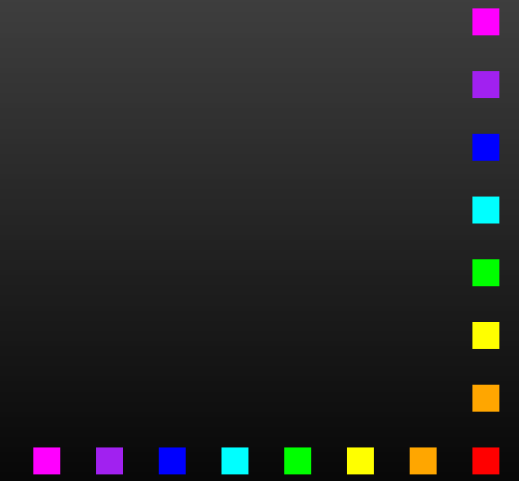
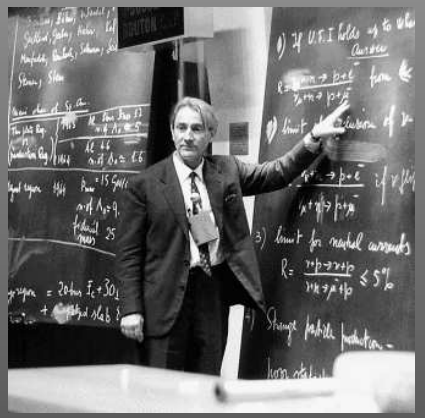
Automatic SUSY Calculations with *FeynArts* and *FormCalc*

Thomas Hahn

Max-Planck-Institut für Physik
München



What are Feynman Diagrams good for?

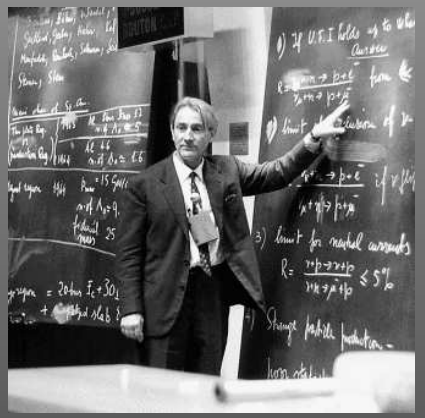


Theorists like

$$\mathcal{L}$$

Why: The Lagrangian defines the Theory.

What are Feynman Diagrams good for?



Theorists like

$\mathcal{L}$

Why: The Lagrangian defines the Theory.

Experimentalists like

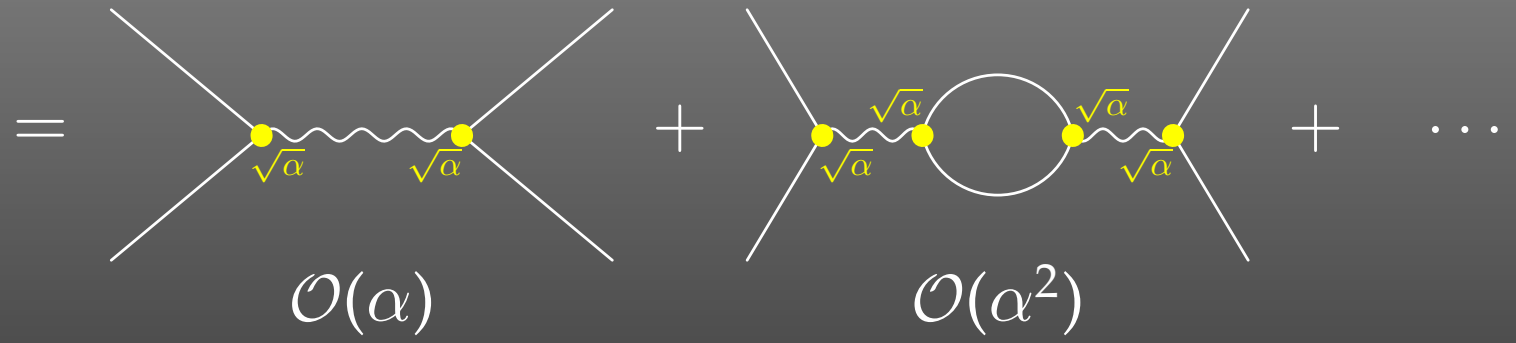
S

Why: Cross sections, decay rates, etc. follow directly from the S-Matrix.

What are Feynman Diagrams good for?

$S = \sum$ (Feynman Diagrams)

$=$ perturbation series in the coupling constant $\sqrt{\alpha}$



$\mathcal{L}$ determines the Feynman Rules which tell us how to translate the diagrams into formulas.

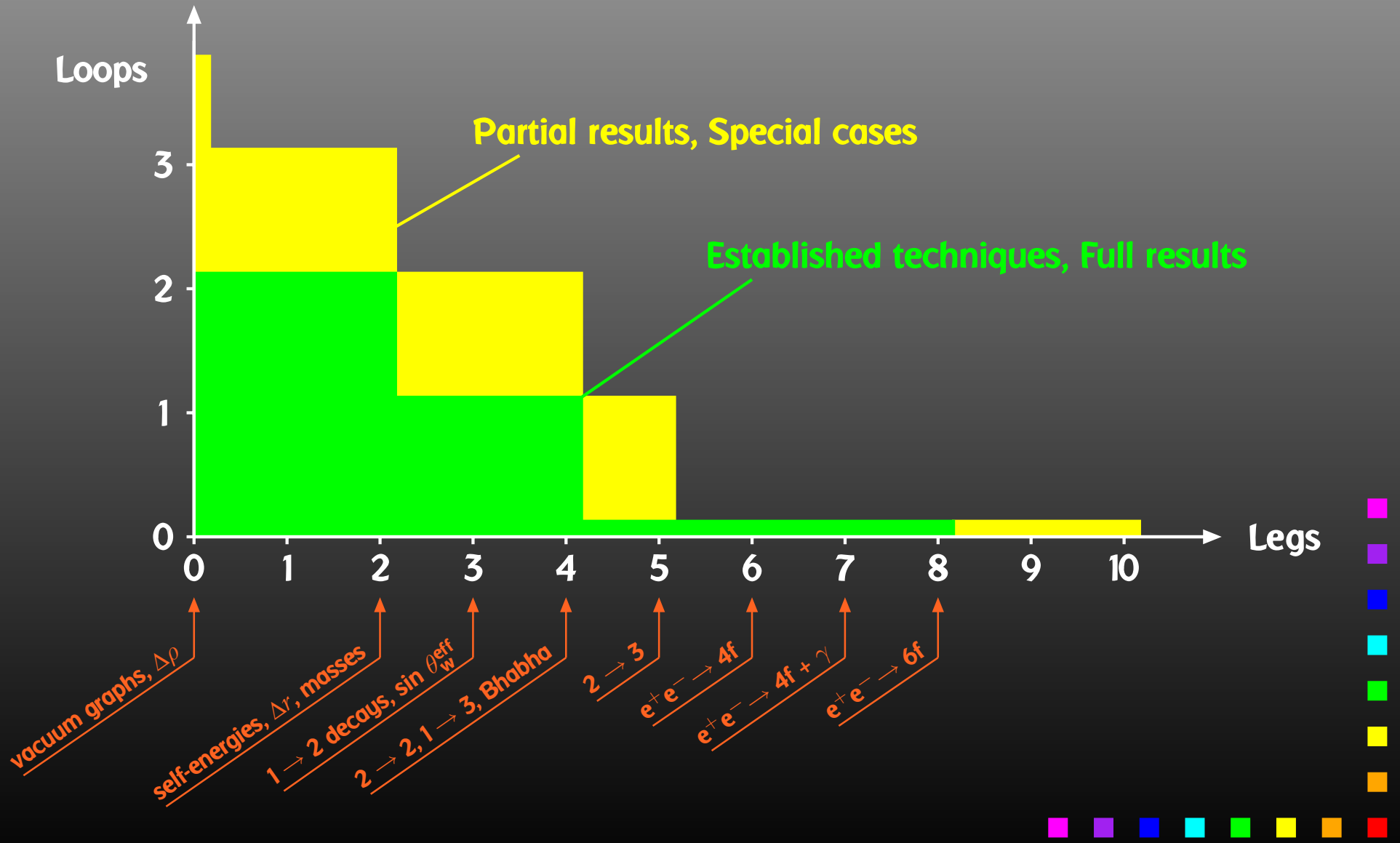
Accuracy of results $\longleftrightarrow$ Order in α $\longleftrightarrow$ # of Loops

State of the Art

# loops	0	1	2	3+
# $2 \to 2$ topologies	4	99	2214	50051
		"curse of topology"		
typical accuracy	10%	1%	.1%	.01%
general procedure known	yes	yes	$1 \to 1$	no
limits	$2 \to 8$	$2 \to 3$	$1 \to 2$	$1 \to 1$

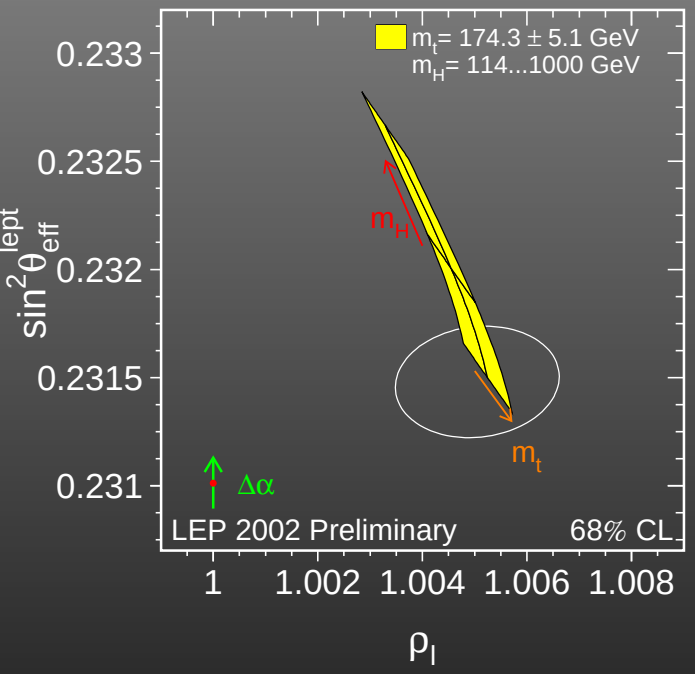
- Work on $2 \to 4$ in progress, desirable for $e^+e^- \to 4f$.
- More legs ($2 \to 2$) now feasible thanks to new technology (Remiddi, Gehrmann et al.).
- Only few results, approximations only (large-mass, large-momentum expansion).

State of the Art



Why Higher Orders?

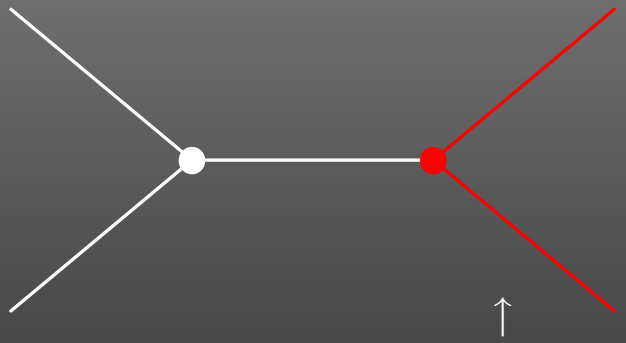
Precision: Higher Orders are seen experimentally



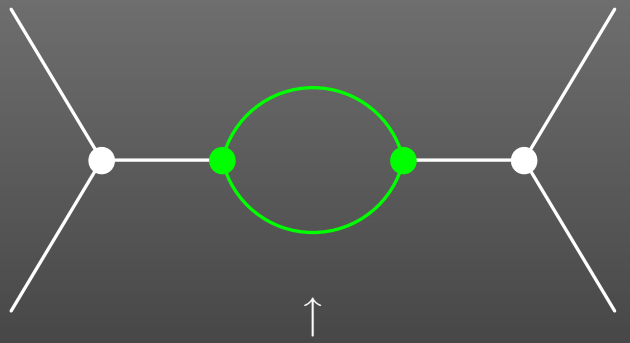
$10^{10} a_\mu = 11614097.29$	**QED**	**1-loop**
41321.76		**2-loop**
3014.19		**3-loop**
36.70		**4-loop**
.63		**5-loop**
690.6	**Had.**	
19.5	**EW**	**1-loop**
−4.3		**2-loop**
11659176	**theory, total**	
11659204	**exp (BNL 2002)**	

Why Higher Orders?

Indirect effects of particles beyond the kinematical limit



inaccessible
(too heavy to be produced)

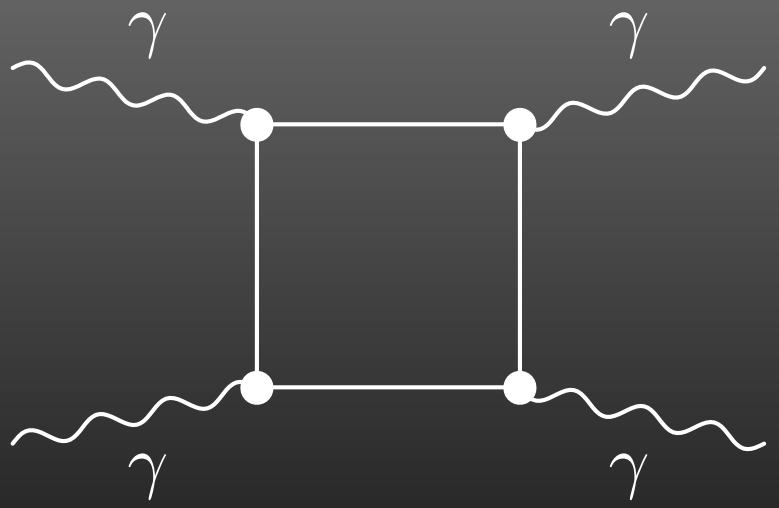


indirectly visible,
requires precision measurements

Why Higher Orders?

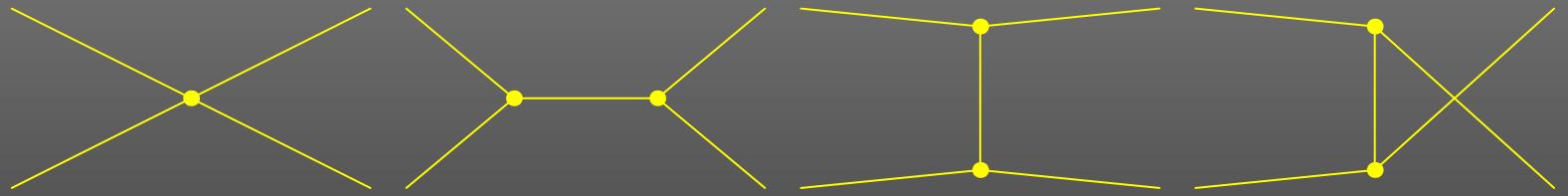
"Rare" (loop-mediated) events

e.g. light-by-light scattering:



How do *I* calculate Feynman Diagrams

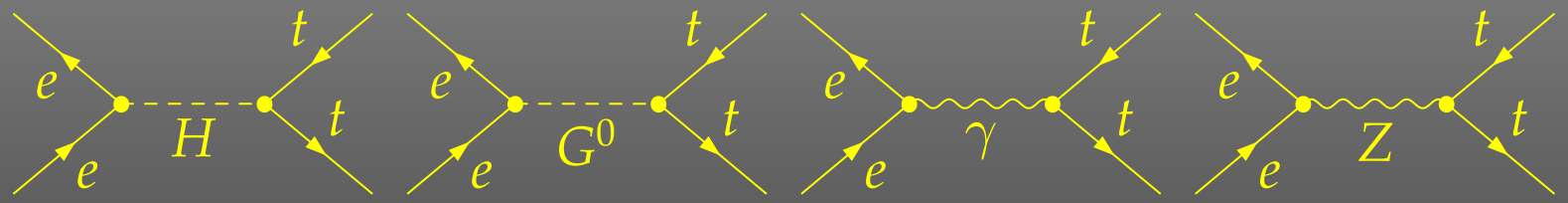
1. Draw all possible types of diagrams with the given number of loops and external legs



Topological task

How do *I* calculate Feynman Diagrams

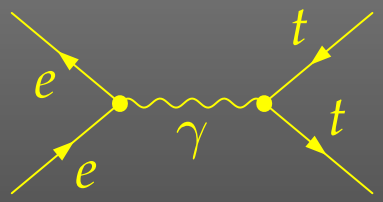
2. Figure out what particles can run on each type of diagram



Combinatorical task, requires physics input (model)

How do *I* calculate Feynman Diagrams

3. Translate the diagrams into formulas by applying the Feynman rules

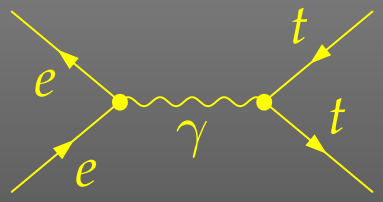


$$= \underbrace{\langle v_1 | \, \mathrm{i}e\gamma^\mu \, | u_2 \rangle}_{\text{left vertex}} \; \underbrace{\frac{g_{\mu\nu}}{(k_1+k_2)^2}}_{\text{propagator}} \; \underbrace{\langle u_4 | \left(-\tfrac{2}{3}\mathrm{i}e\gamma^\nu\right) | v_3 \rangle}_{\text{right vertex}}$$

Database look-up

How do *I* calculate Feynman Diagrams

4. Contract the indices, take the traces, etc.



$$= \frac{8\pi\alpha}{3s} F_1\,, \qquad F_1 = \langle v_1|\,\gamma_\mu\,|u_2\rangle\,\langle u_4|\,\gamma^\mu\,|u_3\rangle$$

Also, compute the fermionic matrix elements, e.g. by squaring and taking the trace:

$$\begin{aligned} |F_1|^2 &= \mathrm{Tr}\left\{(\not{k}_1 - m_e)\gamma_\mu(\not{k}_2 + m_e)\gamma_\nu\right\}\mathrm{Tr}\left\{(\not{k}_4 + m_t)\gamma^\mu(\not{k}_3 - m_t)\gamma^\nu\right\} \\ &= \tfrac{1}{2}s^2 + st + (m_e^2 + m_t^2 - t)^2 \end{aligned}$$

Algebraic simplification

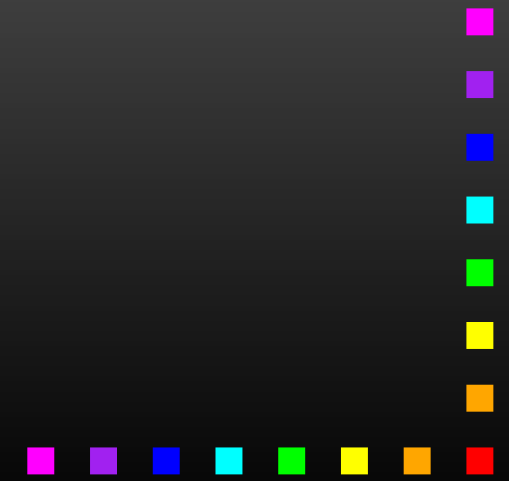
How do *I* calculate Feynman Diagrams

5. Write the results up as a ..... program
(put favourite language here)

5a. Debug that program

6. Run it to produce numerical values

Programming



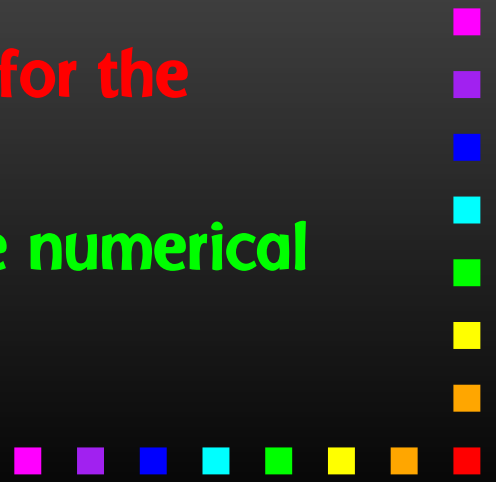
Programming Techniques

- Very different tasks at hand.
- Some objects must/should be handled symbolically, e.g. tensorial objects, Dirac traces, dimension (D vs. 4).
- Reliable results required even in the presence of large cancellations.
- Fast evaluation desirable (e.g. for Monte Carlos).

Hybrid Programming Techniques necessary

Symbolic manipulation (a.k.a. Computer Algebra) for the structural and algebraic operations.

Compiled high-level language (e.g. Fortran) for the numerical evaluation.



Packages

Comprehensive packages for Perturbative Calculations

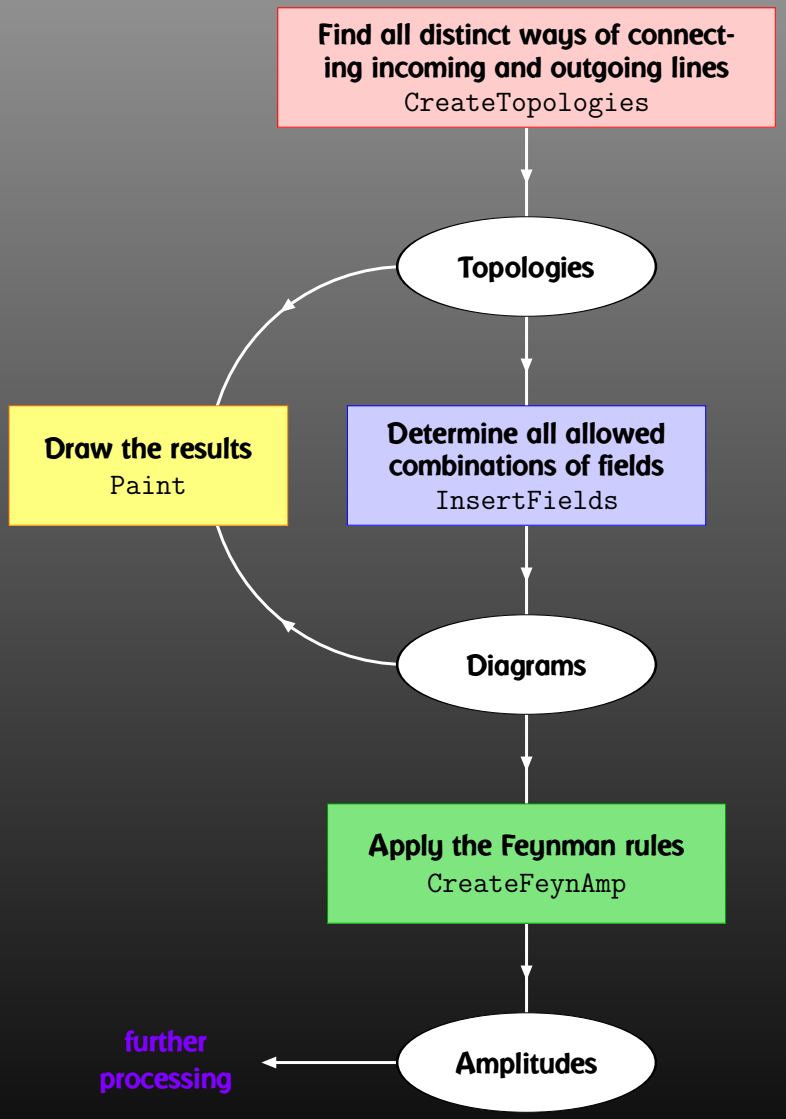
- Tree-level calculations only
- One-Loop calculations only
- "Arbitrary" loop order

Program Name	FeynArts	GRACE	CalcHEP	DIANA	MadGraph
Group	WÜ/KA/M	KEK	Dubna	Bielefeld	Madison
Core language	Mathematica	C	C, Fortran	C	Fortran
Components:					
Diagram generation	FeynArts	grc	CalcHEP	QGRAF	MadGraph
Diagram painting	FeynArts	gracefig	CalcHEP	DIANA	–
Algebraic simplif.	FormCalc FeynCalc	GRACE	CalcHEP	DIANA + FORM	–
Code generation	FormCalc	grcfort	CalcHEP	–	MadGraph
Libraries	LoopTools	chanel, bases, spring	CalcHEP	–	HELAS

How does the *computer* calculate Feynman Diagrams

Diagram generation	• Create the topologies • Insert fields • Apply the Feynman rules • Paint the diagrams	*FeynArts*
↓		
Algebraic simplification	• Contract indices • Calculate traces • Reduce tensor integrals • Introduce abbreviations	*FormCalc*
↓		
Numerical evaluation	• Convert *Mathematica* output to Fortran code • Supply a driver program • Implementation of the integrals	*FormCalc* *LoopTools*

FeynArts



EXAMPLE: generating the photon self-energy

```
top = CreateTopologies[ 1, 1 -> 1 ]
```

one loop

one incoming particle

one outgoing particle

```
Paint[top]
```

```
ins = InsertFields[ top, V[1] -> V[1],
        Model -> SM ]
```

use the Standard Model

the name of the photon in the "SM" model file

```
Paint[ins]
```

```
amp = CreateFeynAmp[ins]
```

```
amp >> PhotonSelfEnergy.amp
```

Three Levels of Fields

Generic level, e.g. `F, F, S`

$$C(F_1, F_2, S) = G_- \omega_- + G_+ \omega_+$$

Kinematical structure completely fixed, most algebraic simplifications (e.g. tensor reduction) can be carried out.

Classes level, e.g. `-F[2], F[1], S[3]`

$$\bar{\ell}_i \nu_j G: \quad G_- = -\frac{\mathrm{i}\, e\, m_{\ell,i}}{\sqrt{2} \sin\theta_w M_W} \delta_{ij}\,, \quad G_+ = 0$$

Coupling fixed except for i, j (can be summed in do-loop).

Particles level, e.g. `-F[2,{1}], F[1,{1}], S[3]`

insert fermion generation (1, 2, 3) for i and j

The Model Files

One has to set up, once and for all, a

- **Generic Model File** (seldomly changed) containing the generic part of the couplings,

Example: the `FFS` coupling

$$C(F,F,S) = G_- \omega_- + G_+ \omega_+ = \vec{G} \cdot \begin{pmatrix} \omega_- \\ \omega_+ \end{pmatrix}$$

```
AnalyticalCoupling[s1 F[j1, p1], s2 F[j2, p2], s3 S[j3, p3]]
== G[1][s1 F[j1], s2 F[j2], s3 S[j3]] .
     { NonCommutative[ ChiralityProjector[-1] ],
       NonCommutative[ ChiralityProjector[+1] ] }
```

The Model Files

One has to set up, once and for all, a

- **Classes Model File** (for each model)
  declaring the particles and the allowed couplings

Example: the $\bar{\ell}_i \nu_j G$ coupling in the Standard Model

$$\vec{G}(\bar{\ell}_i, \nu_j, G) = \begin{pmatrix} G_- \\ G_+ \end{pmatrix} = \begin{pmatrix} -\frac{\mathrm{i}\, e\, m_{\ell,i}}{\sqrt{2}\sin\theta_w M_W}\delta_{ij} \\ 0 \end{pmatrix}$$

```
C[ -F[2,{i}], F[1,{j}], S[3] ]
== { {-I EL Mass[F[2,{i}]]/(Sqrt[2] SW MW) IndexDelta[i, j]},
     {0} }
```

Current Status of Model Files

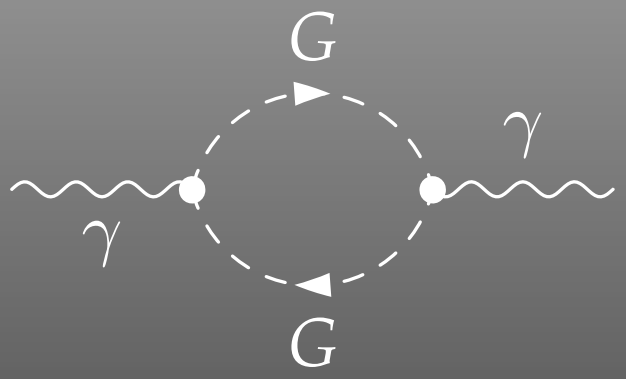
Model Files presently available for *FeynArts*:

- SM [w/QCD], normal and background-field version.
  All one-loop counter terms included.
- MSSM [w/QCD].
  Counter terms not included yet.
- Two-Higgs-Doublet Model.
  Counter terms not included yet.

☺ New ModelMaker utility generates Model Files from the Lagrangian.

☺ Implementation of the MSSM Lagrangian in terms of superfields under way.

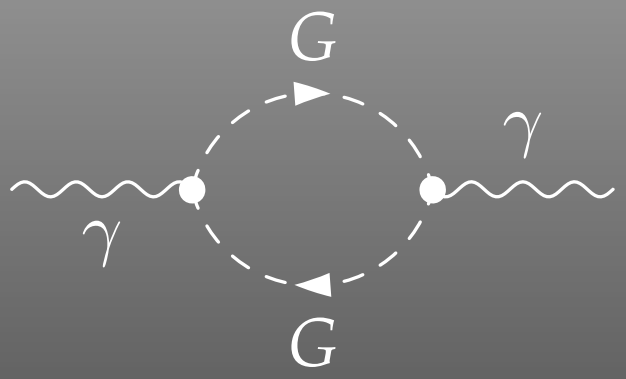
Sample CreateFeynAmp output



= FeynAmp[*identifier* ,
loop momenta ,
generic amplitude ,
insertions]

```
GraphID[Topology == 1, Generic == 1]
```

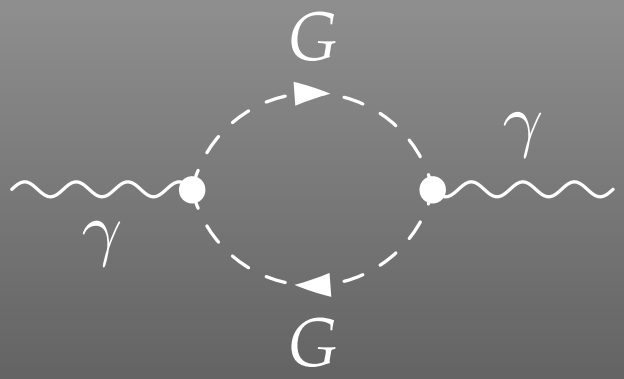
Sample CreateFeynAmp output



= FeynAmp[*identifier* ,
loop momenta ,
generic amplitude ,
insertions]

```
Integral[q1]
```

Sample CreateFeynAmp output



= FeynAmp[*identifier*, *loop momenta*, *generic amplitude*, *insertions*]

$\frac{\mathrm{I}}{32\ \mathrm{Pi}^4}$ RelativeCF ..... *prefactor*

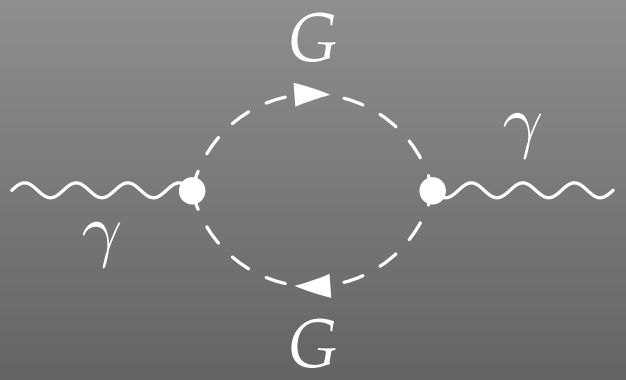
FeynAmpDenominator[$\frac{1}{\mathrm{q1}^2 - \mathrm{Mass[S[Gen3]]}^2}$, $\frac{1}{(-\mathrm{p1} + \mathrm{q1})^2 - \mathrm{Mass[S[Gen4]]}^2}$] ..... *loop denominators*

(p1 - 2 q1)[Lor1] (-p1 + 2 q1)[Lor2] ..... *kin. coupling structure*

ep[V[1], p1, Lor1] ep*[V[1], k1, Lor2] ..... *polarization vectors*

$\mathrm{G}^{(0)}_{\mathrm{SSV}}$[(Mom[1] - Mom[2])[KI1[3]]]
$\mathrm{G}^{(0)}_{\mathrm{SSV}}$[(Mom[1] - Mom[2])[KI1[3]]], ... *coupling constants*

Sample CreateFeynAmp output

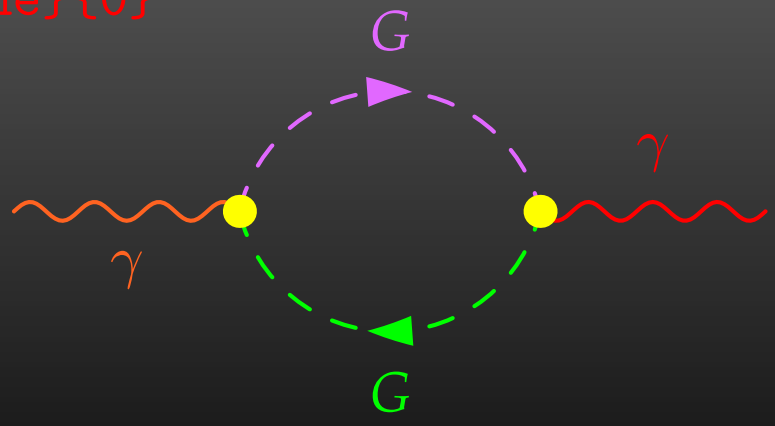


= FeynAmp[*identifier*,
loop momenta,
generic amplitude,
insertions]

```
{ Mass[S[Gen3]],
  Mass[S[Gen4]],
  G_SSV^(0)[(Mom[1] - Mom[2])[KI1[3]]],
  G_SSV^(0)[(Mom[1] - Mom[2])[KI1[3]]],
  RelativeCF } ->
Insertions[Classes][{MW, MW, I EL, -I EL, 2}]
```

Sample Paint output

```
\begin{feynartspicture}(150,150)(1,1)
\FADiagram{}
\FAProp(6.,10.)(14.,10.)(0.8,){/ScalarDash}{-1}
\FALabel(10.,5.73)[t]{$G$}
\FAProp(6.,10.)(14.,10.)(-0.8,){/ScalarDash}{1}
\FALabel(10.,14.27)[b]{$G$}
\FAProp(0.,10.)(6.,10.)(0.,){/Sine}{0}
\FALabel(3.,8.93)[t]{$\gamma$}
\FAProp(20.,10.)(14.,10.)(0.,){/Sine}{0}
\FALabel(17.,11.07)[b]{$\gamma$}
\FAVert(6.,10.){0}
\FAVert(14.,10.){0}
\end{feynartspicture}
```



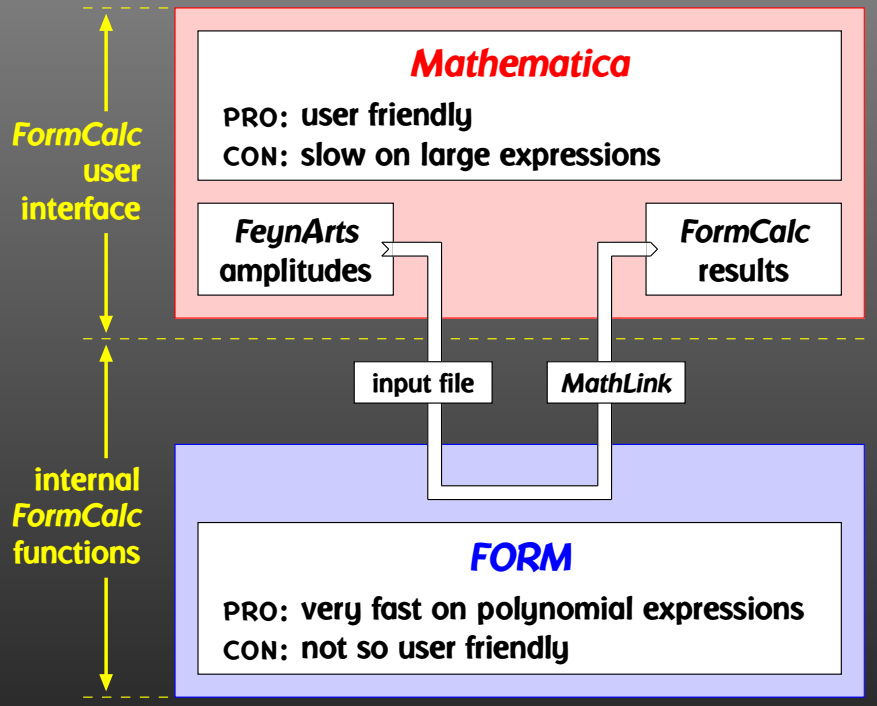
Algebraic Simplification

The amplitudes so far are in no good shape for direct numerical evaluation.

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),
- simplify, and later on compute the square of, open fermion chains and colour structures,
- "compactify" the results as much as possible.

FormCalc



EXAMPLE: Calculating the photon self-energy

```
In[1]:= << FormCalc`

FormCalc 4
by Thomas Hahn
last revised 8 May 03

In[2]:= CalcFeynAmp[<< PhotonSelfEnergy.amp]

preparing FORM code in /tmp/m1.frm
> 2 amplitudes with insertions
> 5 amplitudes without insertions
running FORM...  ok

Out[2]= Amp[{0} -> {0}][(-3 Alfa Pair1 A0[MW2])/(2 Pi) +
   (3 Alfa Pair1 B00[0, MW2, MW2])/Pi +
   ((Alfa Pair1 A0[MLE2[Gen1]])/Pi +
    (Alfa Pair1 A0[MQD2[Gen1]])/(3 Pi) +
    (4 Alfa Pair1 A0[MQU2[Gen1]])/(3 Pi) -
    (2 Alfa Pair1 B00[0, MLE2[Gen1], MLE2[Gen1]])/Pi -
    (2 Alfa Pair1 B00[0, MQD2[Gen1], MQD2[Gen1]])/(3 Pi) -
    (8 Alfa Pair1 B00[0, MQU2[Gen1], MQU2[Gen1]])/(3 Pi)) *
    SumOver[Gen1, 3]]
```

FormCalc Output

A typical term in the output looks like

```
C0i[cc12, MW2, MW2, S, MW2, MZ2, MW2] *
  ( -4 Alfa2 MW2 CW2/SW2 S AbbSum16 +
    32 Alfa2 CW2/SW2 S^2 AbbSum28 +
    4 Alfa2 CW2/SW2 S^2 AbbSum30 -
    8 Alfa2 CW2/SW2 S^2 AbbSum7 +
    Alfa2 CW2/SW2 S(T-U) Abb1 +
    8 Alfa2 CW2/SW2 S(T-U) AbbSum29 )
```

= loop integral

= kinematical variables

= constants

= automatically introduced abbreviations

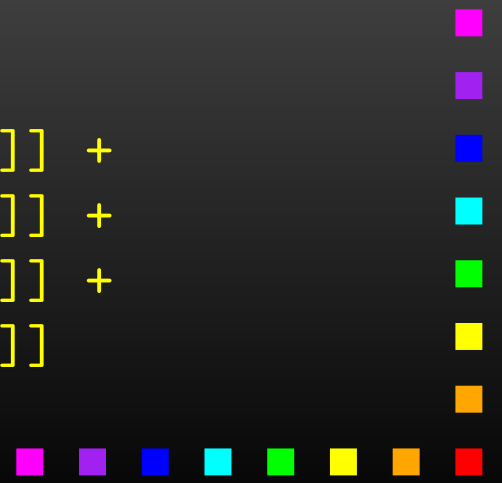
Abbreviations

Outright factorization is usually out of question.
Abbreviations are necessary to reduce size of expressions.

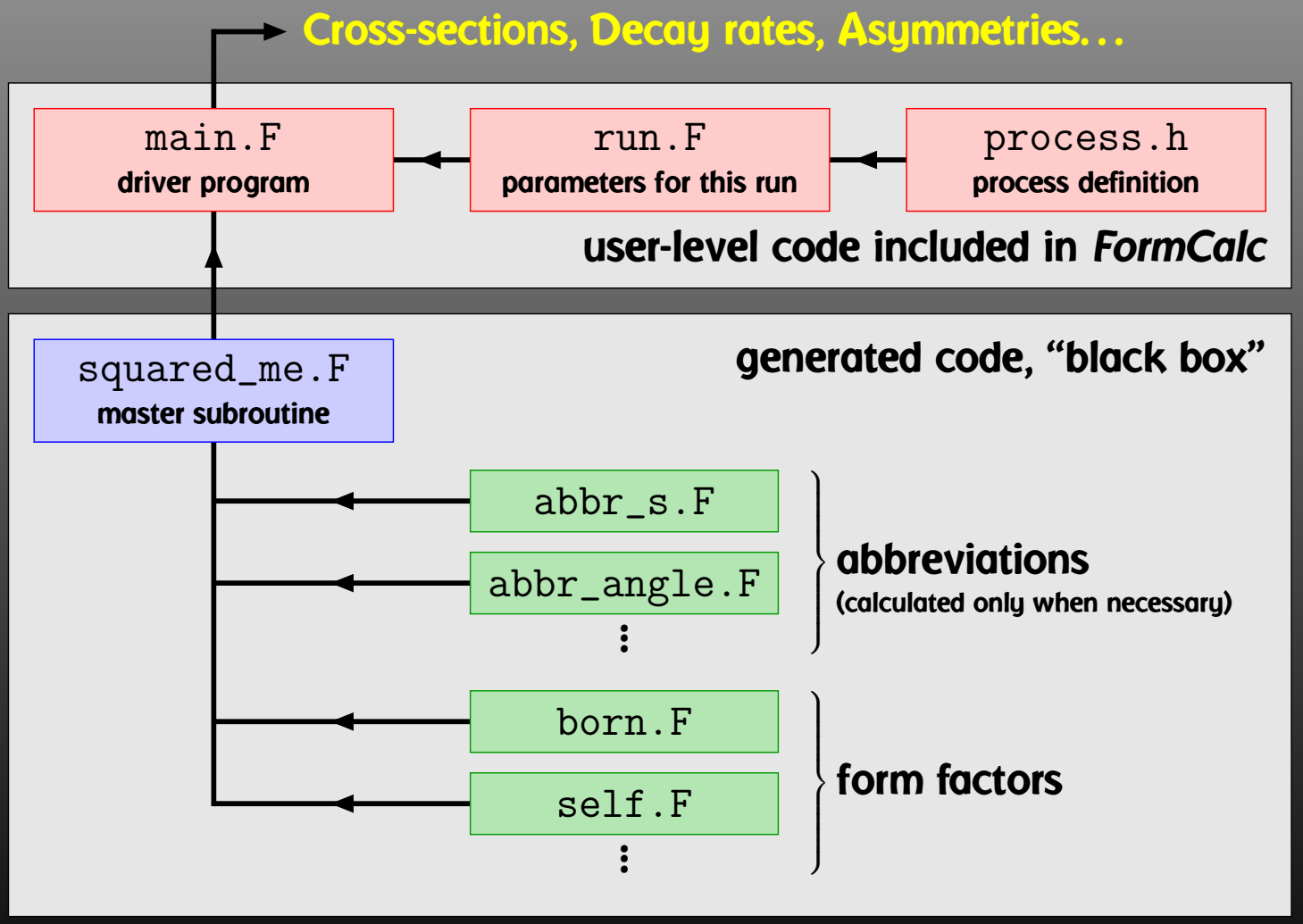
```
AbbSum29 = Abb2 + Abb22 + Abb23 + Abb3

Abb22 = Pair1 Pair3 Pair6

Pair3 = Pair[e[3], k[1]]
```

The full expression corresponding to `AbbSum29` is

```
Pair[e[1], e[2]] Pair[e[3], k[1]] Pair[e[4], k[1]] +
Pair[e[1], e[2]] Pair[e[3], k[2]] Pair[e[4], k[1]] +
Pair[e[1], e[2]] Pair[e[3], k[1]] Pair[e[4], k[2]] +
Pair[e[1], e[2]] Pair[e[3], k[2]] Pair[e[4], k[2]]
```



Numerical Evaluation in Fortran 77



Drivers currently available for $1 \longrightarrow 2$, $2 \longrightarrow 2$, $2 \longrightarrow 3$.

MSSM Parameters

Input parameters:

TB, MA0, At, Ab, Atau, MSusy, M_2, MUE

↓

mssm_ini.F

↓

All parameters appearing in the Model File:

Mh0, MHH, MA0, MHp, CB, SB, TB, CA, SA,
C2A, S2A, C2B, S2B, CAB, SAB, CBA, SBA,
MUE, MGl, MNeu[n], ZNeu[n,n'],
MCha[c], UCha[c,c'], VCha[c,c'],
MSf[s,t,g], USf[t,g][s,s'], Af[t,g]

Details in TH, C. Schappacher, hep-ph/0105349.

Parameter Scans

With the preprocessor definitions in `run.F` one can either

- assign a parameter a fixed value, as in

```
#define LOOP1 TB = 1.5D0
```

- declare a loop over a parameter, as in

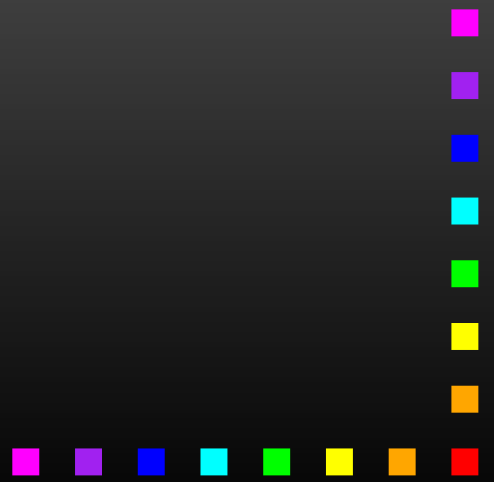
```
#define LOOP1 do 1 TB = 2,30,5
```

which computes the cross-section for `TB` values of 2 to 30 in steps of 5.

Main Program:

```
      LOOP1
      LOOP2
        ⋮
      (calculate
      cross-section)
1     continue
```

Scans are perfect for parallelization:
- SMP Support already available,
- Parallelization over a Network coming soon.



Conclusions

- Serious perturbative calculations these days can generally no longer be done by hand:
  - Required accuracy, Models with many particles, ...
- Hybrid programming techniques are necessary:
  - Computer algebra is an indispensable tool because many manipulations must be done symbolically.
  - Fast number crunching can only be achieved in a compiled language.
- Software engineering and further development of the existing packages is a must:
  - As we move on to ever more complex computations (more loops, more legs), the computer programs must become more "intelligent," i.e. must learn all possible tricks to still be able to handle the expressions.