

Automatic One-Loop Calculations with FeynArts and FormCalc

Thomas Hahn

Max-Planck-Institut für Physik
München



What are Feynman Diagrams good for?



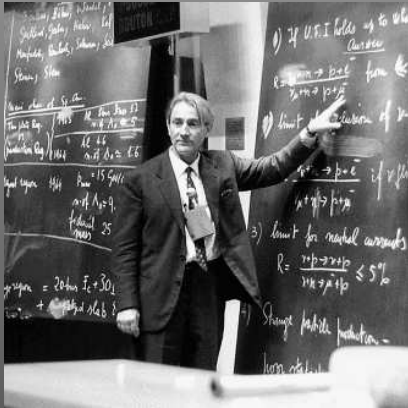
Theorists like

\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

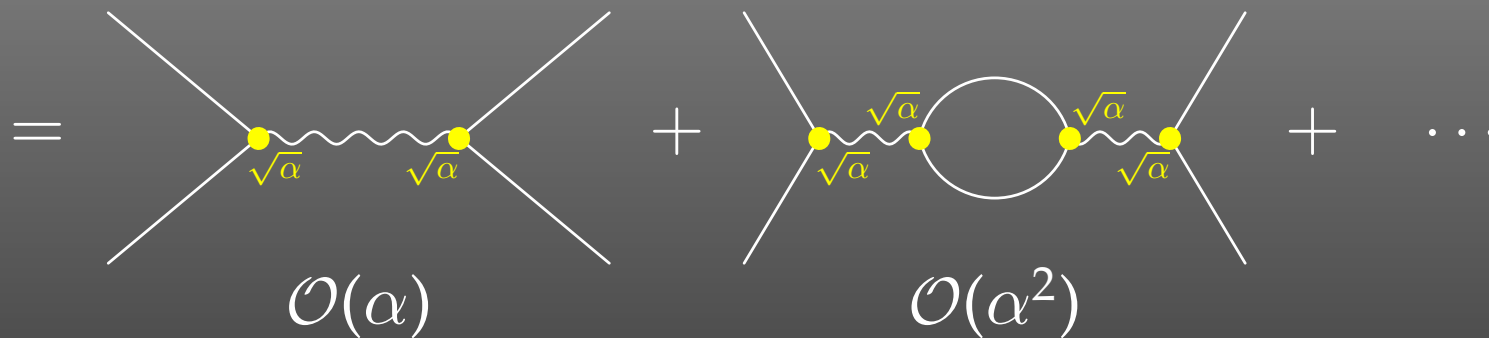
Why: Cross sections, decay rates, etc. follow directly from the S -Matrix.



What are Feynman Diagrams good for?

$S = \sum (\text{Feynman Diagrams})$

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



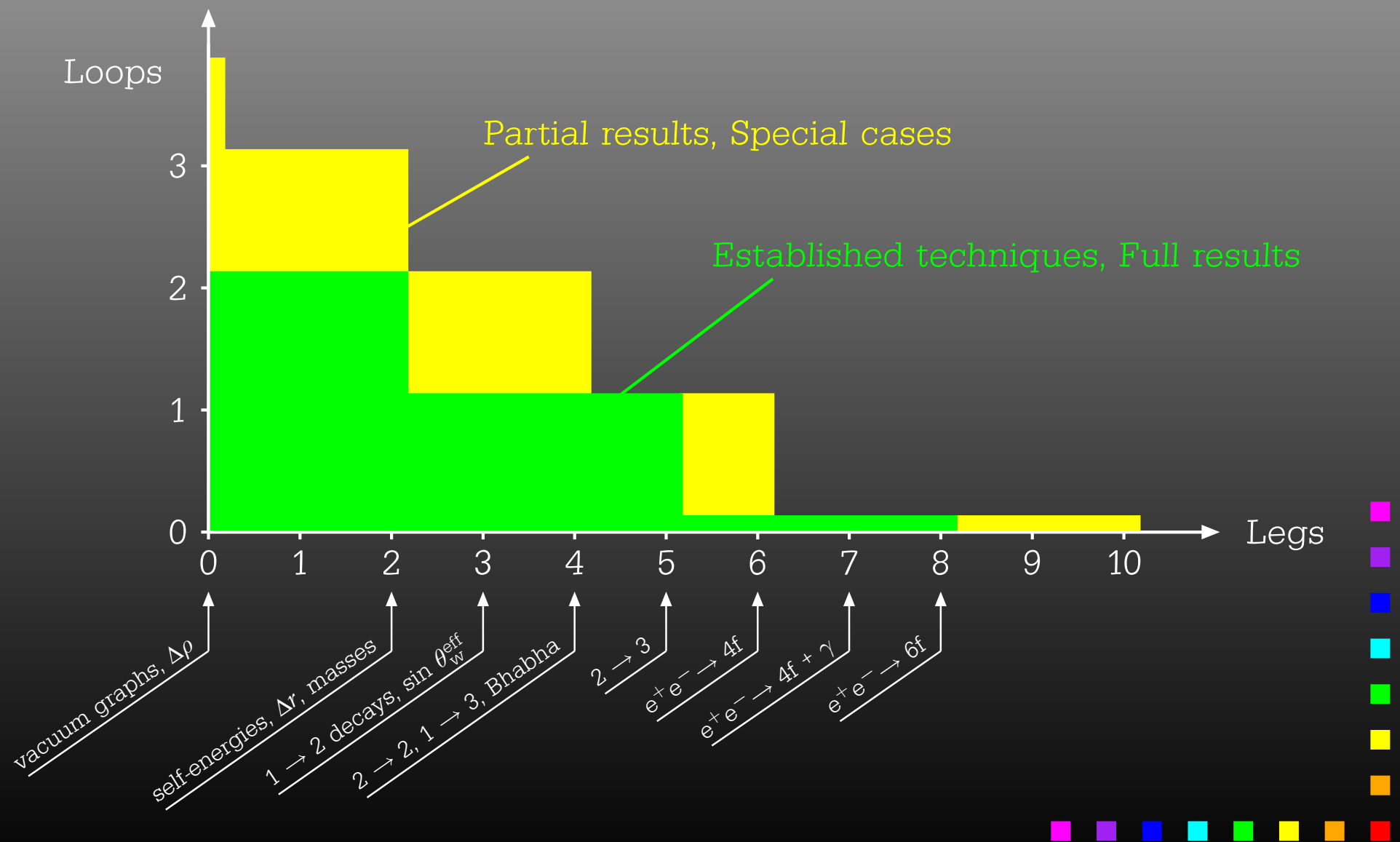
\mathcal{L} determines the Feynman Rules which tell us how to translate the diagrams into formulas.

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

State of the Art

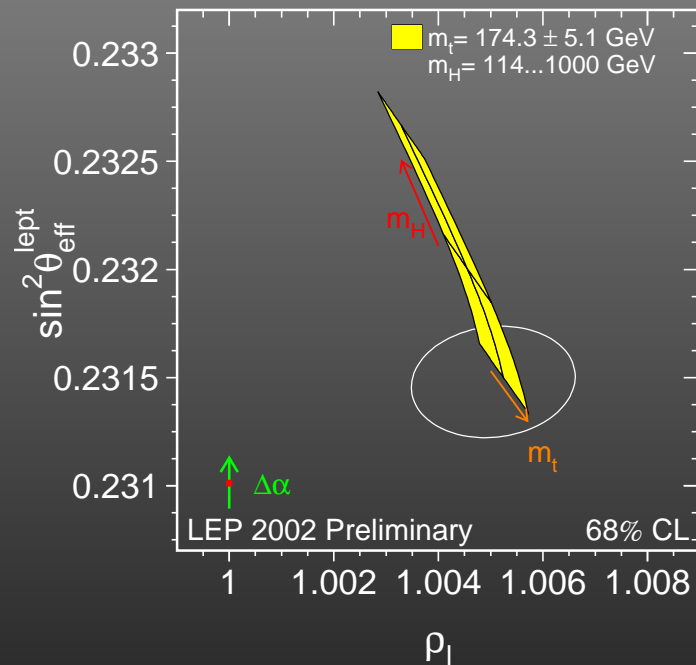
# loops	0	1	2	3+
# $2 \rightarrow 2$ topologies	4	99	2214	50051
typical accuracy	10%	1%	.1%	.01%
general procedure known	yes	yes	$1 \rightarrow 1$	no
limits	$2 \rightarrow 8$	$2 \rightarrow 4$	$1 \rightarrow 2$	$1 \rightarrow 1$

State of the Art



Why Higher Orders?

Precision: Higher Orders are seen experimentally

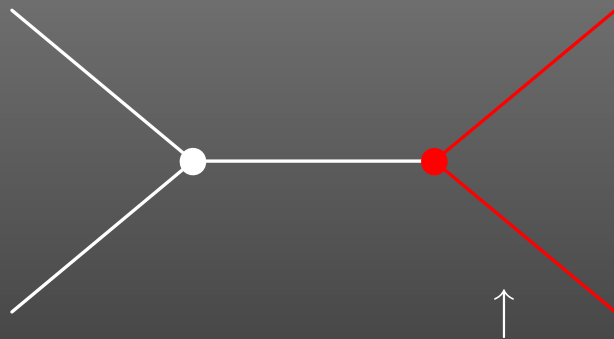


$10^{10} a_\mu = 11614097.29$	QED	1-loop
41321.76		2-loop
3014.19		3-loop
36.70		4-loop
.63		5-loop
690.6	Had.	
19.5	EW	1-loop
-4.3		2-loop
11659176	theory, total	
11659204	exp (BNL 2002)	

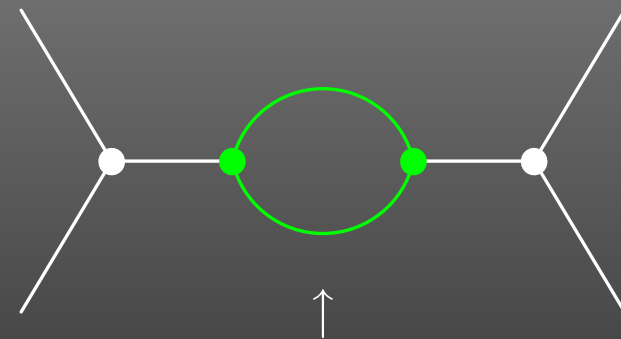


Why Higher Orders?

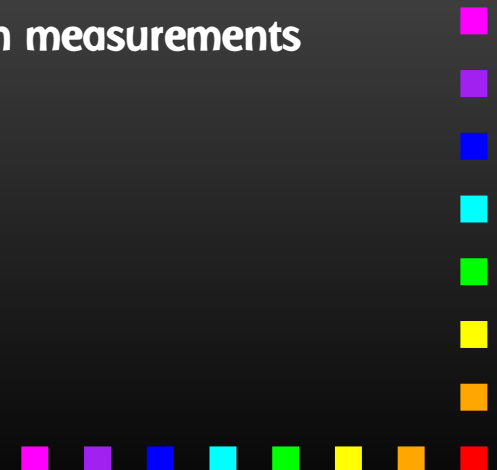
Indirect effects of particles beyond the kinematical limit



inaccessible
(too heavy to be produced)



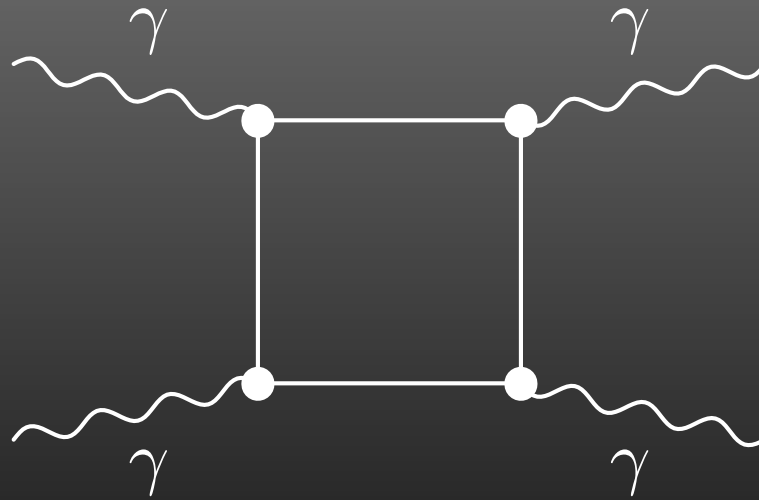
indirectly visible,
requires precision measurements



Why Higher Orders?

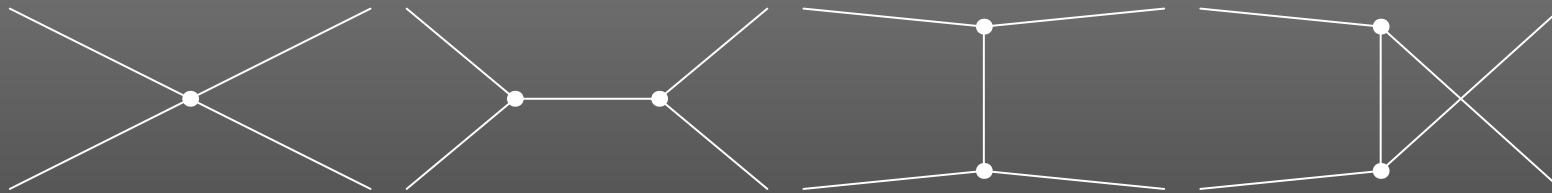
“Rare” (loop-mediated) events

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

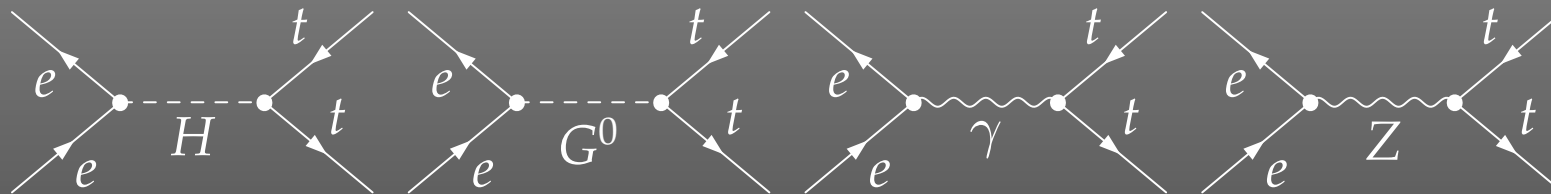


Topological task



How do I calculate Feynman Diagrams

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

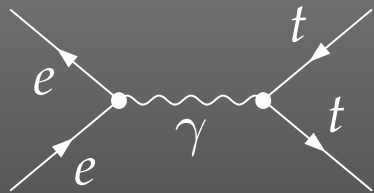


Combinatorial task, requires physics input (model)



How do I calculate Feynman Diagrams

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

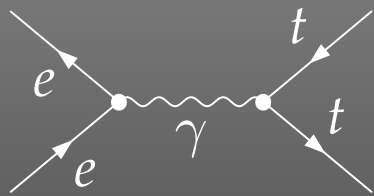


$$= \underbrace{\langle v_1 | ie\gamma^\mu | u_2 \rangle}_{\text{left vertex}} \underbrace{\frac{g_{\mu\nu}}{(k_1 + k_2)^2}}_{\text{propagator}} \underbrace{\langle u_4 | (-\frac{2}{3}ie\gamma^\nu) | v_3 \rangle}_{\text{right vertex}}$$

Database look-up

How do I calculate Feynman Diagrams

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



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

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

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

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

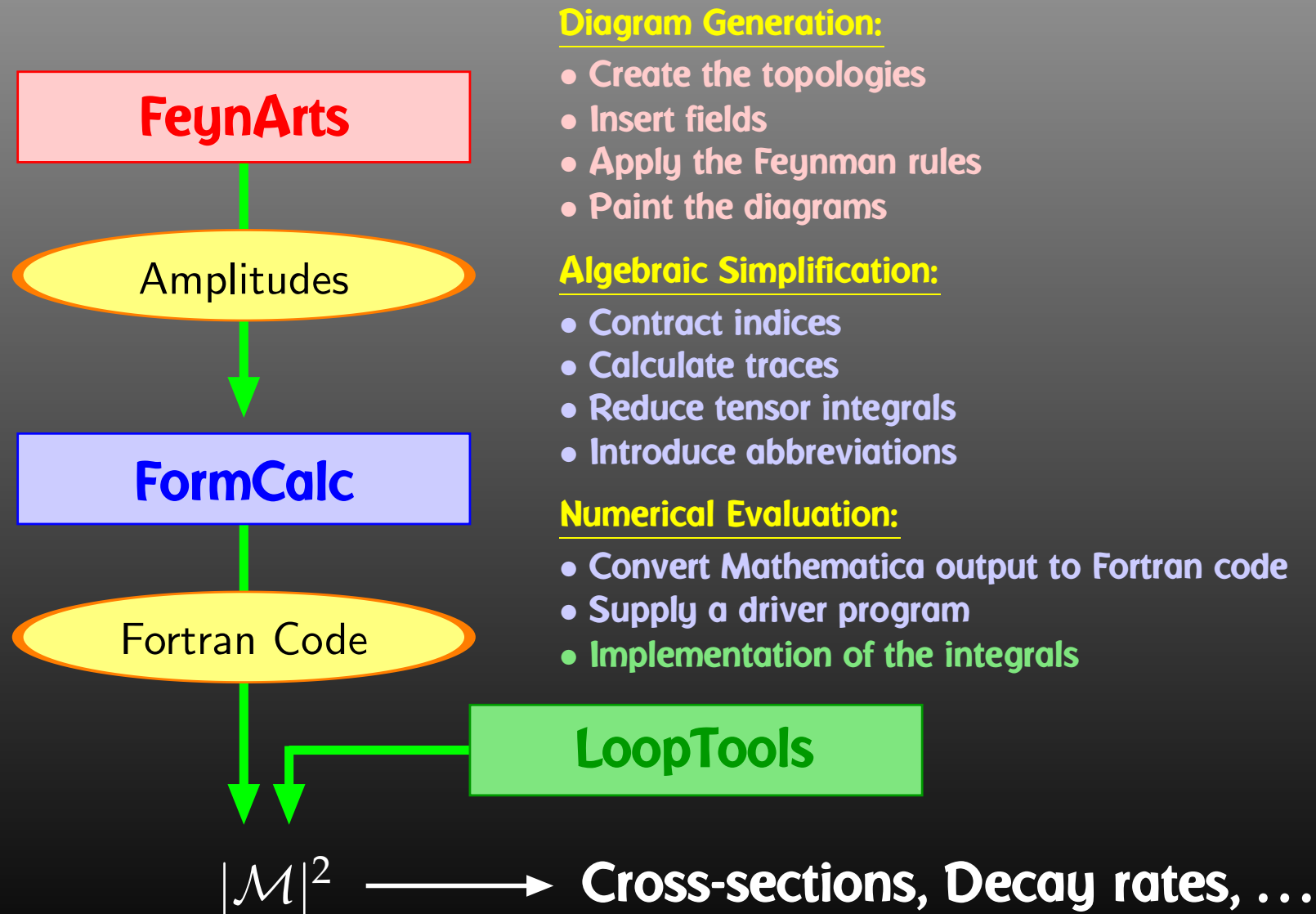
■ = One-Loop calculations only

■ = “Arbitrary” loop order

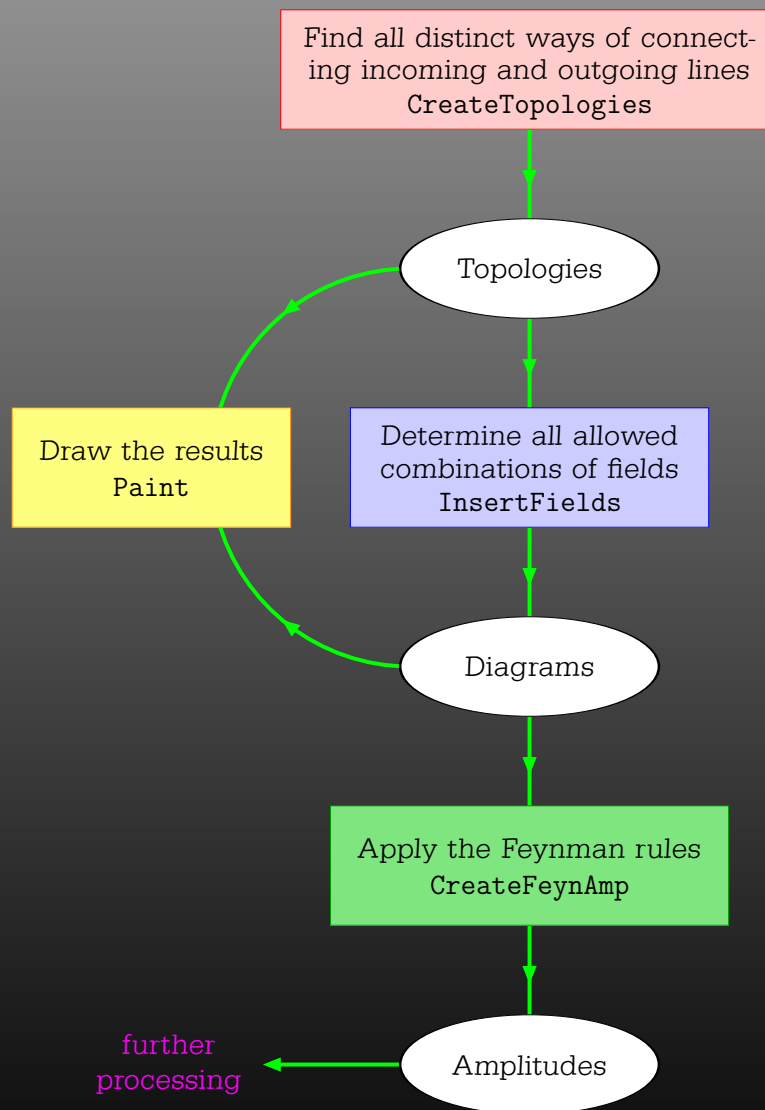
<i>Program Name</i>	FeynArts	GRACE	CalcHEP	DIANA	MadGraph
<i>Group</i>	WÜ/KA/M	KEK	Dubna	Bielefeld	Madison
<i>Core language</i>	Mathematica	C	C, Fortran	C	Fortran
Components:					
<i>Diagram generation</i>	FeynArts	grc	CalcHEP	QGRAF	MadGraph
<i>Diagram painting</i>	FeynArts	gracefig	CalcHEP	DIANA	—
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



How does the *computer* calculate Feynman Diagrams



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. $-F[2]$, $F[1]$, $S[3]$

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

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

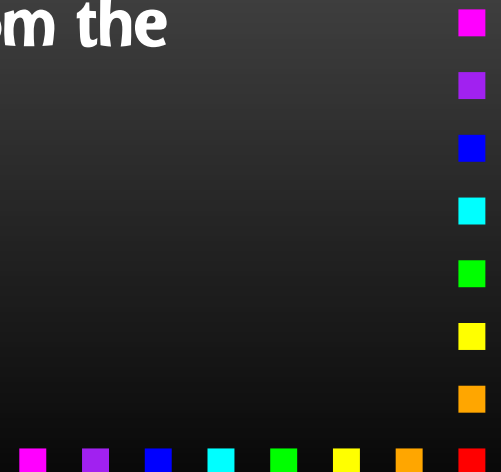
$$\vec{G}(\bar{\ell}_i, \nu_j, G) = \begin{pmatrix} G_- \\ G_+ \end{pmatrix} = \begin{pmatrix} -\frac{i e m_{\ell,i}}{\sqrt{2} \sin \theta_w M_W} \delta_{ij} \\ 0 \end{pmatrix}$$

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

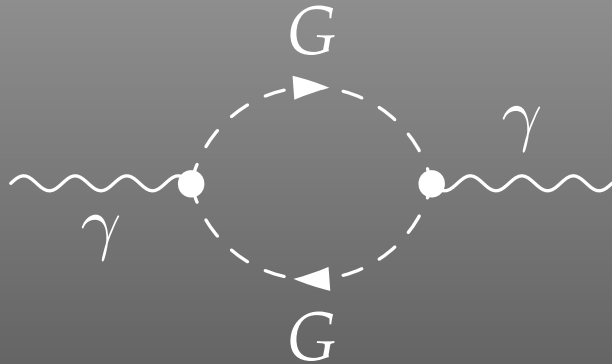
Current Status of Model Files

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 by T. Fritzsche.
- **Two-Higgs-Doublet Model.**
Counter terms not included yet.
- **ModelMaker utility generates Model Files from the Lagrangian.**



Sample CreateFeynAmp output



= FeynAmp[

identifier,

loop momenta,

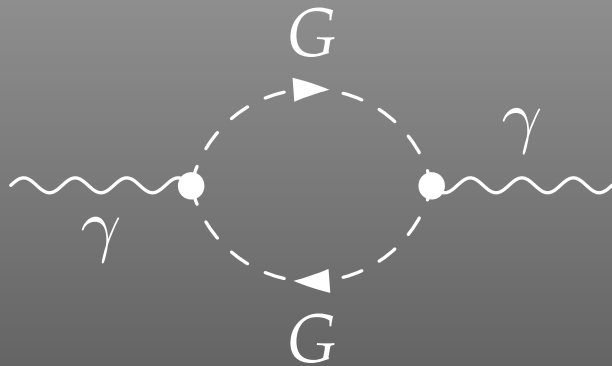
generic amplitude,

insertions]

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



Sample CreateFeynAmp output

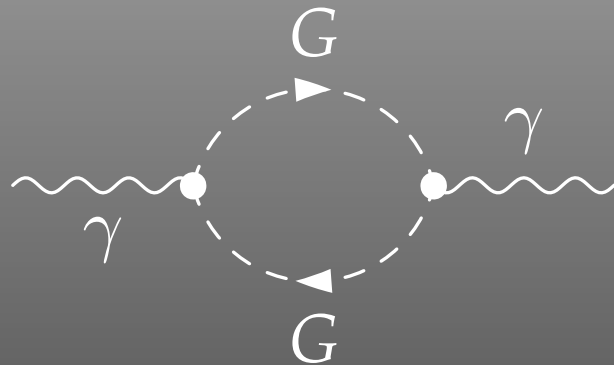


```
= FeynAmp[ identifier,  
            loop momenta,  
            generic amplitude,  
            insertions ]
```

`Integral[q1]`



Sample CreateFeynAmp output



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

$\frac{1}{32 \text{ Pi}^4}$ RelativeCFprefactor

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

$(p1 - 2q1)[\text{Lor1}] (-p1 + 2q1)[\text{Lor2}]$ kin. coupling structure

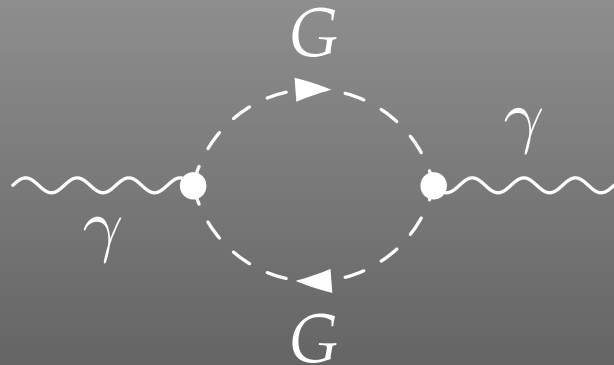
$\text{ep}[\text{V}[1], p1, \text{Lor1}] \text{ep}^*[\text{V}[1], k1, \text{Lor2}]$ polarization vectors

$G_{\text{SSV}}^{(0)}[(\text{Mom}[1] - \text{Mom}[2])[\text{KI1}[3]]]$

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



Sample CreateFeynAmp output



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

insertions

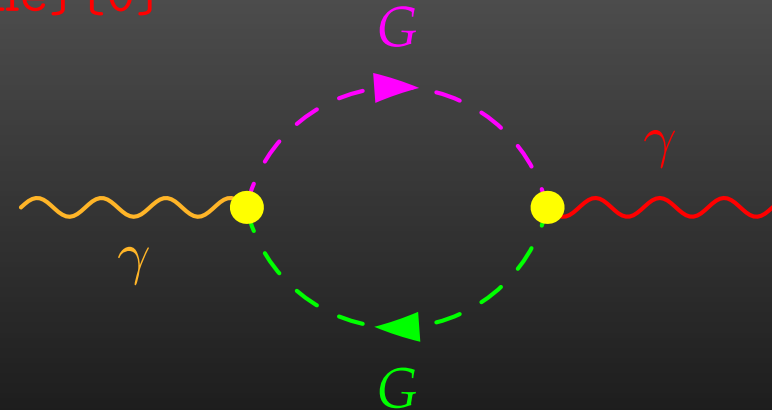
]

```
{ Mass[S[Gen3]],
  Mass[S[Gen4]],
  GSSV(0)[(Mom[1] - Mom[2])[KI1[3]]],
  GSSV(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

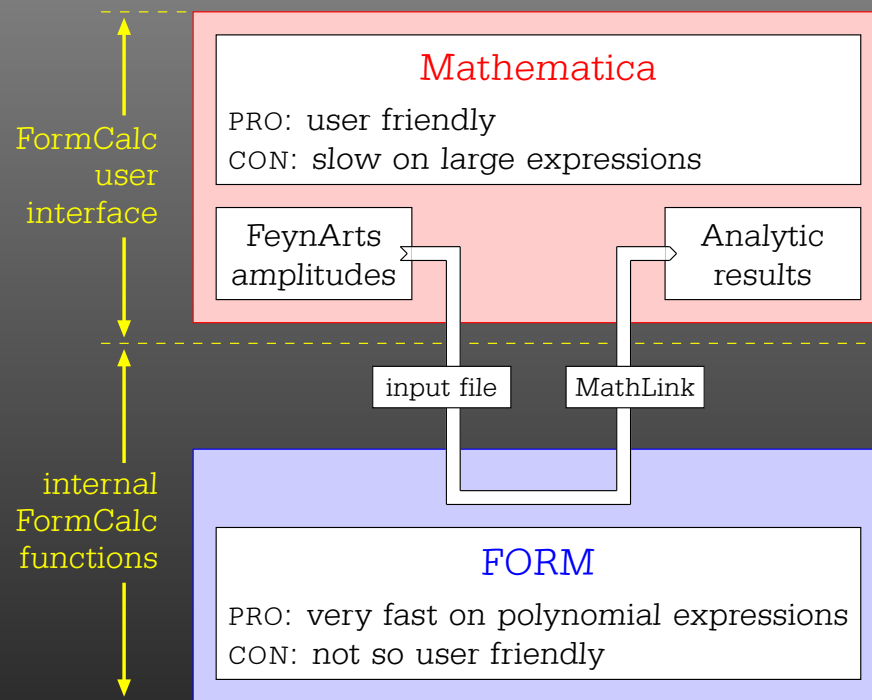
The amplitudes so far are in **no good shape for direct numerical evaluation.**

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 open fermion chains,**
- **simplify and compute the square of $SU(N)$ structures,**
- ▷ **“compactify” the results as much as possible.**



FormCalc



EXAMPLE: Calculating the photon self-energy

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

```
FormCalc 4.1
```

```
by Thomas Hahn
```

```
last revised 1 Mar 05
```

```
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}] [
```

$$\frac{-3 \text{Alfa Pair1 A0[MW2]}}{2 \text{Pi}} +$$

$$\frac{3 \text{Alfa Pair1 B00}[0, \text{MW2}, \text{MW2}]}{\text{Pi}} +$$

$$\left(\frac{\text{Alfa Pair1 A0[MLE2][Gen1]]}{\text{Pi}} + \right.$$

$$\frac{\text{Alfa Pair1 A0[MQD2][Gen1]]}{3 \text{Pi}} +$$

$$\frac{4 \text{Alfa Pair1 A0[MQU2][Gen1]]}{3 \text{Pi}} -$$

$$\frac{2 \text{Alfa Pair1 B00}[0, \text{MLE2}[Gen1], \text{MLE2}[Gen1]]}{\text{Pi}} -$$

$$\frac{2 \text{Alfa Pair1 B00}[0, \text{MQD2}[Gen1], \text{MQD2}[Gen1]]}{3 \text{Pi}} -$$

$$\left. \frac{8 \text{Alfa Pair1 B00}[0, \text{MQU2}[Gen1], \text{MQU2}[Gen1]]}{3 \text{Pi}} \right) *$$

```
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 S2 AbbSum28 +  
 4 Alfa2 CW2/SW2 S2 AbbSum30 -  
 8 Alfa2 CW2/SW2 S2 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.

$$\text{AbbSum29} = \text{Abb2} + \boxed{\text{Abb22}} + \text{Abb23} + \text{Abb3}$$

$$\boxed{\text{Abb22} = \text{Pair1} \boxed{\text{Pair3}} \text{Pair6}}$$

$$\boxed{\text{Pair3} = \text{Pair}[e[3], k[1]]}$$

The full expression corresponding to **AbbSum29** is

$$\begin{aligned} & \text{Pair}[e[1], e[2]] \text{ Pair}[e[3], k[1]] \text{ Pair}[e[4], k[1]] + \\ & \text{Pair}[e[1], e[2]] \text{ Pair}[e[3], k[2]] \text{ Pair}[e[4], k[1]] + \\ & \text{Pair}[e[1], e[2]] \text{ Pair}[e[3], k[1]] \text{ Pair}[e[4], k[2]] + \\ & \text{Pair}[e[1], e[2]] \text{ Pair}[e[3], k[2]] \text{ Pair}[e[4], k[2]] \end{aligned}$$

External Fermion Lines

An amplitude containing **external fermions** has the form

$$\mathcal{M} = \sum_{i=1}^{n_F} c_i F_i \quad \text{where} \quad F_i = \text{(Product of)} \langle u | \Gamma_i | v \rangle .$$

n_F = number of fermionic structures.

Textbook procedure: **Trace Technique**

$$|\mathcal{M}|^2 = \sum_{i,j=1}^{n_F} c_i^* c_j F_i^* F_j$$

where $F_i^* F_j = \langle v | \bar{\Gamma}_i | u \rangle \langle u | \Gamma_j | v \rangle = \text{Tr}(\bar{\Gamma}_i | u \rangle \langle u | \Gamma_j | v \rangle \langle v |) .$

Problems with the Trace Technique

PRO: Trace technique is independent of any representation.

CON: For n_F F_i 's there are n_F^2 $F_i^* F_j$'s.

Things get worse the more vectors are in the game:
multi-particle final states, polarization effects . . .

Essentially $n_F \sim (\# \text{ of vectors})!$ because all
combinations of vectors can appear in the Γ_i .

Solution: Use Weyl-van der Waerden spinor formalism to
compute the F_i 's directly.



Sigma Chains

Define **Sigma matrices** and **2-dim. Spinors** as

$$\begin{aligned}\sigma_\mu &= (\mathbb{1}, -\vec{\sigma}), & \langle u|_{4d} &\equiv (\langle u_+|_{2d}, \langle u_-|_{2d}), \\ \bar{\sigma}_\mu &= (\mathbb{1}, +\vec{\sigma}), & |v\rangle_{4d} &\equiv \begin{pmatrix} |v_- \rangle_{2d} \\ |v_+ \rangle_{2d} \end{pmatrix}.\end{aligned}$$

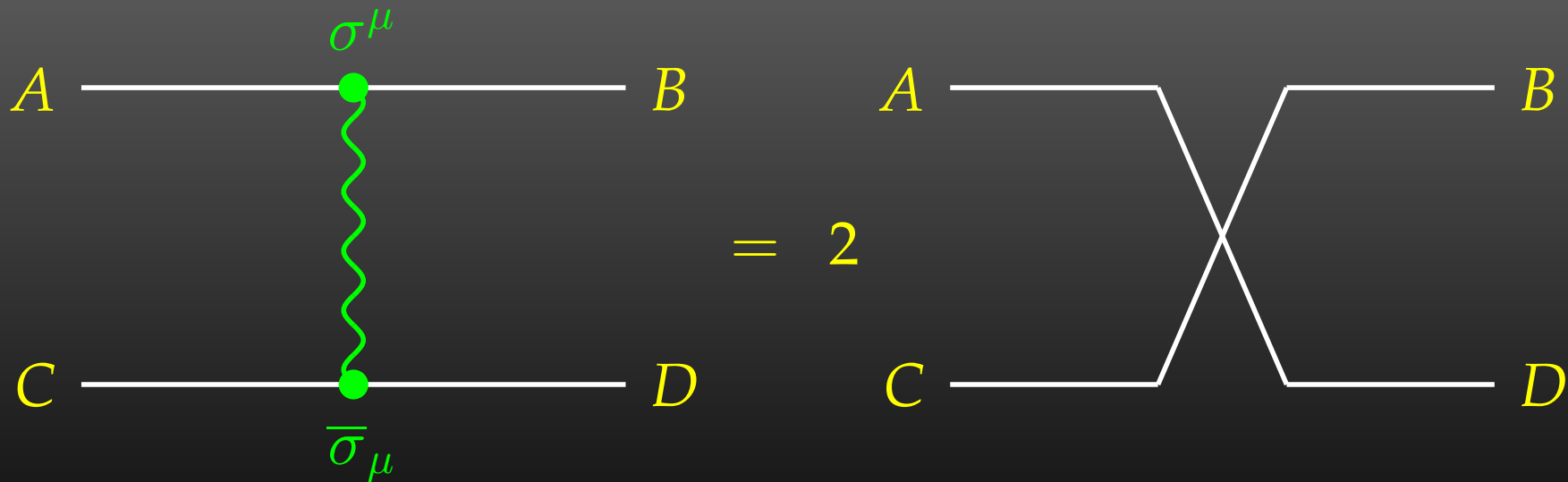
Using the chiral representation it is easy to show that **every chiral 4-dim. Dirac chain** can be converted to a **single 2-dim. sigma chain**:

$$\begin{aligned}\langle u| \omega_- \gamma_\mu \gamma_\nu \cdots |v\rangle &= \langle u_-| \bar{\sigma}_\mu \sigma_\nu \cdots |v_\pm\rangle, \\ \langle u| \omega_+ \gamma_\mu \gamma_\nu \cdots |v\rangle &= \langle u_+| \sigma_\mu \bar{\sigma}_\nu \cdots |v_\mp\rangle.\end{aligned}$$

Fierz Identities

With the Fierz identities for sigma matrices it is possible to **remove all Lorentz contractions** between sigma chains, e.g.

$$\langle A | \sigma_\mu | B \rangle \langle C | \bar{\sigma}^\mu | D \rangle = 2 \langle A | D \rangle \langle C | B \rangle$$



Implementation

- **Objects (arrays):** $|u_{\pm}\rangle \sim \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \quad (\sigma \cdot k) \sim \begin{pmatrix} a & b \\ c & d \end{pmatrix}$
- **Operations (functions):**

$$\langle u | v \rangle \sim (u_1 \ u_2) \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \quad \text{SxS}$$

$$(\overline{\sigma} \cdot k) |v\rangle \sim \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \quad \text{VxS, BxS}$$

Sufficient to compute any sigma chain:

$$\langle u | \sigma_{\mu} \overline{\sigma}_{\nu} \sigma_{\rho} | v \rangle k_1^{\mu} k_2^{\nu} k_3^{\rho} = \text{SxS}(u, \text{VxS}(k_1, \text{BxS}(k_2, \text{VxS}(k_3, v))))$$

More Freebies

- Polarization does not ‘cost’ extra:
= Get spin physics for free.
- Better numerical stability because components of k^μ are arranged as ‘small’ and ‘large’ matrix entries, viz.

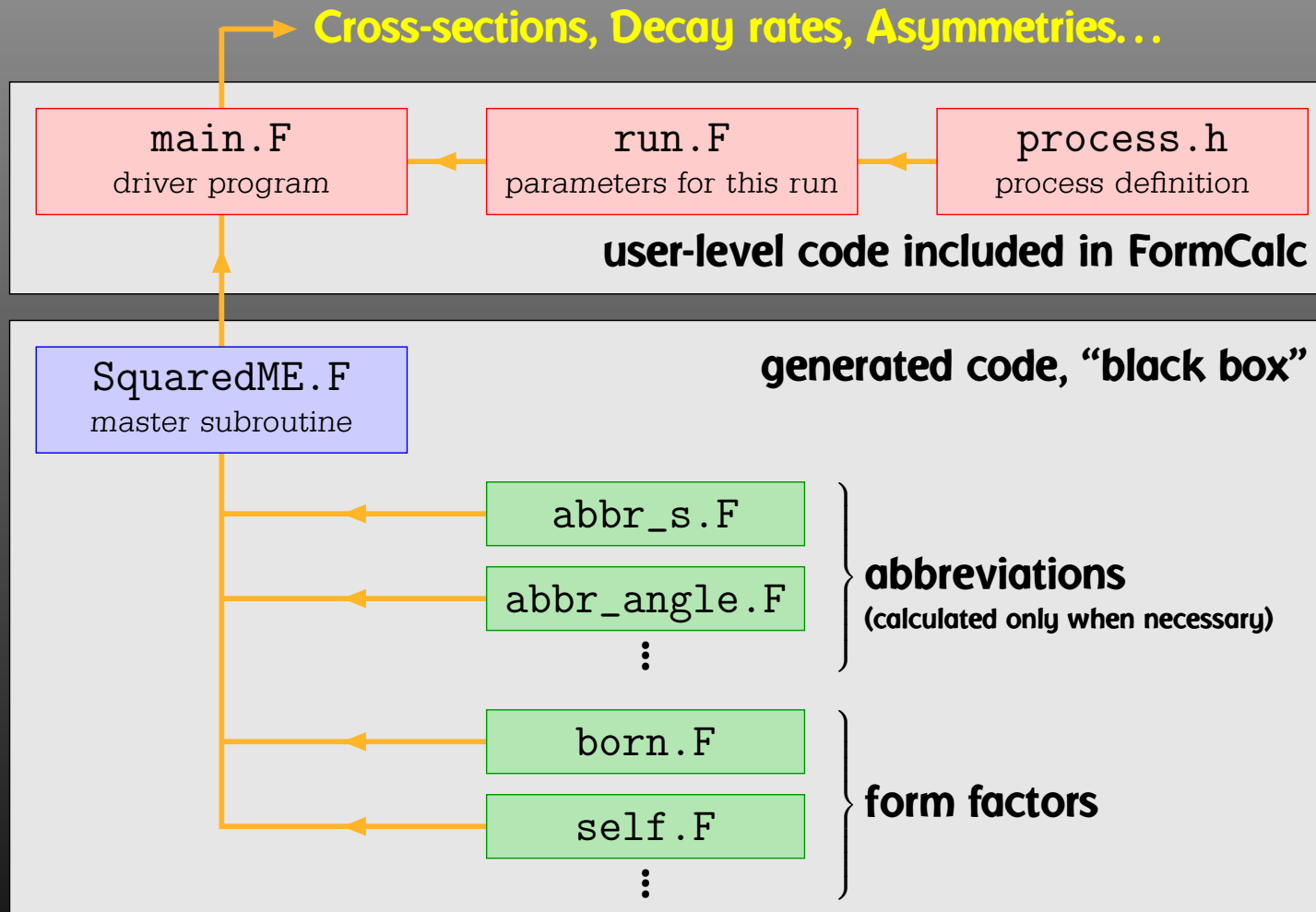
$$\sigma_\mu k^\mu = \begin{pmatrix} k_0 + k_3 & k_1 - ik_2 \\ k_1 + ik_2 & k_0 - k_3 \end{pmatrix}$$

\downarrow

Large cancellations of the form $\sqrt{k^2 + m^2} - \sqrt{k^2}$ when $m \ll k$ are avoided: better precision for mass effects.



Numerical Evaluation in Fortran 77



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

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, MG1, 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 “embarrassingly parallel” – each pass of the loop can be calculated independently.

How to distribute the iterations automatically if the loops are

a) user-defined b) usually nested?

Solution: Introduce a serial number



Unraveling Parameter Scans

```
subroutine ParameterScan( range )
integer serial
serial = 0
LOOP1
LOOP2
  :
  serial = serial + 1
  if( serial  $\notin$  range ) goto 1
  (calculate cross-section)
1 continue
end
```

Distribution on N machines is now simple:

- Send serial numbers $1, N + 1, 2N + 1, \dots$ on machine 1,
- Send serial numbers $2, N + 2, 2N + 2, \dots$ on machine 2, etc.

Shell-script Parallelization

Parameter scans can automatically be distributed on a cluster of computers:

- The **machines are declared in a file .submitrc, e.g.**

```
# Optional: Nice to start jobs with
nice 10
# Pentium 4 3000
pcl301
pcl301a
pcl305
# Dual Xeon 2660
pcl247b 2
pcl319a 2
pcl321 2
...
```

- The command line for distributing a job is *exactly the same* except that **submit is prepended**, e.g.

```
submit run uuuu 0,1000
```

HadCalc

HadCalc is a new front-end for FormCalc, i.e. it uses the generated Fortran code with a custom set of driver programs.

- Automates the convolution with PDFs.
- Hadronic cross-sections can be computed
 - ▷ Fully integrated,
 - ▷ Differential in Invariant mass,
Rapidity,
Transverse momentum.
- Cuts can be applied on
 - ▷ Rapidity,
Transverse momentum,
Jet separation.
- Operates in interactive or batch mode.

HadCalc is not (yet) public. It can currently be obtained from Michael Rauch [⟨mrauch@mppmu.mpg.de⟩](mailto:mrauch@mppmu.mpg.de).

Summary and Outlook

- 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.



Exercise

- **Download the FeynInstall script** from <http://www.feynarts.de> and use it to install FeynArts, FormCalc, and LoopTools on your account.
- **Run some of the demo programs** in the `FormCalc/examples` directory. Have a look at the generated Fortran code and run this, too. Make a plot of the data.
- **Modify one of the examples** for another scattering process, e.g. try to figure out the elastic neutralino-neutralino cross-section in the MSSM. This is a quantity relevant for dark matter physics.

