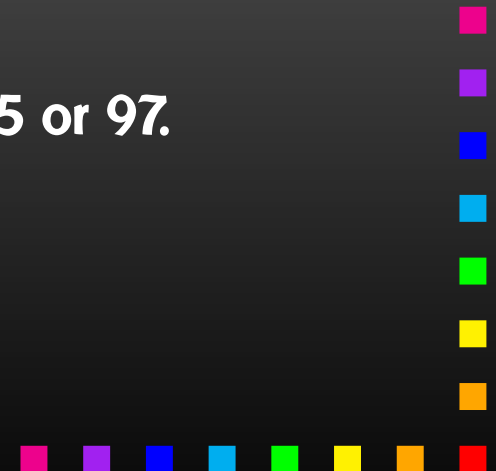


# FormCalc 9 and Extensions

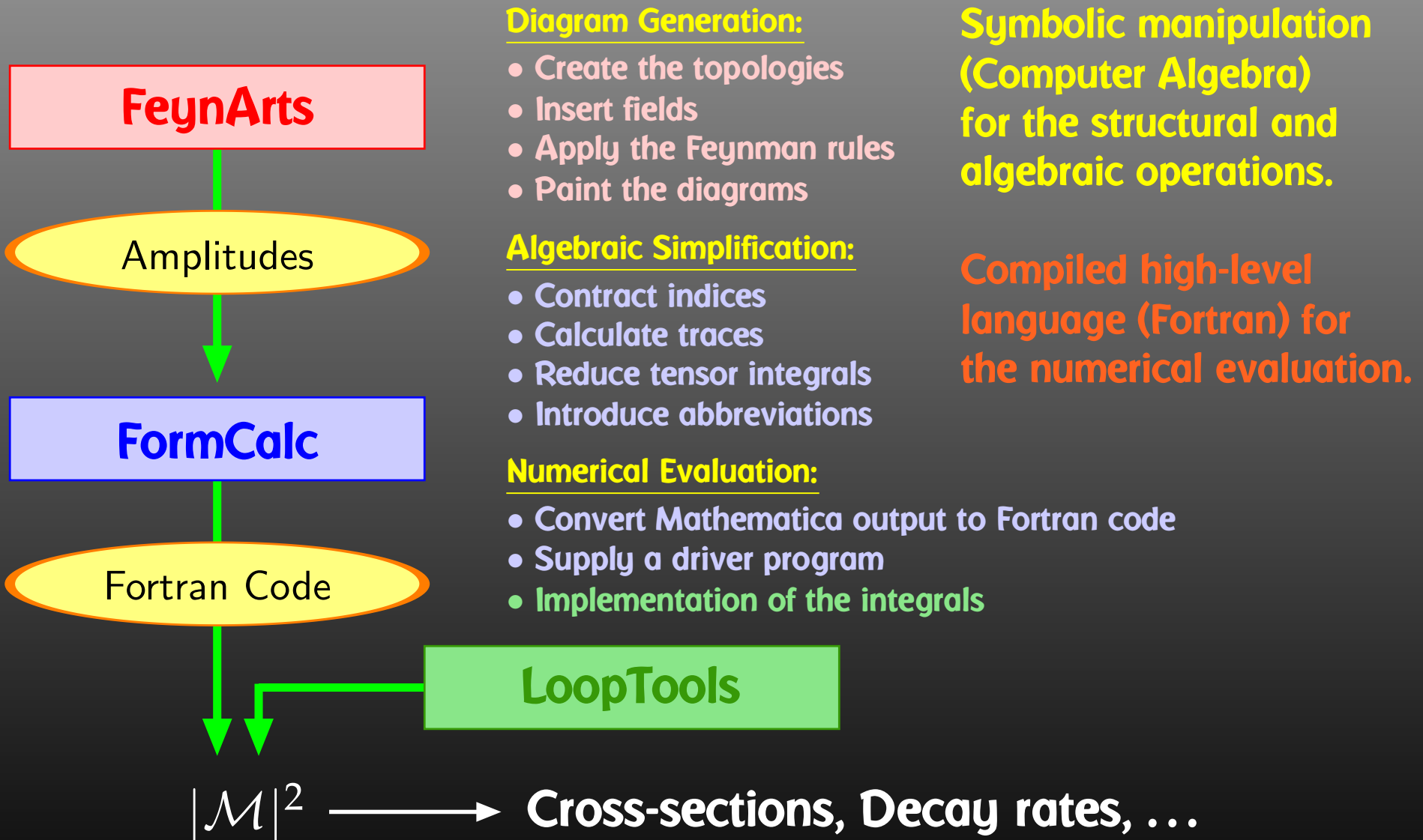
Thomas Hahn

Max-Planck-Institut für Physik  
München

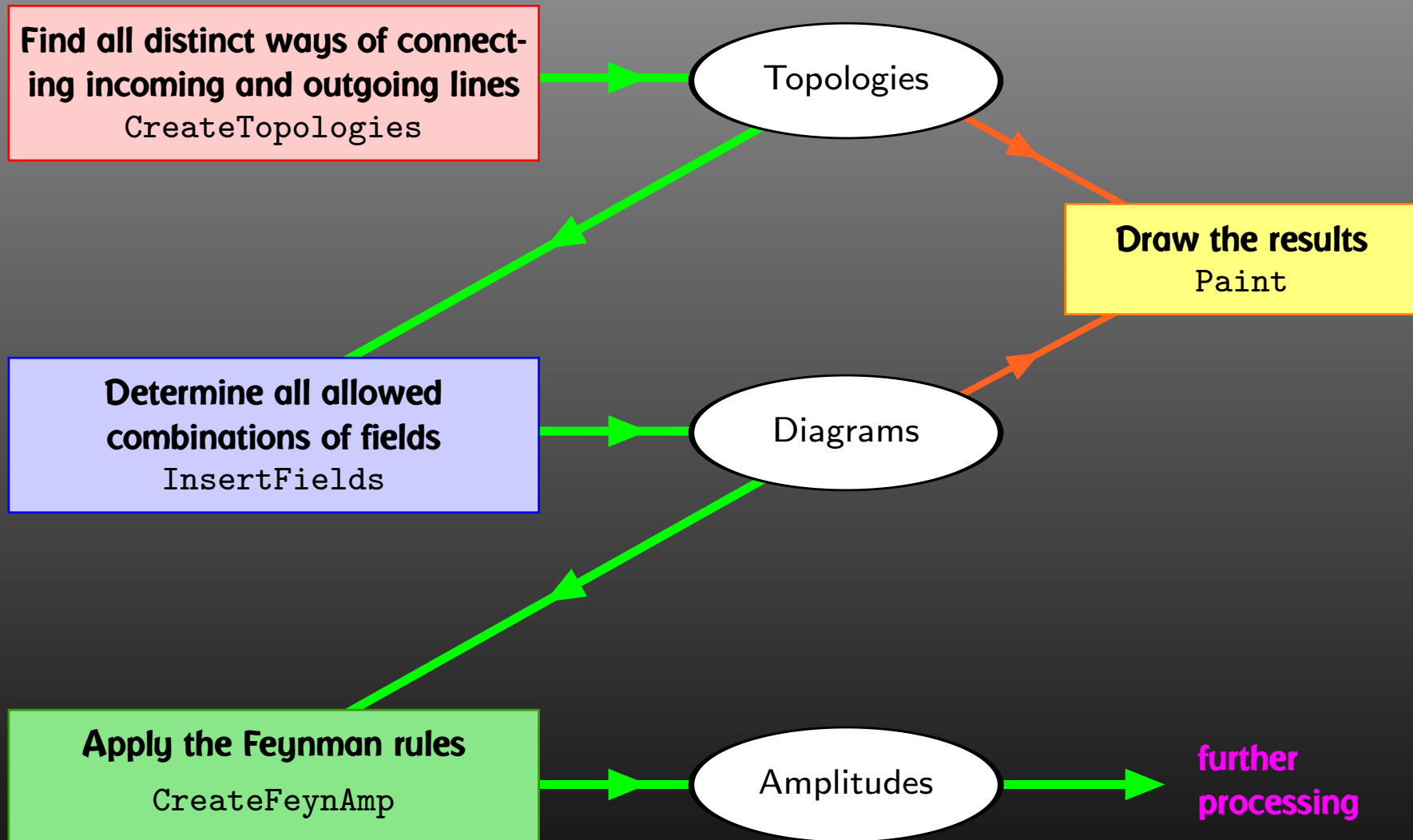
Need not skip Version 9 since there's no FormCalc 95 or 97.



# Diagram Evaluation in FeynArts, FormCalc, LoopTools



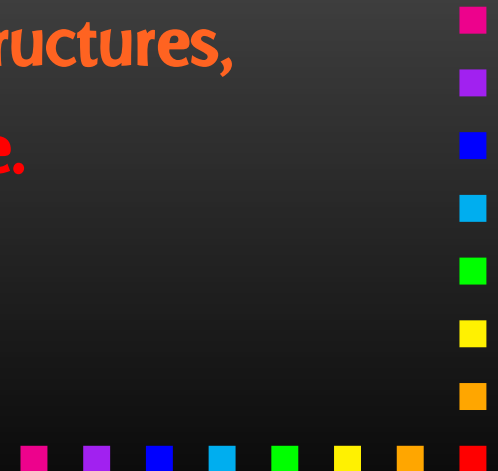
# FeynArts



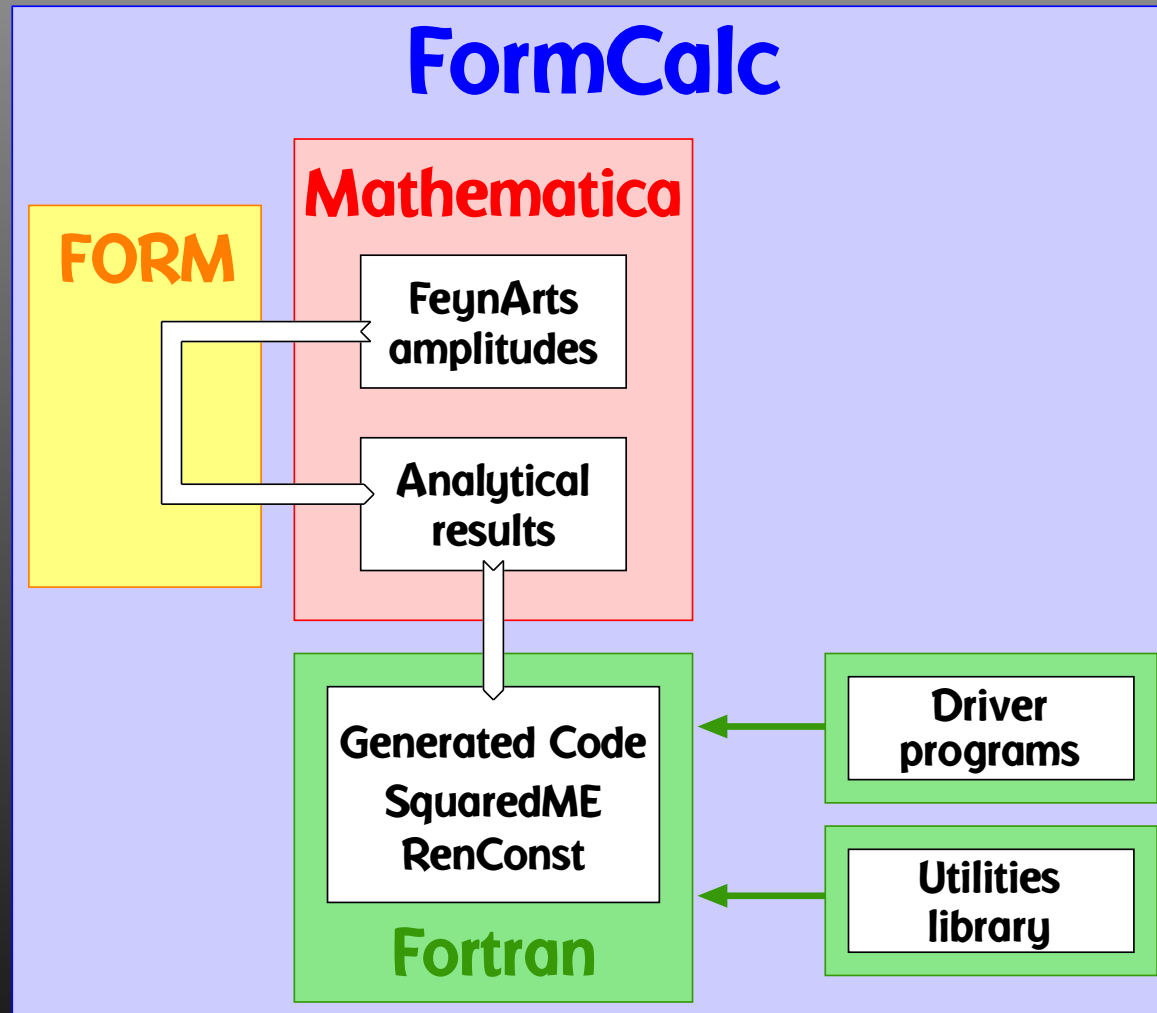
# Algebraic Simplification

A number of steps have to be done analytically:

- contract indices as far as possible,
- evaluate fermion traces,
- perform the tensor reduction,
- add local terms arising from  $D$ ·(divergent integral) (dim reg + dim red),
- simplify open fermion chains,
- simplify and compute the square of  $SU(N)$  structures,
- “compactify” the results as much as possible.



# FormCalc Internals



# FormCalc 9

*Work in collaboration with Christian Schappacher*

‘Collected Wisdom’ from several nontrivial SUSY calculations:  
arXiv:1511.06002, 1410.2787, 1112.2830, 1111.7289

‘Extended support’ for MSSMCT model file.

## New Features:

- Combining processes with (almost) arbitrary kinematics.
- Improved code generation + drivers.
- Optional output of non-vectorized code.
- Added various QCD functions (running,  $\overline{\text{MS}} \rightarrow \overline{\text{DR}}$ , etc.).
- Improvements in makefiles, build.



# Combining Processes

- Generate each (partonic) process.
- Put description for each in `partonic.h`:

```
#define PID 1
#define PARTON1 i
#define PARTON2 j
#include "process/specs.h"
#include "parton.h"
```

- **configure, compile.**
- **Typically used in hadronic computations, but can do also**  
e.g.  $e^+e^- \rightarrow \{e^+e^-, \mu^+\mu^-, \tau^+\tau^-\}$ .
- **Can combine only processes with the same number of external legs (same phase-space dim).**



# Veto Cuts

- Formerly **only selected one-particle cuts** available, applied by **restricting the integration bounds**.
- Now available: ‘Veto’ cuts which **set the cross-section to zero** if the cut condition applies:

CUT\_E(*i*)

CUT\_k(*i*)

CUT\_ET(*i*)

CUT\_kT(*i*)

CUT\_y(*i*)

CUT\_eta(*i*)

CUT\_deltatheta(*i*)

CUT\_cosdeltatheta(*i*)

CUT\_R(*i,j*)

CUT\_rho(*i,j*)

CUT\_deltay(*i,j*)

CUT\_deltaeta(*i,j*)

CUT\_yprod(*i,j*)

CUT\_etaprod(*i,j*)

CUT\_deltaalpha(*i,j*)

CUT\_cosdeltaalpha(*i,j*)

CUT\_invmass(*i,j*)

straightforward to add more

- **Construct cuts in run.F, e.g.**

```
#define CUT1 CUT_kT(3) .gt. 10
```

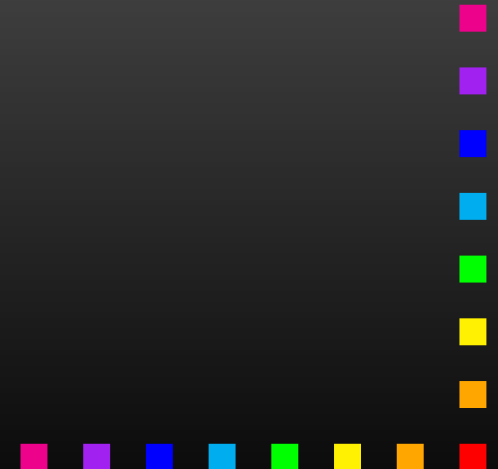




# Non-vectorized Code

- Generated code is **by default vectorized** in the helicities of the external particles.
- Not always useful, e.g. if one wishes to **include the generated code in own code** (adds complexity).
- Can now remove all vector instructions/macros with

```
SetLanguage["Fortran", "novec"]  
SetLanguage["C", "novec"]
```



# Debugging and Checking Instructions

- **Debugging statements** already available for long:

```
var = expr  
DEB("var", var)
```

- **New: Checking statements** of the form

```
CHK_PRE(var)  
var = expr  
CHK_POST("var", var)
```

- **Implement e.g. as**

```
#define DEB(tag,var) print *, tag, var  
#define CHK_PRE(var) tmp = var  
#define CHK_POST(tag,var) if(abs(var - tmp) .gt. 1D7) stop tag
```

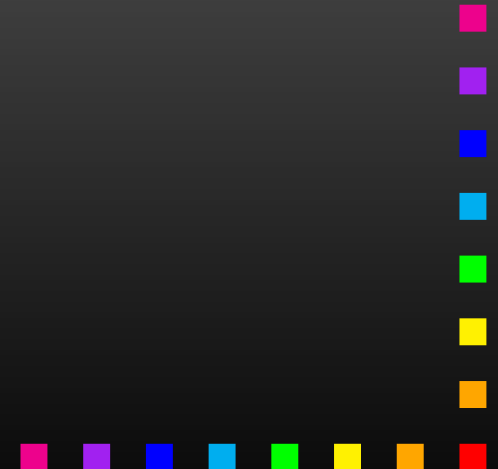
- **Very efficient for finding problems with a particular parameter point** (in a ton of generated code).



# New/improved QCD Functions

Added various QCD utility functions:

- $\alpha_s^{\overline{\text{MS}}}(Q)$  up to 4L, from RunDec [hep-ph/0004189],
- $m_q^{\overline{\text{MS}}}(Q_1) \rightarrow m_q^{\overline{\text{MS}}}(Q_2)$  up to 3L, from RunDec,
- $m_q^{\overline{\text{MS}}}(Q) \rightarrow m_q^{\text{OS}}$  up to 3L, from RunDec,
- $\alpha_s^{\overline{\text{MS}}} \rightarrow \alpha_s^{\overline{\text{DR}}}$  up to 2L, from arXiv:0706.2953.



# More Renormalization Constants

- Can now define Renormalization constants **with arbitrary heads**, e.g. mass shifts. Each head goes in separate file:

```
RenConst[.] := ... → RenConst.F, .h      (normal)
MassShift[.] := ... → MassShift.F, .h     (new)
```

- Creates **subsets of RC-like objects**, i.e. results of loop calculation but constant (external kinematics fixed).

Example: shifts used in the iterative determination of  $\Delta b$ :

RCs need not all be recomputed in each iteration = faster.

- **Computation:**

```
CalcRenConst[expr, MassShift]           (manual)
$RenConst = {RenConst, MassShift}       (automatic)
```

**i.e. the Model file can request that MassShift be computed by adding it to \$RenConst.**



# Extensions

Often special requirements:

- **Resummations** (e.g. *hbb* in MSSM),
- **Approximations** (e.g. gaugeless limit),
- **K-factors**,
- **Nontrivial renormalization.**

Software design so far:

- Mostly **'monolithic'** (one package does everything).
- Often controlled by **parameter cards**, not easy to use beyond intended purpose.
- May want to/must use other packages.



# Example: $\mathcal{O}(\alpha_t^2)$ MSSM Higgs-mass corrections

Work in collaboration with Sebastian Paßehr

- FeynHiggs computes MSSM Higgs mass corrections, including leading 2L contributions:  $\mathcal{O}(\alpha_s \alpha_t)$ ,  $\mathcal{O}(\alpha_t^2)$ .
- $\mathcal{O}(\alpha_t^2)$ : Diagrammatic 2L calculation in full cMSSM.

Hollik, Paßehr 2014

① **Unrenormalized 2L self-energies**

$$\Sigma_{hh}^{(2)}, \Sigma_{hH}^{(2)}, \Sigma_{hA}^{(2)}, \Sigma_{HH}^{(2)}, \Sigma_{HA}^{(2)}, \Sigma_{AA}^{(2)}, \Sigma_{H^+H^-}^{(2)}$$

in gaugeless approximation at  $p^2 = 0$  at  $\mathcal{O}(\alpha_t^2)$ .

② **1L diagrams with insertions of 1L counterterms.**

③ **2L counterterms for ①.**

④ **2L tadpoles  $T_h^{(2)}$ ,  $T_H^{(2)}$ ,  $T_A^{(2)}$  at  $\mathcal{O}(\alpha_t^2)$  appearing in ③.**



# Template for Calculations

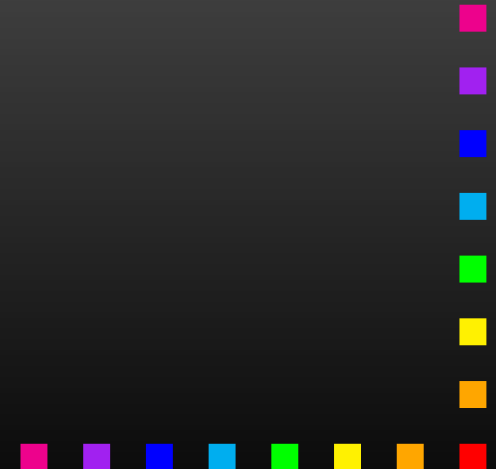
- Starting point: several Mathematica 8 (*sic*) notebooks with parallel instructions poured all over, duplicate code (e.g. gaugeless limit implemented multiply), etc.
- Reorganization of entire procedure:
  - Break calculation into **several steps**.
  - Implement each step as **independent program** (invoked from command line).
  - In lieu of 'in vivo' debugging **keep detailed logs**.
  - Coordinate everything through a **makefile**.
- Outcome: **Template for 2L calculation in nontrivial model with nontrivial renormalization with optimized output.**
- No single control program (e.g. single Mathematica session) like in package's demo programs.



# Wheels we don't reinvent

## Will use external packages

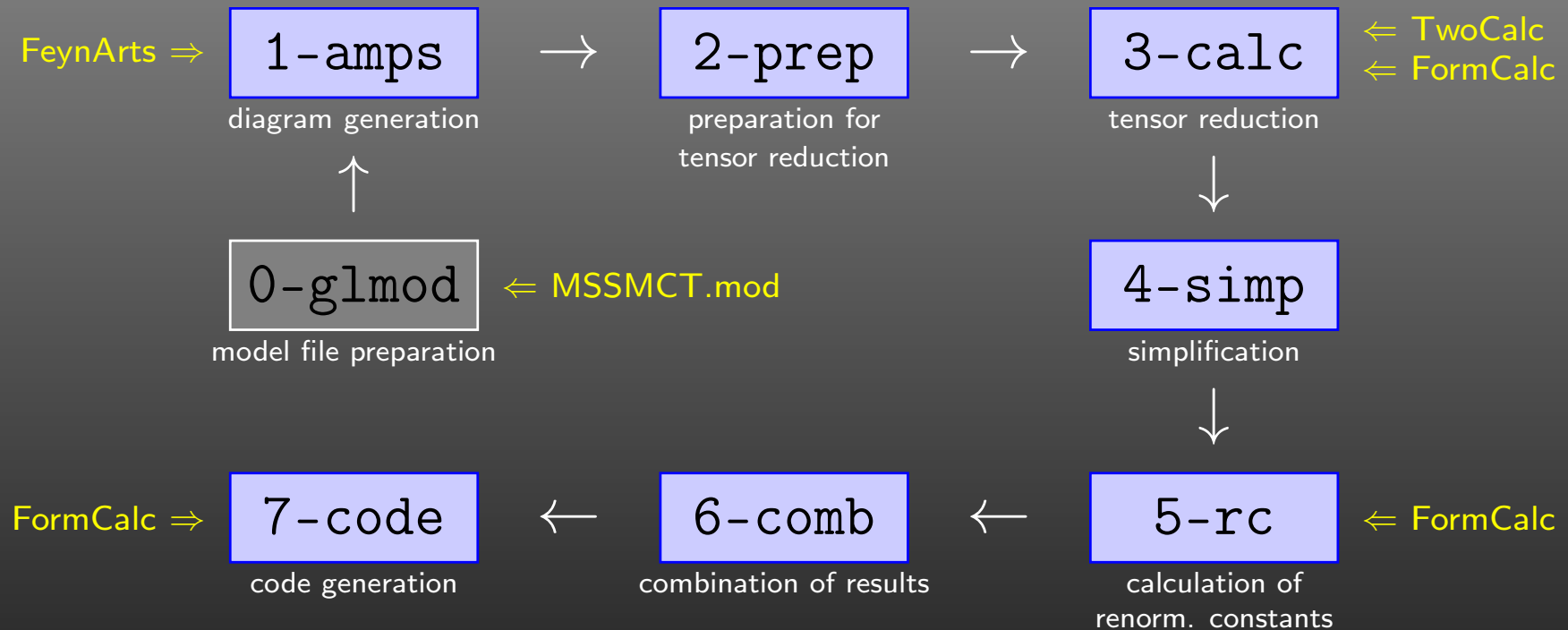
- **FeynArts** (for diagram generation),  
Hahn 2001–2015
- **MSSMCT.mod** (MSSM model file with 1L counterterms),  
Schappacher et al. 2014
- **FormCalc** (for 1L tensor reduction, code generation),  
Hahn 1996–2015
- **TwoCalc** (for 2L tensor reduction).  
Weiglein et al. 1992, 1994





# Steps of the Calculation

Calculation split into 7 (8) steps:



# Script Structure

- **Shell scripts** (/bin/sh), run from command line as e.g.  
`./1-amps arg1 arg2`
- `arg1` = `h0h0`, `h0HH`, `h0A0`, `HHHH`, `HHA0`, `A0A0`, `HmHp` (**self-energies**),  
`h0`, `HH`, `A0` (**tadpoles**).
- `arg2` = `0` for virtual 2L diagrams,  
`1` for 1L diagrams with 1L counterterms.

- **Inputs/outputs defined in first few lines, e.g.**

```
in=m/$1/2-prep.$2  
out=m/$1/3-calc.$2
```

- **Symbolic output + log files go to 'm' subdirectory.**  
**Log file = Output file + .log.gz**
- **Fortran code goes to 'f' subdirectory.**



# Scripting Mathematica

## Efficient batch processing with Mathematica:

Put everything into a script, using **sh's Here documents**:

```
#!/bin/sh ..... Shell Magic
math << \_EOF_ ..... start Here document (note the \)
  << FeynArts'
  << FormCalc'
  top = CreateTopologies[...];
  ...
\_EOF_ ..... end Here document
```

Everything between “<< *tag*” and “*tag*” goes to Mathematica as if it were typed from the keyboard.

Note the “\” before *tag*, it makes the shell pass everything literally to Mathematica, without shell substitutions.



# Scripting Mathematica

- Everything contained in **one compact shell script**, even if it involves several Mathematica sessions.
- Can combine with arbitrary shell programming, e.g. can use **command-line arguments** efficiently:

```
#!/bin/sh
math -run "arg1=$1" -run "arg2=$2" ... << \END
...
END
```

- Can easily be **run in the background**, or combined with utilities such as **make**.

**Debugging hint:** **-x flag** makes shell echo every statement,

```
#!/bin/sh -x
```



# Step 0: Gaugeless Limit

Gaugeless approximation:

- ① Set gauge couplings  $g, g' = 0 \Rightarrow M_W, M_Z = 0$ .
- ② Keep finite weak mixing angle.
- ③ Keep  $\frac{\delta M_W^2}{M_W^2}$  and  $\frac{\delta M_Z^2}{M_Z^2}$  finite.

Must set  $m_b = 0$  so that  $\mathcal{O}(\alpha_t^2)$  corrections form supersymmetric and gauge-invariant subset.

Most efficient to **modify Feynman rules** (not ③, though):

- Load MSSMCT.mod model file.
- Modify couplings, remove zero ones.
- Write out MSSMCTg1.mod model file.



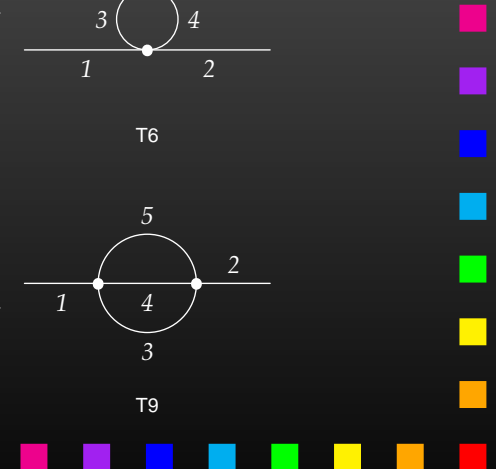
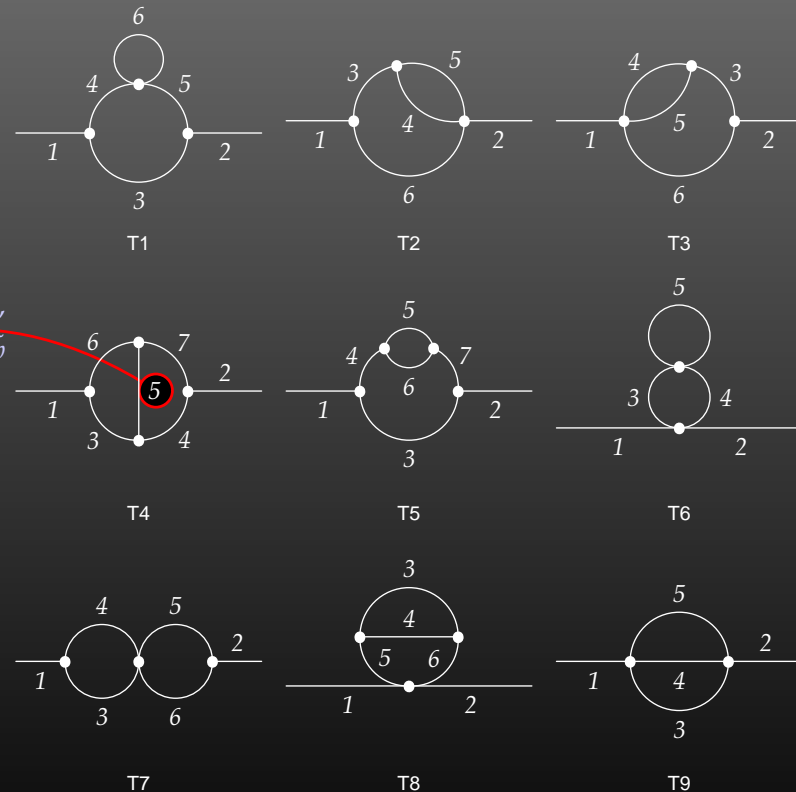
# Step 1: Diagram Generation

- Generate 2L virtual and 1L+counterterm diagrams using wrappers for FeynArts functions.

Simple diagram selection functions, e.g.

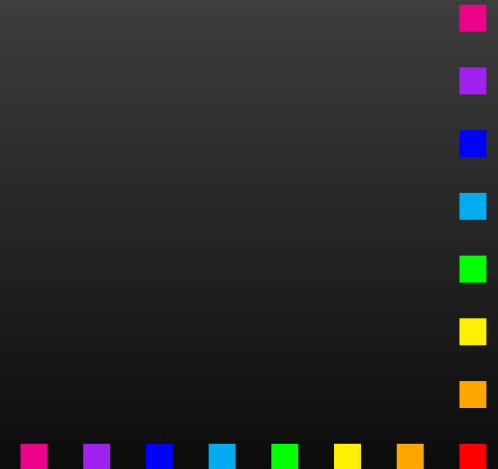
```

sel[0][S[_] -> S[_]] = {
  t[3] && htb[6],
  t[3] && tb[6],
  t[3] && tb[6],
  t[3] && t[4] && htb[5],
  t[3] && htb[5|6],
  t[3] && htb[5],
  t[3] && t[5],
  t[5] && ht[3|4],
  t[3|4|5] && ht[3|4|5] }
  
```



## Step 2: Preparation for Tensor Reduction

- Take  $p^2 \rightarrow 0$  limit.
- Simplify ubiquitous sfermion mixing matrices  $U_{ij}$ , mostly by exploiting unitarity ( $\sim 50\%$  size reduction).



# Efficiently Exploit Unitarity in Mathematica

Unitarity of 2 x 2 matrix:  $UU^\dagger = U^\dagger U = \mathbb{1}$ , i.e.

$$U_{11}U_{11}^* + U_{12}U_{12}^* = 1, \quad U_{11}U_{21}^* + U_{12}U_{22}^* = 0,$$

$$U_{21}U_{21}^* + U_{22}U_{22}^* = 1, \quad U_{21}U_{11}^* + U_{22}U_{12}^* = 0,$$

$$U_{11}U_{11}^* + U_{21}U_{21}^* = 1, \quad U_{11}U_{12}^* + U_{21}U_{22}^* = 0,$$

$$U_{12}U_{12}^* + U_{22}U_{22}^* = 1, \quad U_{12}U_{11}^* + U_{22}U_{21}^* = 0.$$

**Problem:** Simplify will **rarely arrange the  $U$ 's in just the way** that these rules can be applied directly.

**Solution:** Introduce auxiliary symbols which **immediately deliver** the r.h.s. once Simplify considers the l.h.s., i.e. **increase the 'incentive'** for Simplify to use the r.h.s.

**But:** Upvalues work only one level deep.





# Efficiently Exploit Unitarity in Mathematica

## Introduce

$$USf [1, j] USfC [1, j] \rightarrow UCSf [1, j],$$

$$USf [2, j] USfC [2, j] \rightarrow UCSf [2, j],$$

$$USf [1, j] USfC [2, j] \rightarrow UCSf [3, j], \quad + \text{ ditto for 1}^{\text{st}} \text{ index}$$

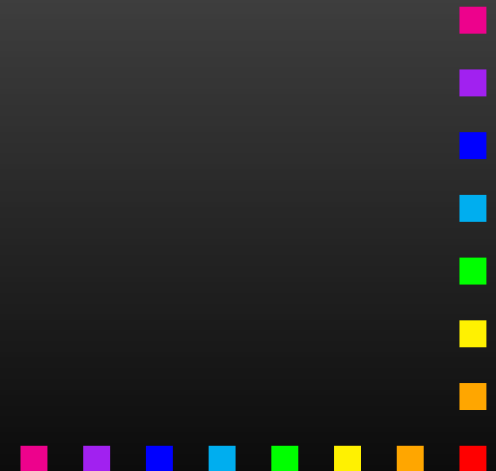
and formulate unitarity for the UCSf:

$$\begin{aligned} UCSf [2, 1] &= UCSf [1, 2]; & UCSf [3, 2] &= -UCSf [3, 1]; \\ UCSf [2, 2] &= UCSf [1, 1]; & UCSfC [3, 2] &= -UCSfC [3, 1]; \\ & & UCSf [2, 3] &= -UCSf [1, 3]; \\ \dots & & UCSfC [2, 3] &= -UCSfC [1, 3]; \end{aligned}$$



## Step 3: Tensor Reduction

- Relatively straightforward application of **TwoCalc** and **FormCalc** for tensor reduction.
- Observe: Need **two Mathematica sessions** since TwoCalc and FormCalc cannot be loaded into one session, easily accomodated in shell script.



## Step 4: Simplification

- Tensor reduction traditionally increases # of terms most.
- Step 4 reduces size before combination of results.
- Empirical simplification recipe.
- **'DiagMark' trick** (D. Stöckinger):
  - Introduce `DiagMark[mi]` where  $m_i$  = masses in loop in FeynArts output.
  - Few simplifications can be made between parts with different `DiagMark`  $\Rightarrow$  Can apply simplification as  
`Collect[amp, _DiagMark, simpfunc]`
  - Much faster.



## Step 5: Calculation of Renormalization Constants

- Compute 1L renormalization constants (RC) with FormCalc.

- Substitute explicit mass dependence in

$$dM_{Vsq1} \rightarrow M_{V2} dM_{Vsq1} M_{V2} \quad (V = W, Z)$$

such that gaugeless limit can be taken safely.

- Expand in  $\varepsilon$ , collect powers for easier handling later, e.g.

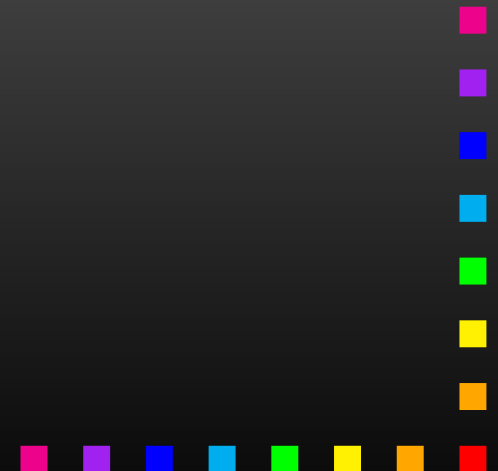
$$\left\{ \begin{array}{l} dM_{f1}[3,3] \rightarrow RC[-1, dM_{f1}[-1,3,3]] + \\ \quad \quad \quad RC[0, dM_{f1}[0,3,3]], \end{array} \right\} \text{ expansion}$$

$$\left\{ \begin{array}{l} dM_{f1}[-1,3,3] \rightarrow \dots, \\ dM_{f1}[0,3,3] \rightarrow \dots \end{array} \right\} \text{ actual expressions for } \varepsilon\text{-coeffs}$$



## Step 6: Combination of Results

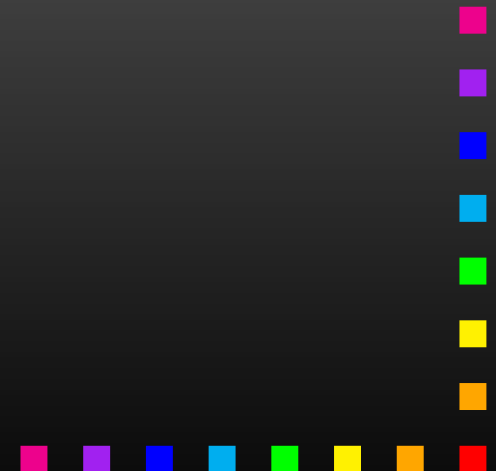
- Expand amplitude in  $\varepsilon$  (similar as RC).
- Insert RCs.
- Add genuine 2L counterterms (hand-coded).
- Pick only  $\varepsilon^0$  term (unless debug flag set).
- Perform final simplification.



## Step 7: Code Generation

- Introduce abbreviations to shorten code.
- Write out Fortran code using FormCalc's code-generation functions.
- Add static code which computes e.g. the necessary parameters for the generated code.
- Total final code size: 350 kBytes.

More details in [arXiv:1508.00562](https://arxiv.org/abs/1508.00562).



# Universality w.r.t. Loop Order

✓ **Model** → **Model file** → **Diagrams**

Diagram Generation

✓ **Fermion algebra (traces, Dirac eq)**

Symbolic Simplification

✓ **Color algebra ( $SU(N)$  traces)**

✗ **Perform tensor reduction, or  
Isolate integrals for OPP**

✓ **Other simplifications (e.g. Fierz, abbreviations)**

✓ **Code generation (Fortran, C/C++)**

Numerical Evaluation

✓ **Phase-space integration**

✗ **Evaluation of loop integrals**

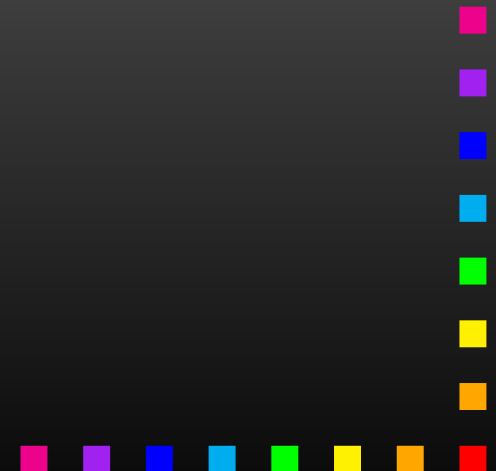


# Loopedia Database

*Work in collaboration with Viktor Papara et al.*

Question: Which master integrals are presently known?

- Vast literature.
- No central repository yet.
- **Loopedia** first attempt at community-driven database.
- Infrastructure by Loopedia group, integrals to be added by external researchers.
- Mixture of
  - Literature references,
  - Analytic results (Maple, Mathematica),
  - Numeric results (Fortran, C++ programs).





# Loopedia Input Scheme

- **Specify integral via Graph Info** (e.g. Adjacency List)  
Graph is drawn automatically
- **Select properties of Propagators and External Legs**  
Propagators: Mass, Power  
External Legs: Virtuality
- **Provide result and reference**  
Upload (text-) file with result  
Specify details in Comment field (or file)

Loopedia try-out interface currently not publicly available due to restructuring of storage format.

Contact [papara@mpp.mpg.de](mailto:papara@mpp.mpg.de) to be added to mailing list.



# Résumé

- **FormCalc 9 with many new features**, most adapted/generalized from real-life SUSY calculations:  
[feynarts.de/formcalc](http://feynarts.de/formcalc)
- **Suite of scripts** as by-product of  $\mathcal{O}(\alpha_t^2)$   $M_h$  project, may be used as **Template for similar calculations**:  
[feynhiggs.de](http://feynhiggs.de) (gen/tlsp directory)
- **Loopedia: database for loop integrals**  
no URL yet (early stage), contact [papara@mpp.mpg.de](mailto:papara@mpp.mpg.de)

