt^2 LQ3[2, 2, 2, 3] + $\frac{1}{2}$ Vts^2 yt^2 LQ3[2, 2, 2, 3] + $\frac{1}{2}$ Vtb Vts yt^2 LQ3[2, 2, 3, 3] + $\frac{2}{3}$ g^2 LQ3[3, 3, 2, 3] + $\frac{2}{3}$ g^2 QQ1[2, 2, 2, 3] + $\frac{2}{3}$ g^2 QQ1[2, 3, 3, 3] − $\frac{2}{3}$ g^2 QQ3[2, 2, 2, 3] + 4 g^2 QQ3[2, 3, 2, 2] + $\frac{10}{3}$ g^2 QQ3[2, 3, 3, 3] − Vtb Vts yt^2 φL3[2, 2] + $\frac{1}{3}$ g^2 φQ3[2, 3]