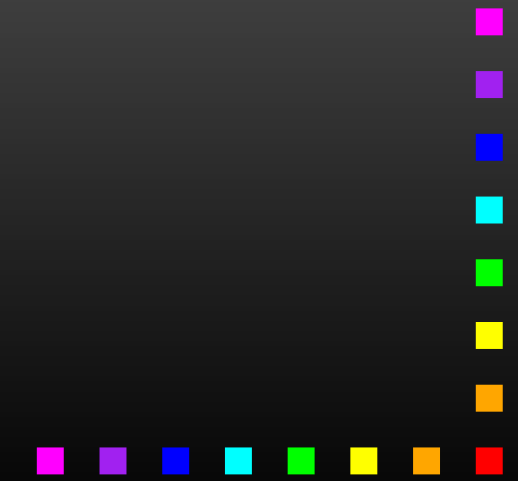
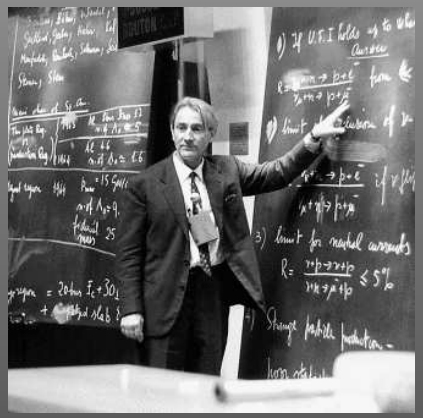
Automating Feynman-diagrammatic calculations

Thomas Hahn

Max-Planck-Institut für Physik



What those "Loop Calculations" are all about

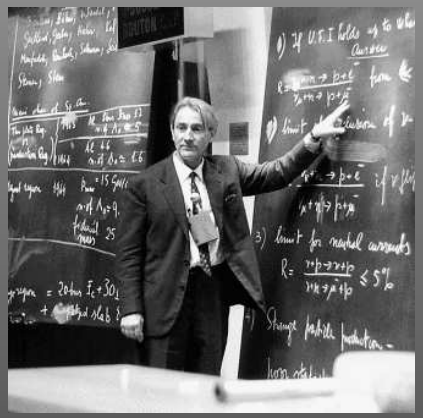


Theorists like

$\mathcal{L}$

Why: The Lagrangian defines the Theory.

What those "Loop Calculations" are all about



Theorists like

$\mathcal{L}$

Why: The Lagrangian defines the Theory.

Experimentalists like

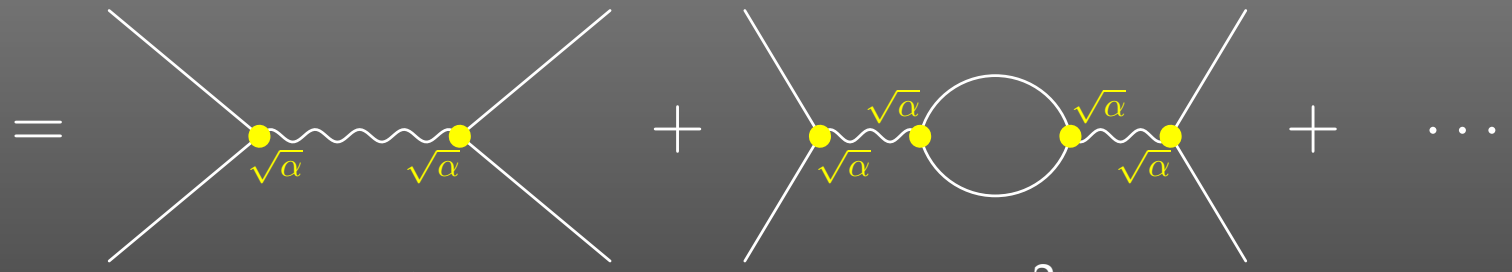
S

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

How to connect $\mathcal{L}$ and S?

$S = \sum$ (Feynman Diagrams)

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



$\mathcal{O}(\alpha)$ $\mathcal{O}(\alpha^2)$

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

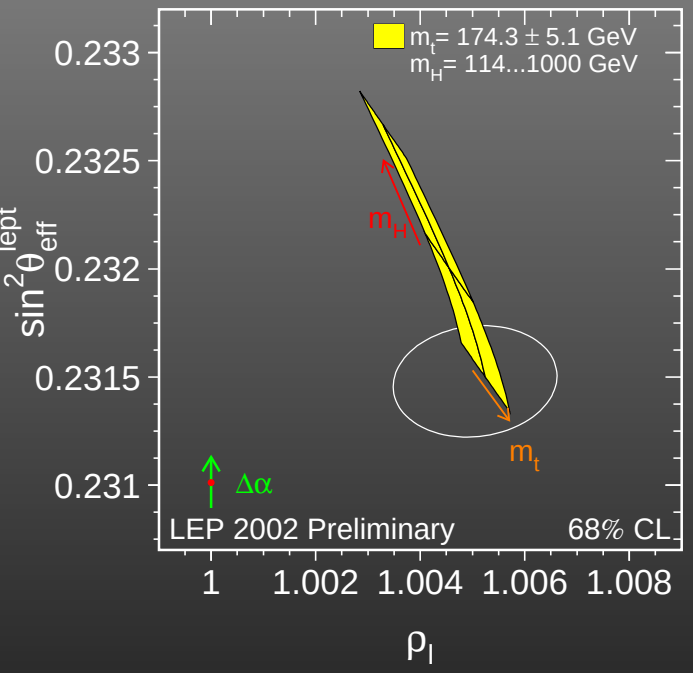
Accuracy of results $\longleftrightarrow$ Order in α $\longleftrightarrow$ # of Loops

$\Updownarrow$

Quantum effects $\longleftrightarrow$ Order in $\hbar$ $\longleftrightarrow$ # of Loops

Why Radiative Corrections?

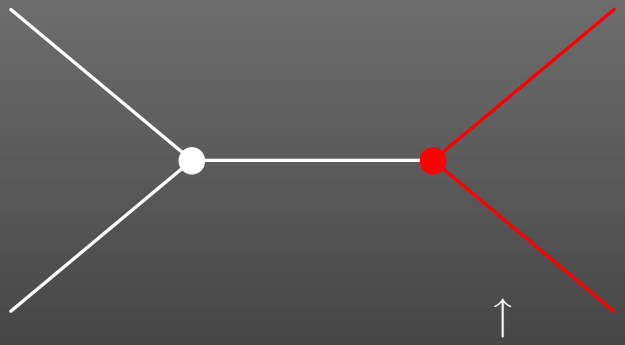
Precision: Radiative Corrections are seen experimentally



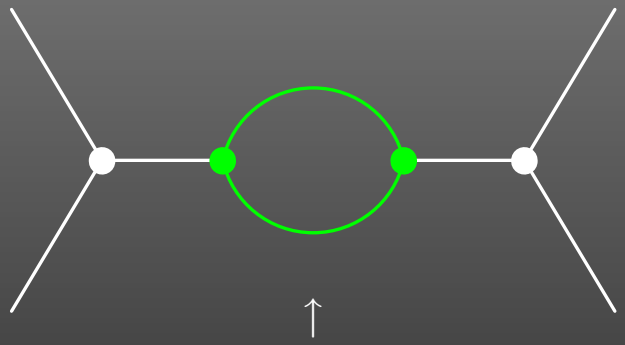
$10^{10} a_\mu =$ 11614097.29	QED	1-loop
41321.76		2-loop
3014.19		3-loop
36.70		4-loop
.63		5-loop
690.6	Had.	
19.5	EW	1-loop
−4.3		2-loop
11659176	theory, total	
11659204	exp (BNL 2002)	

Why Radiative Corrections?

Indirect effects of particles beyond the kinematical limit



inaccessible
(too heavy to be produced)

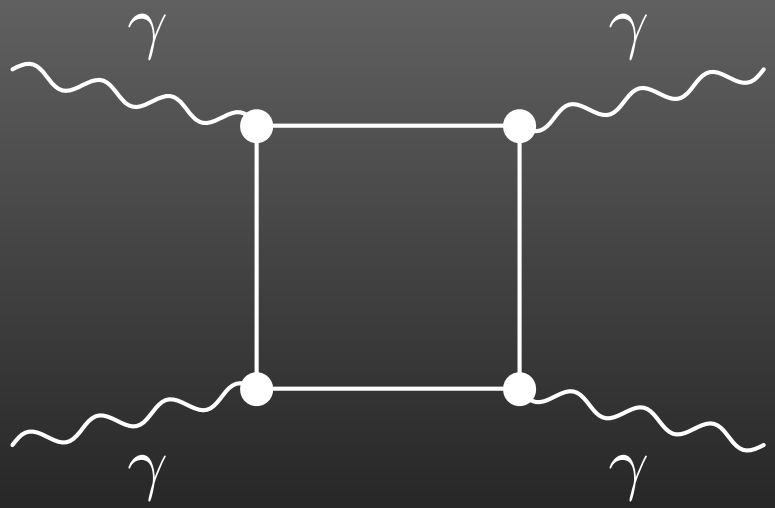


indirectly visible,
requires precision measurements

Why Radiative Corrections?

"Rare" (loop-mediated) events

e.g. light-by-light scattering:

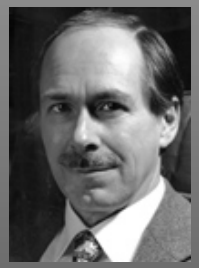
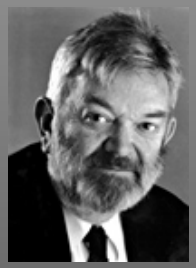


State of the Art of Perturbative Calculations

# loops	0	1	2	3+
# $2 \to 2$ topologies	4	99	2214	50051
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 ($2 \to 2$) may soon be feasible thanks to new technology (Remiddi, Gehrmann).
- Only few results, approximations only (large-mass, large-momentum expansion).

Recipe for Feynman Diagrams

Thanks to  and  (and others) we have a

Recipe for an ARBITRARY Feynman diagram up to one loop

①	Draw all possible types of diagrams	topological task
②	Figure out what particles can run on each type of diagram	combinatorical task
③	Translate the diagrams into formulas by applying the Feynman rules	database look-up
④	Contract the indices, take the traces, etc.	algebraic simpl.
⑤	Write up the results as a computer program	programming
⑥	Run the program to get numerical results	waiting

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.

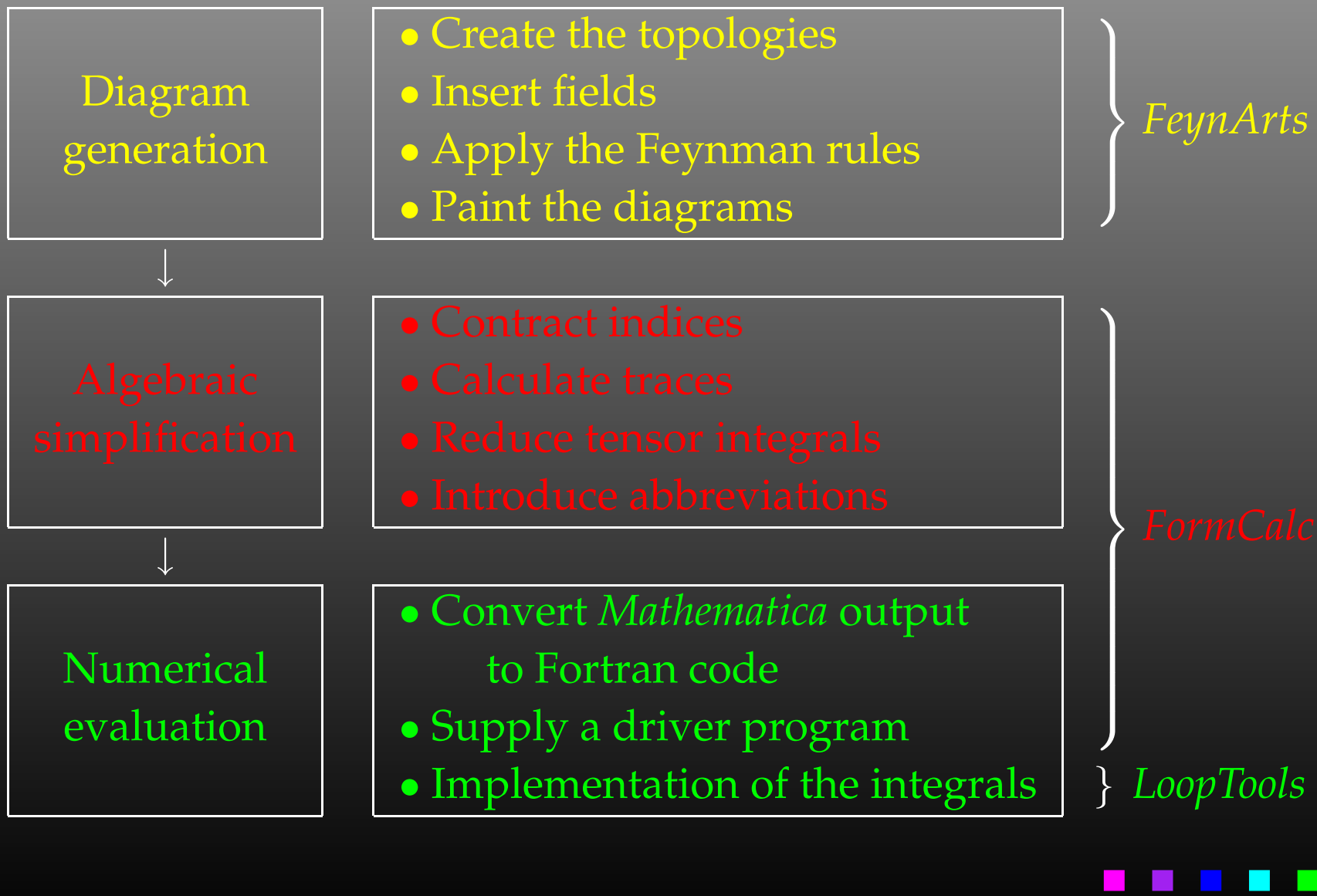
Compiled high-level language (e.g. Fortran) for the numerical evaluation.

Packages

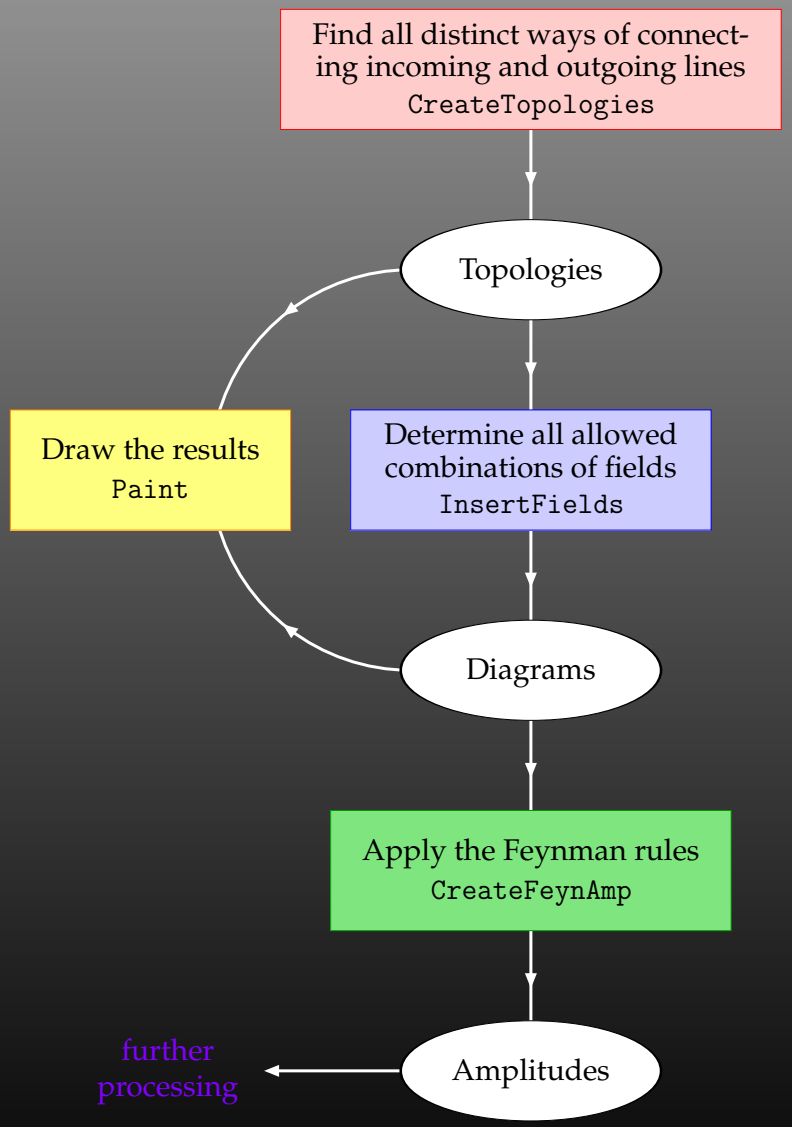
Comprehensive packages for Perturbative Calculations
– Tree-level calculations only
– One-Loop calculations only
– "Arbitrary" loop order

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

The packages *FeynArts*, *FormCalc*, and *LoopTools*



FeynArts



EXAMPLE: generating the Higgs self-energy

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

one loop

one incoming particle

one outgoing particle

```
Paint[top]
```

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

use the Standard Model

the name of the Higgs boson in the "SM" model file

```
Paint[ins]
```

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

```
amp >> HiggsSelfEnergy.amp
```

Three Levels of Fields

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

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

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]`

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

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

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

```
AnalyticalCoupling[
  s1 F[j1, mom1], s2 F[j2, mom2], s3 S[j3, mom3] ] ==
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

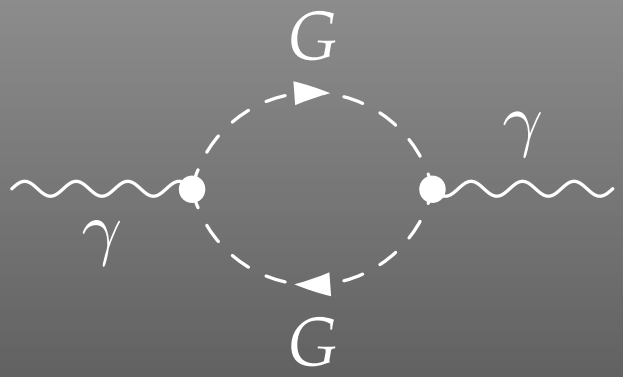
- Classes Model File (for each model)
  declaring the particles and the allowed couplings

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

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

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

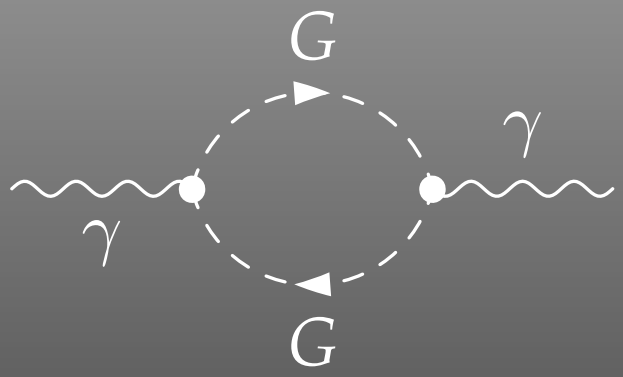
Sample `CreateFeynAmp` output



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

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

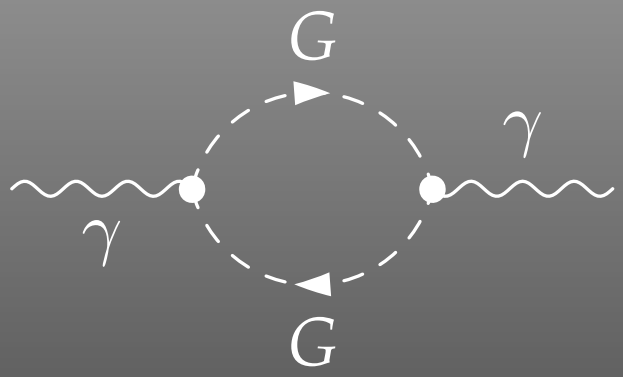
Sample `CreateFeynAmp` output



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

`Integral[q1]`

Sample CreateFeynAmp output



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

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

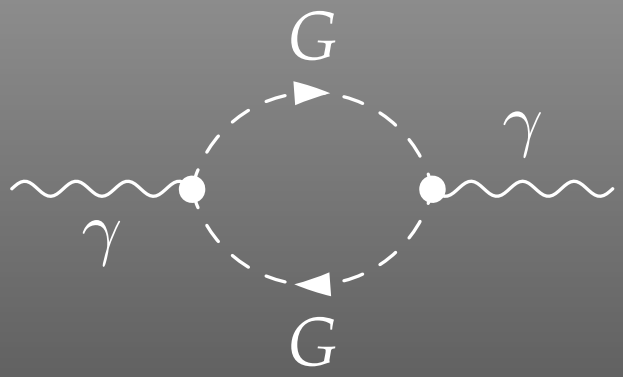
FeynAmpDenominator[$\frac{1}{\text{q1}^2 - \text{Mass[S[Gen3]]}^2}$, $\frac{1}{(-\text{p1} + \text{q1})^2 - \text{Mass[S[Gen4]]}^2}$] ..... *loop denominators*

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

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

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

Sample CreateFeynAmp output

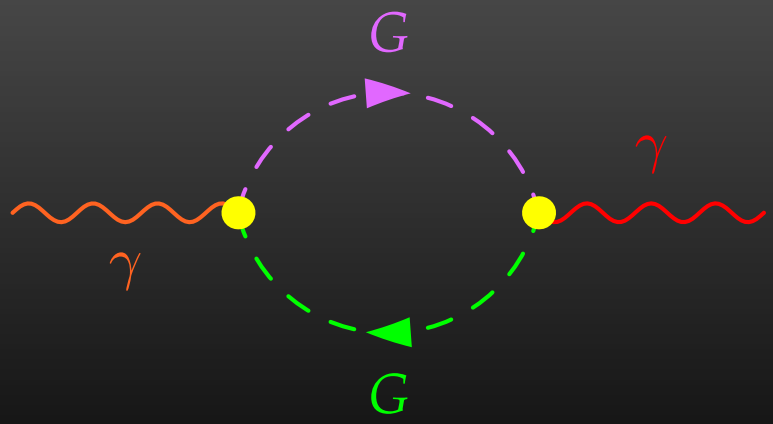


= FeynAmp[*identifier*,
loop momenta,
generic amplitude,
insertions]

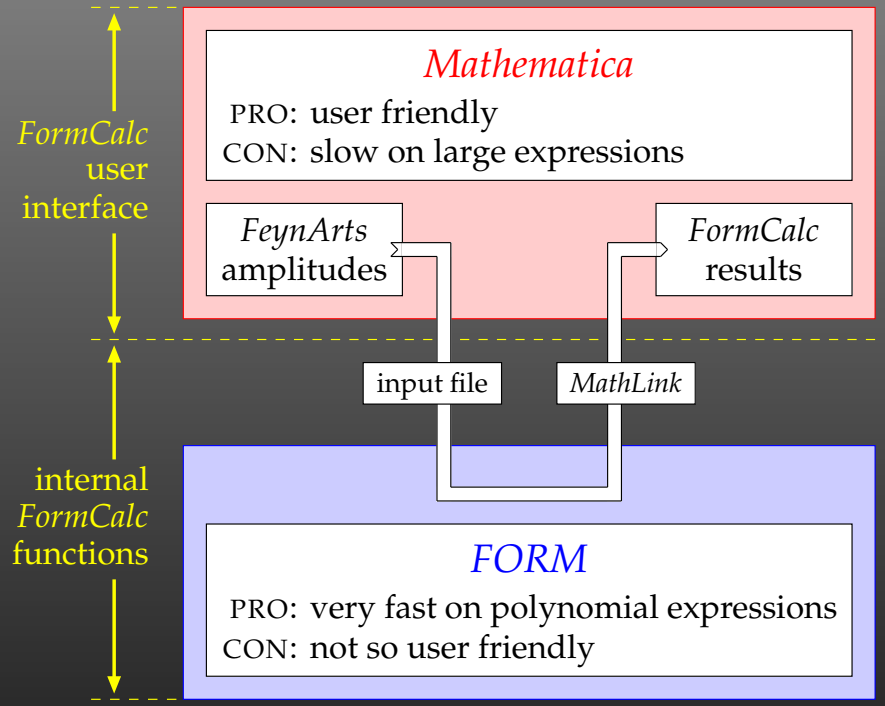
```
{ Mass[S[Gen3]],
  Mass[S[Gen4]],
  G_SSV^(0)[(Mom[1] - Mom[2])[KI1[3]]],
  G_SSV^(0)[(Mom[1] - Mom[2])[KI1[3]]],
  RelativeCF } ->
Insertions[Classes][{MW, MW, I EL, -I EL, 2}]
```

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}
```



FormCalc



EXAMPLE: Calculating the Higgs self-energy

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

FormCalc 3.2
by Thomas Hahn
last revised 14 Jan 03

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

preparing FORM code in /tmp/m1.frm
> 11 amplitudes with insertions
> 0 amplitudes without insertions
running FORM...  ok

                                                  -3 Alfa Pair1 A0[MW2]
Out[2]= Amp[{MH} -> {MH}][------------------------------------------------ +
                                                        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 CW2 MW2/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

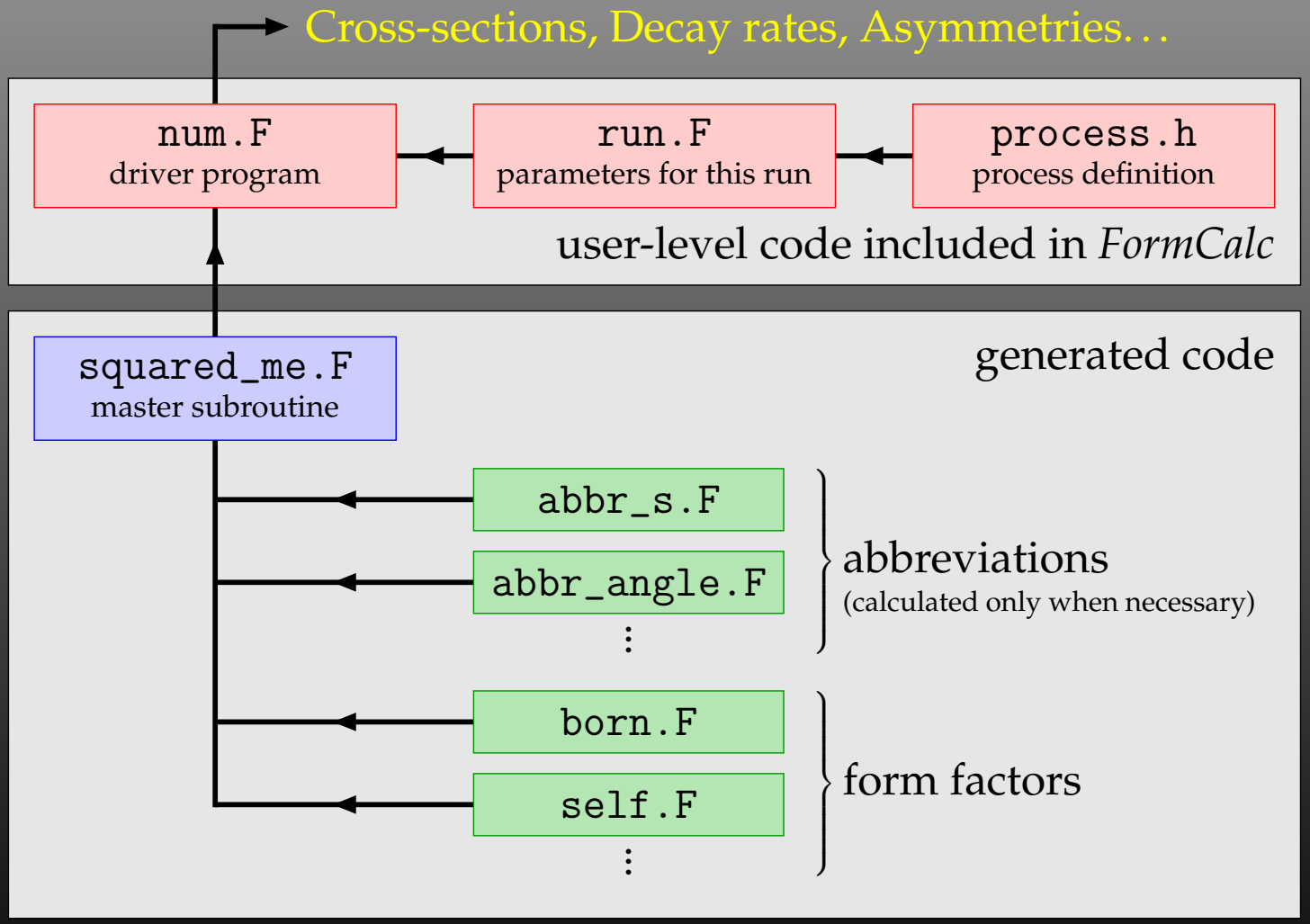
- f77 has a proven track record for fast and reliable numerics ("number crunching"),
- good compilers are available for all platforms,
- possibility to link with existing code.

Translation *Mathematica* $\rightarrow$ Fortran necessary

Issues of code generation:

- self-sufficiency: user should not need to modify the generated code,
  - ▷ anxillary files have to be generated, too (e.g. makefile),
  - ▷ encapsulation (well-defined interface, no loose ends),
- optimization, speed.

Structure of Generated Code



Summary of Features

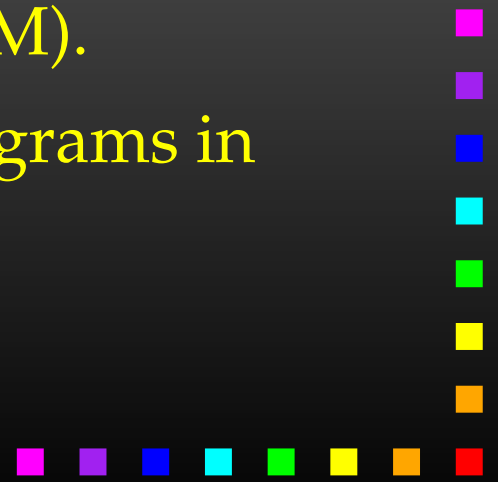
The three programs

FeynArts, *FormCalc*, and *LoopTools*

can be used for one-loop calculations and they work together smoothly. Their main features are:

FeynArts

- Three-level field structure.
- Model files can be supplied/modified by the user (Currently included: QED, SM, QCD, MSSM).
- Produces publication-quality Feynman diagrams in PostScript, LaTeX, and other formats.
- `http://www.feynarts.de`



Summary of Features

The three programs

FeynArts, *FormCalc*, and *LoopTools*

can be used for one-loop calculations and they work together smoothly. Their main features are:

FormCalc

- Fast, because of *FORM*.
- Analytical calculation as far as possible.
- Includes the translation into Fortran code and the Fortran driver programs.
- `http://www.feynarts.de/formcalc`

Summary of Features

The three programs

FeynArts, *FormCalc*, and *LoopTools*

can be used for one-loop calculations and they work together smoothly. Their main features are:

LoopTools

- Stable, easy-to-use implementation of the one-loop scalar and tensor integrals in Fortran, C++, and *Mathematica*.
- `http://www.feynarts.de/looptools`